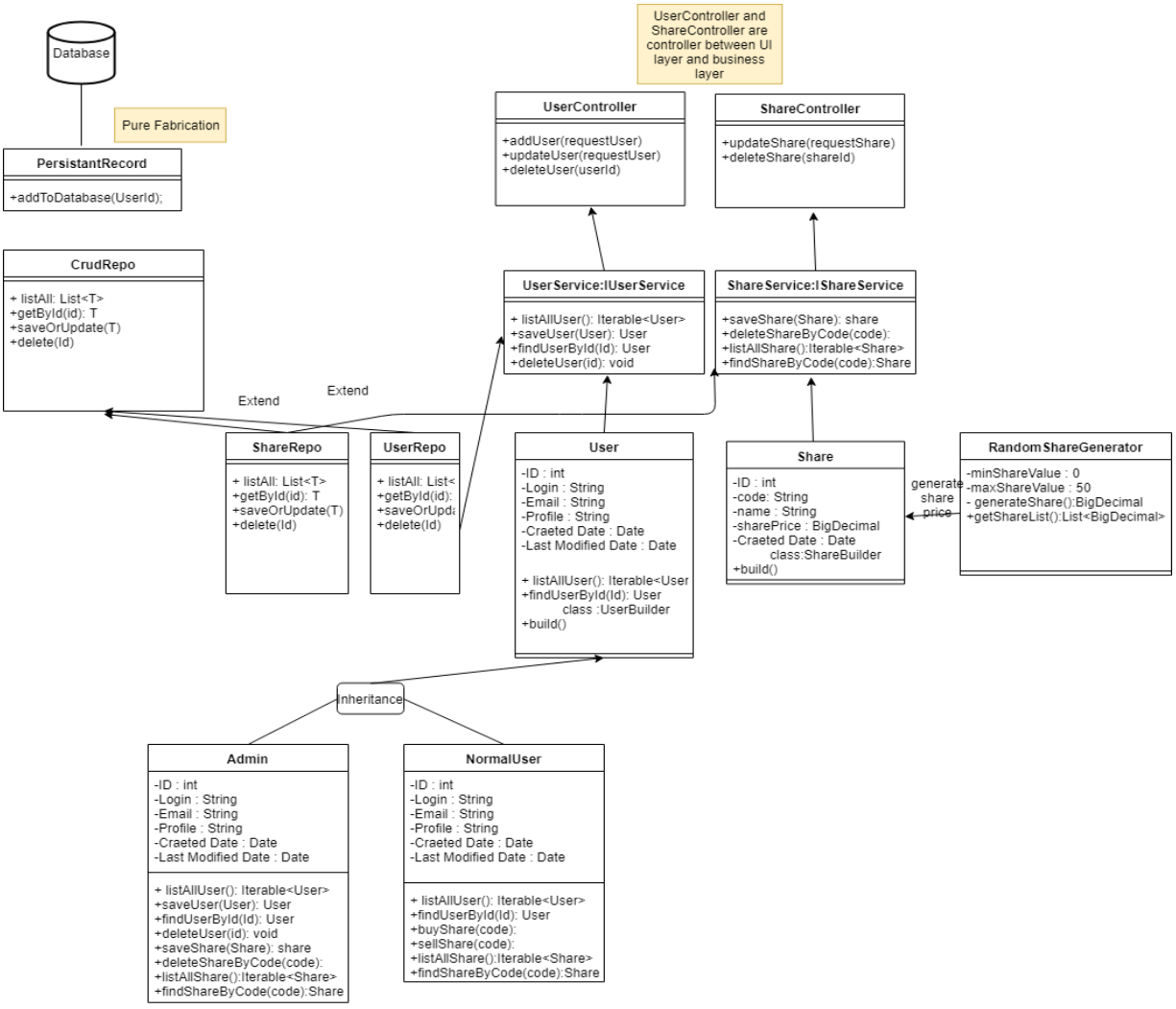
Stock Market App

Stock Market Project

The stock market app is a REST application that allow to sell or buy share and calculates best profit interval.

In this project, assume that admin user is created in a database. Admin can make some user operations via UserController, he/she can add a new user to the system, delete them or update their information. UserController connect to the interface which name is IUserService. Under favour of this interface, admin can achieve user operations.

In addition to this, Admin can create shares of companies and also delete these shares. User operations and share operations are performed by their services. UserService and ShareService use userRepo and shareRepo which are extends from CRUDRepo. It makes basic CRUD operations. By means of services and repos, share is being written on the database.



Generate Random Share Price

This part of the code aim that generate a random variable between preconcerted values. Random variables are added to the arrayList. ArrayList is chosen for this operation because after the share price generation is done, max profit is calculated. This process include some comparison between element of the arrays and a lot of use of get(arrayIndex) method. Get() method is performed on linkedList with O(n) but on arrayList just O(1). This is why arrayList is used for generating random share price.

Maximum Profit Algorithm

Calculating the maximum profit of shares can be achieved with three different ways.

First of all, according to the brute force method two different for loop can be used and outer loop control element as taking share index and compare it inner loops elements. This method uses two different for loop so time complexity of algorithm is $O(n^2)$.

Another method is divide and conquer method for finding maximum profit. The main approach of the divide and conquer method is splitting the input array into half subarray until each half array contains just one element. Getting the values of each subarray take O(1) complexity. Merging subarrays into the one array and taking maximum profit value from these merged array takes O(lg n) time complexity according to the Master Theorem. Divide and conquer method is performed via recursive calls.

The last method is dynamic programming which time complexity is O(n). Kadane's Algorithm is a dynamic programming approach algorithm which calculates the maximum sum from the subarray. Scan the array from left to right and calculates the sum of subarray. If sum of subarray is less than zero then take new subarray. Similar to Kadane's algorithm maximum profit problem can be solved using this approach. Scan array until reach the last element. At each iteration, update the profit according to the calculating minimum and maximum value. If iteration profit is better than the previous one, updating the profit as a new value. Each step take O(1) time complexity for compare operation and O(n) time complexity for scanning all array elements. For this reason, in this project dynamic programming approach is implemented.

Bilal Emre Gülşen