Міністерство освіти і науки України   
Національний авіаційний університет   
Факультет кібербезпеки комп’ютерної та програмної інженерії  
Кафедракомп’ютеризованих систем управління

Домашнє завдання   
 з дисципліни «Інтернет технології»   
 на тему «Додаток на Node.js, шо працює в реальному часі»

Виконав:   
студент групи СП-425 Щербина Р.І. Перевірив:   
Сябрук І.М.

Київ 2019

**Опис додатку**

Додаток представляє собою SPA (Single Page Application), серверна частина якого отримує різноманітні дані по криптовалютам через API запити.

Клієнтська частина додатку приймає відсортовані дані з серверу, та виводить їх у зручному форматі карток (кожна картка містить інформацію про окрему криптовалюту). API запити на сервері відбуваються кожні 30 секунд, які одразу ж відправляються клієнту та оновлюються у режимі реального часу.

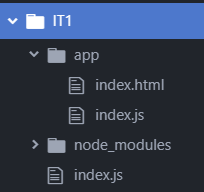
**Використані інструменти**

**Socket.IO** — [JavaScript](https://uk.wikipedia.org/wiki/JavaScript" \o "JavaScript)-[бібліотека](https://uk.wikipedia.org/wiki/%D0%91%D1%96%D0%B1%D0%BB%D1%96%D0%BE%D1%82%D0%B5%D0%BA%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC) для [веб-застосунків](https://uk.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D0%B7%D0%B0%D1%81%D1%82%D0%BE%D1%81%D1%83%D0%BD%D0%BE%D0%BA) і обміну даними в реальному часі. Складається з двох частин: клієнтської, яка запускається в [браузері](https://uk.wikipedia.org/wiki/%D0%91%D1%80%D0%B0%D1%83%D0%B7%D0%B5%D1%80" \o "Браузер) і серверної для [node.js](https://uk.wikipedia.org/wiki/Node.js). Обидва компоненти мають схожий [прикладний програмний інтерфейс](https://uk.wikipedia.org/wiki/%D0%9F%D1%80%D0%B8%D0%BA%D0%BB%D0%B0%D0%B4%D0%BD%D0%B8%D0%B9_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BD%D0%B8%D0%B9_%D1%96%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81" \o "Прикладний програмний інтерфейс). Подібно node.js, Socket.IO [подієво-орієнтована](https://uk.wikipedia.org/wiki/%D0%9F%D0%BE%D0%B4%D1%96%D1%94%D0%B2%D0%B0_%D0%B0%D1%80%D1%85%D1%96%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0" \o "Подієва архітектура).

**Express** — [програмний каркас](https://uk.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BD%D0%B8%D0%B9_%D0%BA%D0%B0%D1%80%D0%BA%D0%B0%D1%81" \o "Програмний каркас) розробки серверної частини [веб-застосунків](https://uk.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D0%B7%D0%B0%D1%81%D1%82%D0%BE%D1%81%D1%83%D0%BD%D0%BE%D0%BA) для [Node.js](https://uk.wikipedia.org/wiki/Node.js), реалізований як [вільне і відкрите програмне забезпечення](https://uk.wikipedia.org/wiki/%D0%92%D1%96%D0%BB%D1%8C%D0%BD%D0%B5_%D1%96_%D0%B2%D1%96%D0%B4%D0%BA%D1%80%D0%B8%D1%82%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BD%D0%B5_%D0%B7%D0%B0%D0%B1%D0%B5%D0%B7%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%BD%D1%8F" \o "Вільне і відкрите програмне забезпечення) під [ліцензією MIT](https://uk.wikipedia.org/wiki/%D0%9B%D1%96%D1%86%D0%B5%D0%BD%D0%B7%D1%96%D1%8F_MIT" \o "Ліцензія MIT). Він спроектований для створення веб-застосунків і [API](https://uk.wikipedia.org/wiki/API).

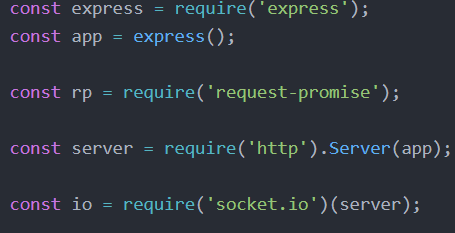
**Mustache** – система веб-шаблонів. У даному випадку використана JavaScript реалізація.

**Реалізація додатку**

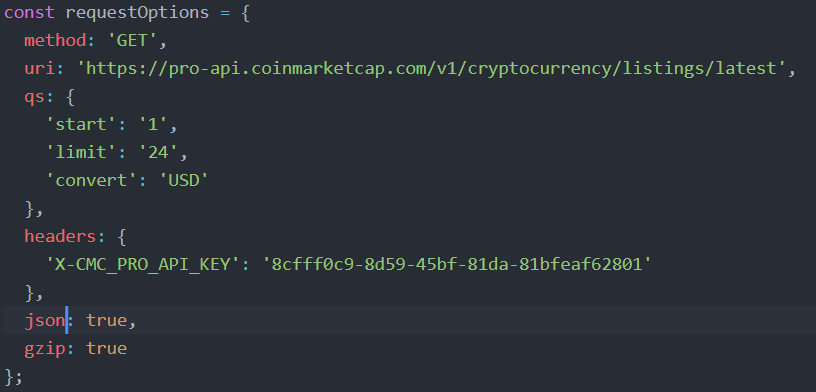
Спочатку було створено файлову ієрархію додатку такого вигляду:   


Після ініціалізації додатку за допомогою команди npm init, з’являється також файл package.json з усіма залежностями нашого проекту. Зовнішній файл index.js містить лише 1 рядок «require("./app/index.js")». Більше ми цей файл не чіпаємо, а будемо використовувати для запуску додатку.

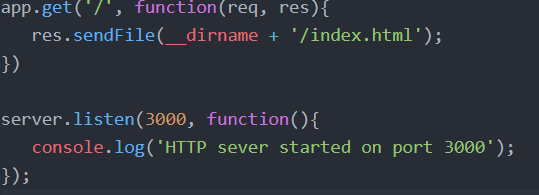
Переходимо до редагування внутрішнього файлу index.js. Спочатку підключаємо усі необхідні нам пакети, які спочатку булі завантажені за допомогою команди npm install назва\_пакету.



Далі створюємо налаштування API запиту.



Тепер прописуємо початкову логіку сервера. Перші 3 рядки повертають клієнту сторінку index.html при переході на початкову адресу нашого додатку.

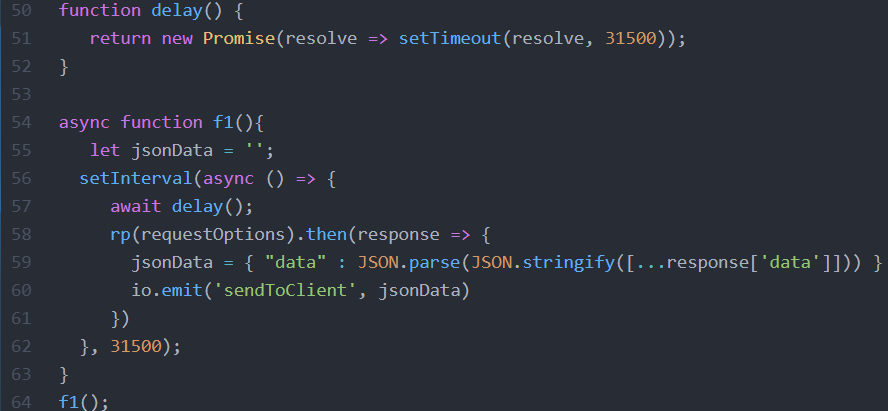


Тепер починається робота з пакетом Socket.io. Рядок 39 створює подію «sendToClient», та відправляє клієнту API дані, які містить зміна jsonData.

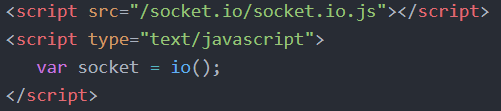


45 рядок прослуховує подію «receivedFromClient», яка приймає дані від користувача та виводить їх у консоль сервера.

Далі створюємо функції, що забезпечують регулярне повторення API запиту. Функція f1() також відправляє дані клієнту.



Сервер налаштований. Тепер переходимо до клієтської частини. У створеному файлі index.js прописуємо наступні рядки, що слугують для з’єднання клієнта з сервером.



У кінці тегу <body> прописуємо ще один скрипт. Він буде прослухувати подію  
«sendToClient», приймаючи дані, що надсилаються сервером.



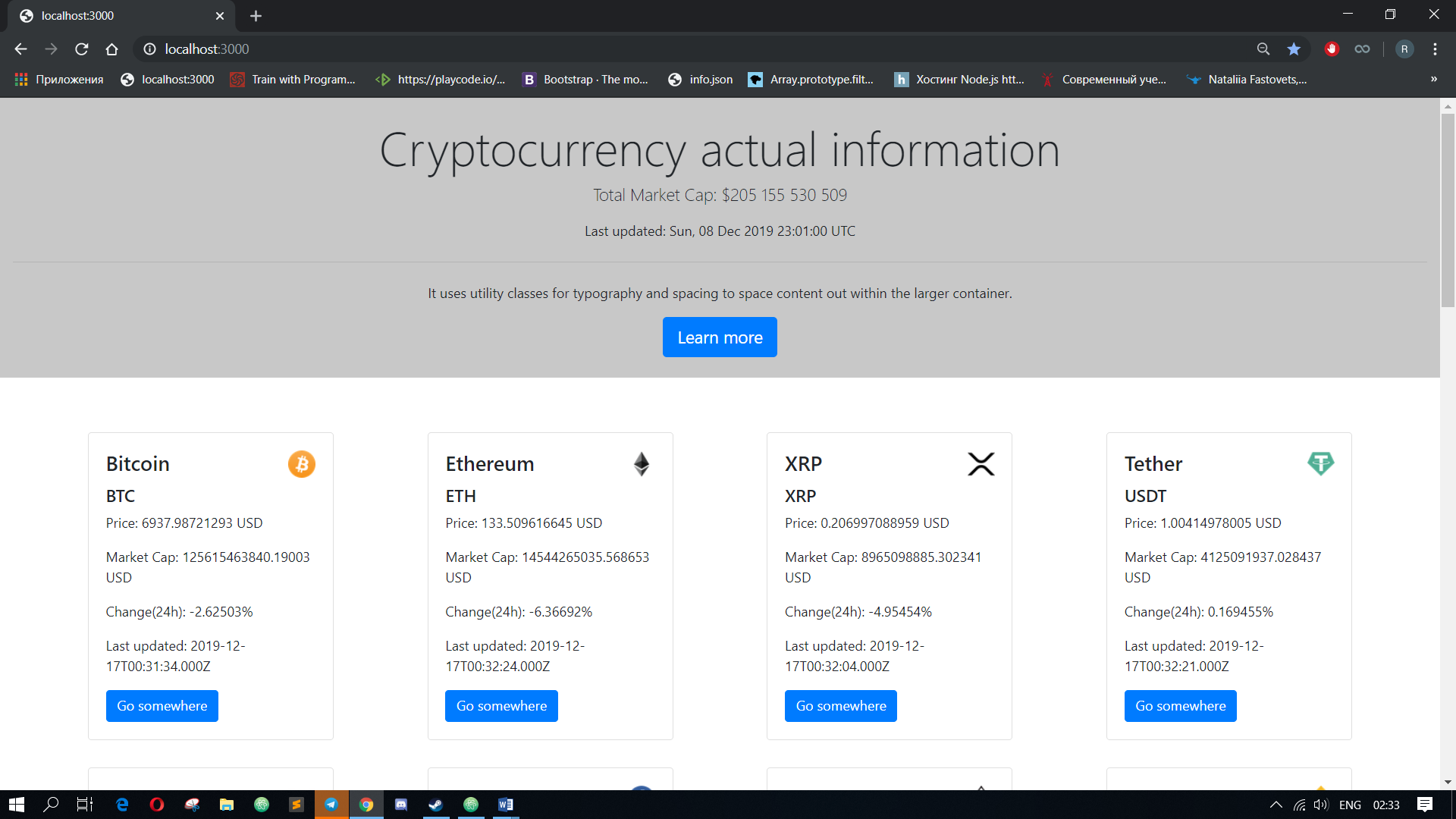
Рядки 80 81 82 розподіляють отримані дані по Mustache шаблону, код якого буде приведено нижче. У рядку 74 створюється об’єкт Date(), з якого отримується поточний час на машині клієнта. Отриманий час відправляється на сервер кожного разу, коли клієнт отримує оновлені API дані, у рядку 85, де створюється подія «receivedFromClient».

Скрипт, в я кому ми прописуємо наш Mustache шаблон:



Для старту роботи додатку прописуємо в консолі команду node index. Щоб команда знайшла файл, працюючи в консолі, потрібно знаходитися у директорії, в якій знаходиться той самий зовнішній index.js, в якому ми прописували рядок require("./app/index.js").

Погравшись зі стилями, ми отримали наступний результат на стороні клієнта:



**Висновок:**

Під час виконання даного завдання ми розглянули приклад створення Node.js додатку, що працює в режимі реального часу. Даними що оновлюються послужила інформація статистики криптовалют, яку ми отримуємо через АPI, та використовуємо у форматі JSON.