Документация и комментарии к реализации тестового задания

Тестовое задание для Frontend-разработчика на Angular

Описание задачи:

Вам необходимо создать Angular-приложение, которое работает с потоками данных котировок валют в реальном времени. Приложение должно получать обновления котировок с сервера с высокой частотой и отображать их на пользовательском интерфейсе в виде таблицы.

Выбор протокола обмена данными с провайдером данных

Для решения задания был выбран протокол WebSocket, в силу его ориентации на обмен данными в режиме реального времени. В отличии от HTTP отсутствует необходимость запроса данных с сервера, что дает возможность ускорить обмен данными.

С точки зрения развития компонента, видится разумным реализация обоих протоколов, при котором WebSocket будет основным, а HTTP резервным.

Подготовка к разработке приложения и тестирование

Реализован тестовый сервер на базе Node JS, который передает котировки в нужном формате через протокол WebSocket.

Список кодов инструментов храниться в виде массива строк в модуле symbols-list.js

Сервер и вся бизнес логика сервера реализованы в модуле ws-server.js

При необходимости смоделировать поток котировок серверу можно передать в качестве параметров время жизни потока, периодичность отправки данных клиентам и количество инструментов в потоке.

На основании переданных параметров, сервер формирует список инструментов из модуля symbols-list.js и для каждого инструмента определяется случайным образом будет передаваться котировка в этом обновлении или нет. Если котировка передается, то опять же случайным образом формируется котировка bid и на основании нее котировка ask (случайное отклонение). Данные рассылаются клиентам в виде массива json объектов

Клиент может запросить начало передачи данных или остановить поток

Компоненты приложения

src\app\rt-quotes-table\rt-quotes-table.component - компонент интерфейса пользователя: таблица, фильтр на инструменты, кнопка подключения к потоку

src\app\quotes-data.service.ts - сервис по обработке потока котировок

Реализация интерфейса

Чтобы обеспечить работу под высокой нагрузкой и оптимизировать производительность был реализована облегченная версия интефейса, состоящая из трех элементов:

1. Соединение с источником данных - кнопка Connect to WS - подключение к потоку котировок. После подключения к потоку функция кнопки меняется на Stop WS - отключение от потока данных
2. Фильтр списка - Filter - поле для фильтрации списка получаемого списка котировок. Пользователь вводит список кодов интересующих его кодов инструментов через запятую. Фильтр снимается при очистке поля.
3. Таблица котировок - список вместе с заголовком стилизованный под таблицу. Список полей таблицы:
    - Symbol - код инструмента
    - Bid - котировка на покупку инструмента
    - Ask - котировка на продажу инструмента
    - Time - время котировок

Оптимизация интерфейса и потока данных

Все компоненты реализуют стратегию ChangeDetectionStrategy.OnPush для избежания излишних циклов проверки изменений, а таблица котировок использует trackBy функцию директивы ngFor для оптимизации рендеринга. Функция trackBy проверят необходимость обновления записи на основании изменения двух полей: Symbol и Time.

Изначально был разработан интерфейс на основании MatTable из библиотеки Material. Но в рамках анализа рендеринга через Chrome DevTools, при высокой нагрузке, компонент обновлял излишнее количество элементов и тратил больше ресурсов на change detection.

Поэтому было принято решение отказаться от лишней библиотеки и реализовать таблицу на основании встроенного списка, стилизованного под таблицу. По визуальному восприятию компоненты не отличаются, однако список производит стабильный выборочный рендеринг только вновь полученных котировок.

Поток данных управляется библиотекой RxJS и буферизуется с использованием оператора throttleTime(500). Данный подход позволяет убрать чрезмерное мерцание таблицы и снизить нагрузку на интерфейс.

Поток котировок буферизируется через массив кеша, в который при каждом обновлении данных записываются новые пары котировок и добавляются, котировки по которым не пришло обновления. По истечении таймера буферизации источник данных обновляется, добавляя новые инструменты, и обновляя элементы массива для существующих, чтобы оптимизировать рендеринг.

Интерфейс работает стабильно при высоких нагрузках

Тестирование

Приложение настроено на тестирование с использование фреймворка jest.js.

Для проведения юнит тестов необходимо запустить тестовый сервер ws-server.js. В дальнейшем юнит тест самостоятельно управляет созданием и закрытие потока котировок.

Основные тесты на консистентность данных и работу при высоких нагрузках реализованы в файле src\app\rt-quotes-table\rt-quotes-table.component.ts

'receiving data with high frequency testing' - тест на нагрузку

'data consistency check bid equal or less then ask quote' - тест на консистентность

Тесты и код прокомментированы.