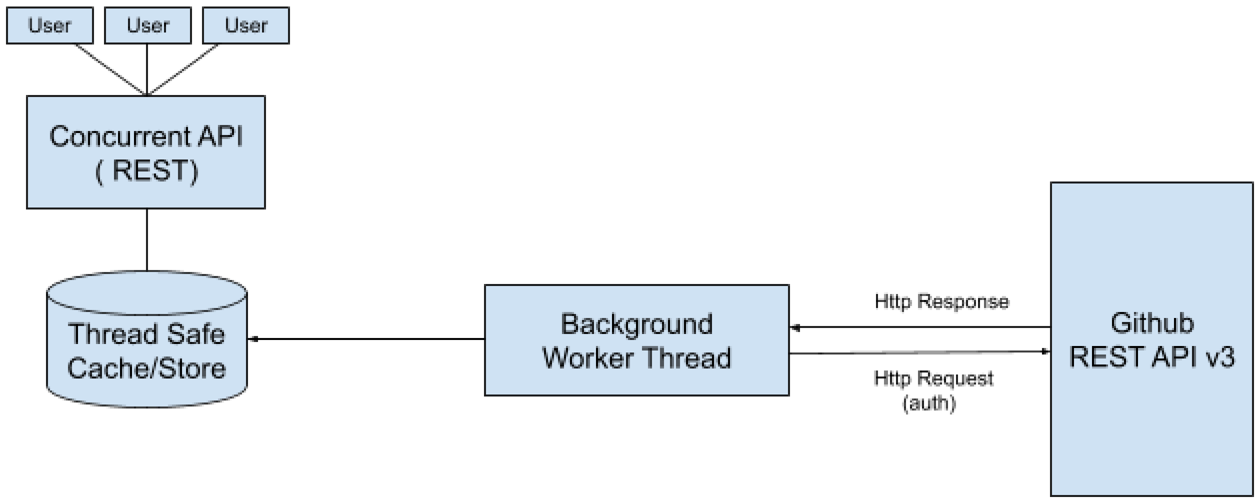
Simple Git Server

Overview:

Following sample application is written to demonstrate the ability to create an asynchronous API server, query an in-memory cache, query Github Rest API with an authorization token and maintain and update a local store using a background worker thread. There are three main components of the program:

1. An in-memory cache for storing some interesting facts about Github repositories. A thread safe global hash table is used for this purpose.
2. A background worker thread that periodically connects to Github REST API v3, and uses a sync algorithm to update the in-memory cache. Please note, Github API has a limit of only 11 requests per minute for unauthorized calls. For authorized calls, Github API has a limit of 30 requests per minute. To get this benefit, the program makes authorized calls every five seconds by passing an authorization token in the HTTP request header.
3. A simple Rest API that provides concurrent access to some facet of the cached data.



Steps to run:

1. Clone the repository.
2. Open the Github.Api.sln file in Visual Studio 2019 Version 16.4+
3. Set Git.Data.Api as a startup project.
4. Add your personal access token to appsettings.json.
5. Build and Run.
6. Use postman to test the following GET end point. GET https://localhost:5001/v1/api/Repos?description=Apollo Note: Change port from 5001 to your port if necessary.

Some non-functional features:

1. The program is easy to understand.
2. It is easy to enhance, extend and maintain.
3. It is written so that merge conflicts are minimal.
4. It is fast, querying in-memory cache is 200+ times faster than the Github API call.
5. It is platform independent.
6. It is easy to test and debug.
7. Supports logging and elegant error handling.

Absent features and possible disadvantages:

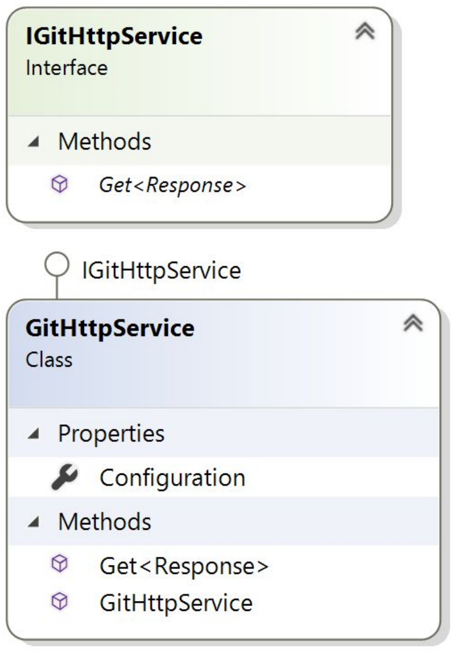
1. In-Memory space is limited.
2. Cache cannot be distributed over multiple nodes.
3. Cache could go stale. Refresh happens every 5 seconds.
4. Cache maintains only 30 most popular repositories of assembly language.
5. API is exposing only one simple GET endpoint for demonstration purposes only but capable of extending to more functionality as needed.
6. Token is kept in the configuration.

Program uses some clean code principles like:

1. Dependency injection.
2. Interface segregation.
3. Single responsibility.
4. Separation of concerns.
5. Generality and componentization.

Classes:

1. **GitHttpService**: I have implemented a generic class that connects to Github Rest API , gets the Json data and deserializes the data into entity objects. Entity objects are explained later in this document. It fetches the credentials from the injected configuration. Due to its generic nature, it can handle multiple types of request and response objects. It exposes a simple Get method to the client clases.

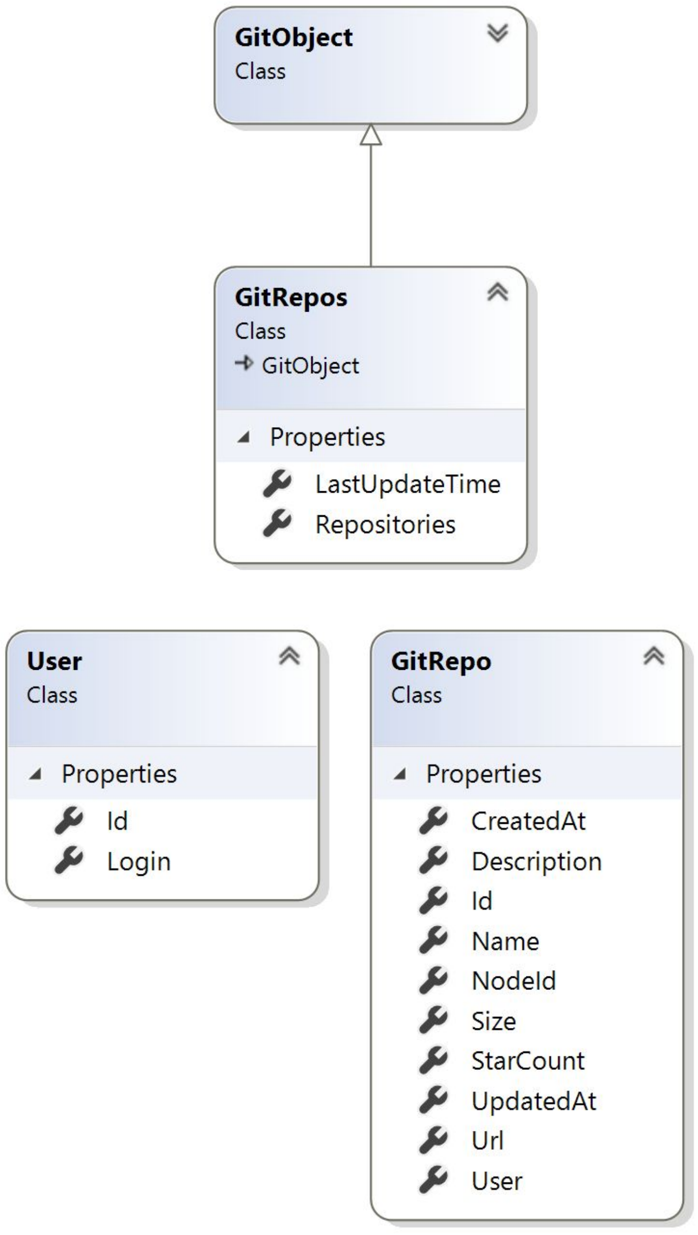


2. **GitObject**: It is a simple abstract class. Json data that is returned from Github API is deserialized into this object model.

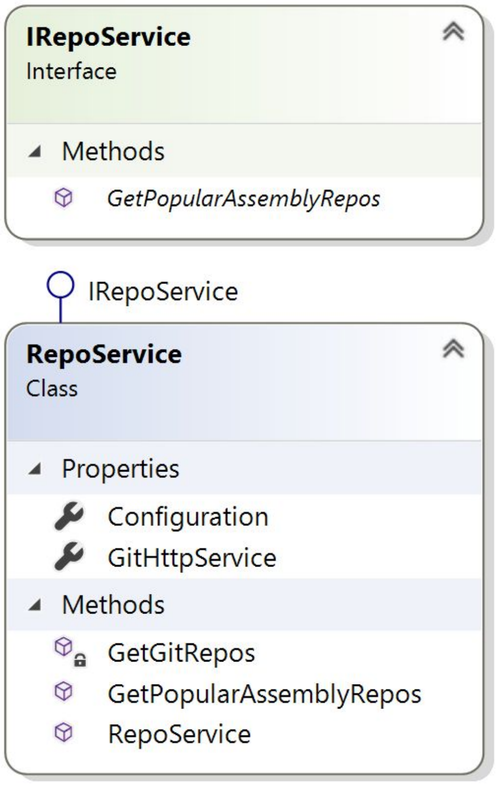
3. **GitRepos**: It is a simple class to encapsulate the repository data that is returned from Github. It contains the list of repositories.

4. **GitRepo:** GitRepo is used to encapsulate some attributes of a GitHub repository as shown below.

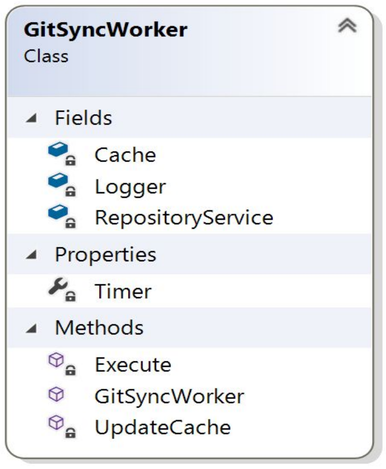
5. **User:** Class contains the id and login of the user who owns the repository.



RepoService: This class consumes the **GitHttpService** object described above using dependency injection. It builds a query to return the most popular repositories of assembly language. Interestingly, the original Apollo 11 Guidance Computer (AGC) source code for the command and lunar modules came up in the search so I named this document Apollo 11.



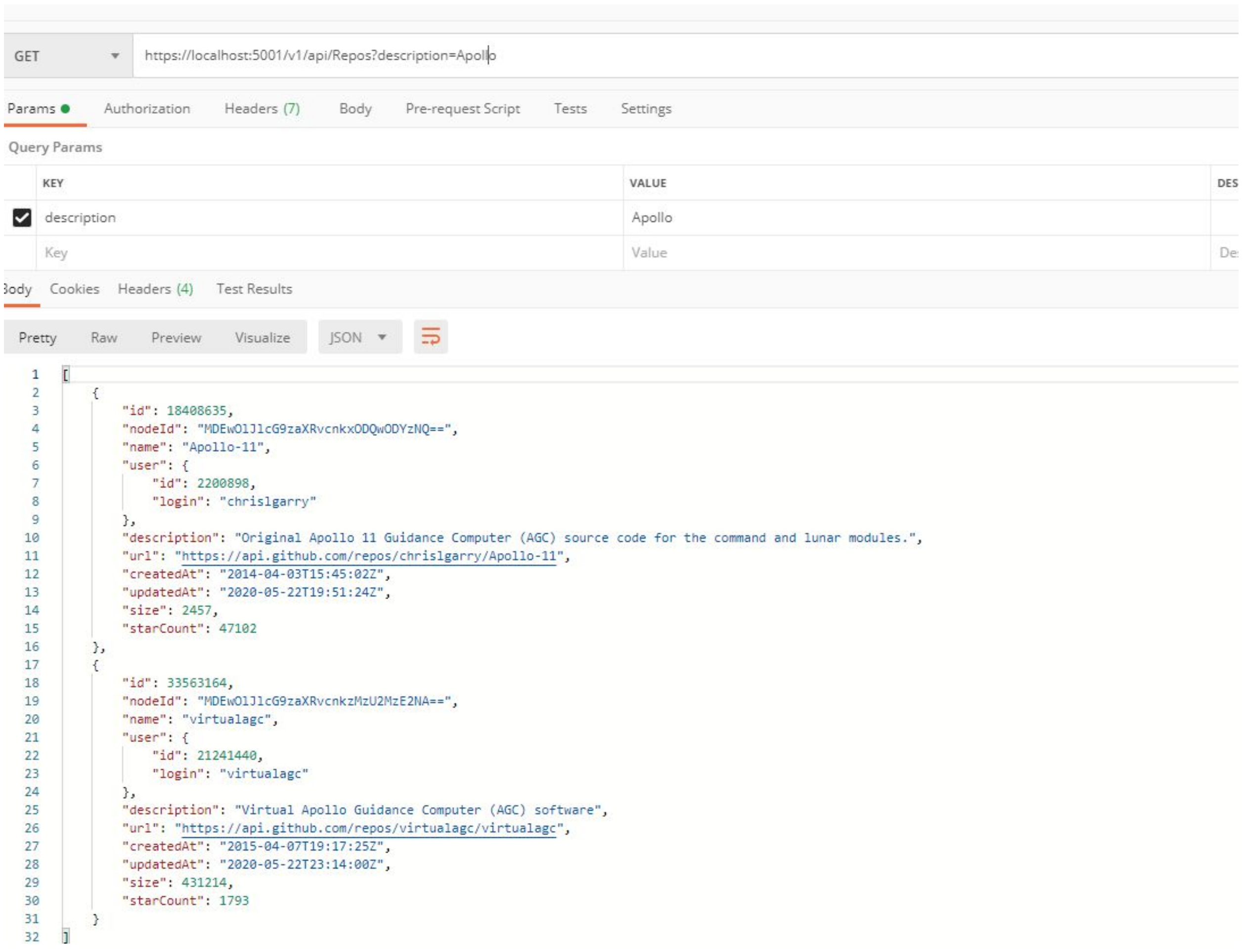
GitSyncWorker: This is one of the important classes in the program. It leverages the functionality of RepoService class and gets the top 30 assembly repositories.



Continued: After receiving the top thirty assembly repositories from Github, it updates the cache i.e. thread safe in-memory hash table. If an older version of the repository exists in the cache then it is replaced with the newer version. If the repository does not exist in the cache, the new repository is added to the cache. Other repositories removed from the cache. I am using the framework Timer construct to fire the fire update cache functionality.

ReposController: The ReposController class implements the HttpGet method. It returns all repositories from the cache to the user. Users can provide the query string to filter the repositories by description. Following is an example when you call GET method from Postman:

v1/api/Repos?description=Apollo



KEY	VALUE	DES
✓ description	Apollo	
Key	Value	De

```
[
    {
        "id": 18408635,
        "nodeId": "MDEwOlJlcG9zaXRvcnkxODQwODYzNQ==",
        "name": "Apollo-11",
        "user": {
            "id": 2200898,
            "login": "chrislgarry"
        },
        "description": "Original Apollo 11 Guidance Computer (AGC) source code for the command and lunar modules.",
        "url": "https://api.github.com/repos/chrislgarry/Apollo-11",
        "createdAt": "2014-04-03T15:45:02Z",
        "updatedAt": "2020-05-22T19:51:24Z",
        "size": 2457,
        "starCount": 47102
    },
    {
        "id": 33563164,
        "nodeId": "MDEwOlJlcG9zaXRvcnkzMzU2MzE2NA==",
        "name": "virtualagc",
        "user": {
            "id": 21241440,
            "login": "virtualagc"
        },
        "description": "Virtual Apollo Guidance Computer (AGC) software",
        "url": "https://api.github.com/repos/virtualagc/virtualagc",
        "createdAt": "2015-04-07T19:17:25Z",
        "updatedAt": "2020-05-22T23:14:00Z",
        "size": 431214,
        "starCount": 1793
    }
]
```

Dependency Injection mapping: The dependency injection mapping is located in Start class that is executed during application bootstrapping. Last but not least,

Tools and Technologies: I used C#, .Net core framework 3.1 and Visual Studio 2019 to build the application. Code is located in the GitHub repository. Project contains other boilerplate code and auto generated code that can be ignored.