МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

Факультет прикладної математики

Кафедра програмного забезпечення комп’ютерних систем

**Практична робота № 4**

з дисципліни “**Web-технології та розробка web-систем**”

Виконав:

студент III курсу

групи КП-21

Бураков Д. С.

Прийняв:

доцент Дичка А.І.

Київ 2025

Налаштування Casdoor та веб-сервісу аналогічні до лаби3, так як дана робота робиться на їх основі

Змінні які використовуються, збережені в application.properties:

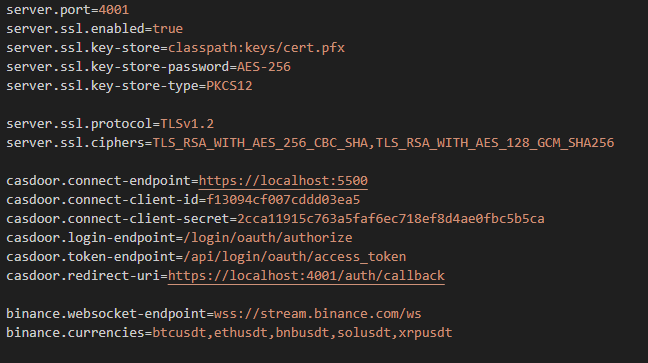


Рис. 1. Визначення змінних в application.properties

Entity-клас для представлення сутності в системі:

package lab4.lab4.Binance;

public class PriceUpdate {

private final String symbol;

private final String price;

public PriceUpdate(String symbol, String price) {

this.symbol = symbol;

this.price = price;

}

public String getSymbol() {

return symbol;

}

public String getPrice() {

return price;

}

@Override

public String toString() {

return String.format("PriceUpdate{symbol='%s', price='%s'}", symbol, price);

}

}

Налаштування підключення до Binance:

package lab4.lab4.Binance;

import java.util.LinkedList;

import java.util.List;

import org.springframework.boot.context.properties.ConfigurationProperties;

import org.springframework.context.annotation.Configuration;

@Configuration

@ConfigurationProperties(prefix = "binance")

public class BinanceProperties {

private List<String> currencies = new LinkedList<>();

public List<String> getCurrencies() {

return currencies;

}

public void setCurrencies(List<String> currencies) {

this.currencies = currencies != null ? currencies : new LinkedList<>();

}

}

Налаштування відправки через Protobuf:

package lab4.lab4.Binance;

import java.util.List;

import org.springframework.stereotype.Service;

import com.binance.connector.client.WebSocketStreamClient;

import com.binance.connector.client.impl.WebSocketStreamClientImpl;

import com.fasterxml.jackson.databind.DeserializationFeature;

import com.fasterxml.jackson.databind.JsonNode;

import com.fasterxml.jackson.databind.ObjectMapper;

import jakarta.annotation.PostConstruct;

import jakarta.annotation.PreDestroy;

@Service

public class BinanceWebSocketService {

private final BinanceProperties binanceProperties;

private final WebSocketStreamClient client;

private final TradeWebSocketHandler tradeWebSocketHandler;

private final ObjectMapper objectMapper = new ObjectMapper()

.configure(DeserializationFeature.FAIL\_ON\_UNKNOWN\_PROPERTIES, false);

public BinanceWebSocketService(BinanceProperties binanceProperties, TradeWebSocketHandler tradeWebSocketHandler) {

this.binanceProperties = binanceProperties;

this.tradeWebSocketHandler = tradeWebSocketHandler;

this.client = new WebSocketStreamClientImpl();

}

@PostConstruct

public void init() {

List<String> currencies = binanceProperties.getCurrencies();

if (currencies.isEmpty()) {

return;

}

currencies.stream().forEach(this::subscribeToTradeStream);

}

@PreDestroy

public void cleanup() {

client.closeAllConnections();

}

private void subscribeToTradeStream(String symbol) {

String lowerCaseSymbol = symbol.toLowerCase();

client.tradeStream(lowerCaseSymbol, event -> {

try {

JsonNode tradeEvent = objectMapper.readTree(event);

final String TRADE\_EVENT = "trade";

if (tradeEvent.has("e") && TRADE\_EVENT.equals(tradeEvent.get("e").asText()) &&

tradeEvent.has("s") && tradeEvent.has("p")) {

String receivedSymbol = tradeEvent.get("s").asText();

String price = tradeEvent.get("p").asText();

lab4.lab4.protobuf.PriceUpdateClass.PriceUpdate protoUpdate = lab4.lab4.protobuf.PriceUpdateClass.PriceUpdate

.newBuilder()

.setSymbol(receivedSymbol)

.setPrice(price)

.build();

tradeWebSocketHandler.broadcastPriceUpdate(protoUpdate);

} else {

System.out.println(String.format("Received non-trade event or unexpected format for %s: %s",

lowerCaseSymbol, event));

}

} catch (Exception e) {

System.out.println(

String.format("Error processing Binance trade event for %s: %s. Error: %s",

lowerCaseSymbol, event, e.getMessage()));

}

});

}

}

package lab4.lab4.Binance;

import java.io.IOException;

import java.util.List;

import java.util.concurrent.ConcurrentHashMap;

import java.util.concurrent.ConcurrentLinkedQueue;

import java.util.concurrent.CopyOnWriteArrayList;

import java.util.concurrent.ExecutorService;

import java.util.concurrent.Executors;

import java.util.concurrent.locks.Lock;

import java.util.concurrent.locks.ReentrantLock;

import org.springframework.stereotype.Component;

import org.springframework.web.socket.BinaryMessage;

import org.springframework.web.socket.CloseStatus;

import org.springframework.web.socket.WebSocketSession;

import org.springframework.web.socket.handler.BinaryWebSocketHandler;

import jakarta.annotation.PreDestroy;

import lab4.lab4.protobuf.PriceUpdateClass;

@Component

public class TradeWebSocketHandler extends BinaryWebSocketHandler {

private final List<WebSocketSession> sessions = new CopyOnWriteArrayList<>();

private final ConcurrentHashMap<String, ConcurrentLinkedQueue<BinaryMessage>> sessionQueues = new ConcurrentHashMap<>();

private final ExecutorService messageSenderExecutor = Executors.newCachedThreadPool();

private final Lock sendLock = new ReentrantLock();

private long lastSendTime = 0;

private static final long MIN\_SEND\_INTERVAL = 100; // мс

@Override

public void afterConnectionEstablished(WebSocketSession session) {

sessions.add(session);

System.out.println("WebSocket client connected: " + session.getId());

ConcurrentLinkedQueue<BinaryMessage> queue = new ConcurrentLinkedQueue<>();

sessionQueues.put(session.getId(), queue);

messageSenderExecutor.submit(() -> processSessionMessages(session, queue));

}

@Override

public void afterConnectionClosed(WebSocketSession session, CloseStatus status) {

sessions.remove(session);

System.out.println("WebSocket client disconnected: " + session.getId());

sessionQueues.remove(session.getId());

}

public void broadcastPriceUpdate(PriceUpdateClass.PriceUpdate protoUpdate) {

long currentTime = System.currentTimeMillis();

if (currentTime - lastSendTime < MIN\_SEND\_INTERVAL) {

return;

}

lastSendTime = currentTime;

byte[] bytes = protoUpdate.toByteArray();

BinaryMessage message = new BinaryMessage(bytes);

for (WebSocketSession session : sessions) {

if (session.isOpen()) {

ConcurrentLinkedQueue<BinaryMessage> queue = sessionQueues.get(session.getId());

if (queue != null) {

queue.offer(message);

}

}

}

}

private void processSessionMessages(WebSocketSession session, ConcurrentLinkedQueue<BinaryMessage> queue) {

while (session.isOpen()) {

BinaryMessage message = queue.poll();

if (message != null) {

try {

session.sendMessage(message);

} catch (IOException e) {

System.err.println("Failed to send message for session " + session.getId() + ": " + e.getMessage());

if (!session.isOpen()) {

sessions.remove(session);

sessionQueues.remove(session.getId());

break;

}

} catch (IllegalStateException e) {

System.err.println("State error sending message for session " + session.getId() + ": " + e.getMessage());

}

} else {

try {

Thread.sleep(20);

} catch (InterruptedException e) {

Thread.currentThread().interrupt();

System.err.println("Message sender thread interrupted for session " + session.getId());

break;

}

}

}

System.out.println("Message sender thread stopped for session " + session.getId());

}

@PreDestroy

public void shutdownExecutor() {

messageSenderExecutor.shutdown();

}

}

Конфігурація сокету, який відповідає за фронтенд:

package lab4.lab4.Binance;

import org.springframework.context.annotation.Configuration;

import org.springframework.web.socket.config.annotation.EnableWebSocket;

import org.springframework.web.socket.config.annotation.WebSocketConfigurer;

import org.springframework.web.socket.config.annotation.WebSocketHandlerRegistry;

@Configuration

@EnableWebSocket

public class WebSocketConfig implements WebSocketConfigurer {

private final TradeWebSocketHandler tradeWebSocketHandler;

public WebSocketConfig(TradeWebSocketHandler tradeWebSocketHandler) {

this.tradeWebSocketHandler = tradeWebSocketHandler;

}

@Override

public void registerWebSocketHandlers(WebSocketHandlerRegistry registry) {

registry.addHandler(tradeWebSocketHandler, "/ws/price")

.setAllowedOriginPatterns("\*");

}

}

Для роботи з Protobuf форматом, була створена така модель:

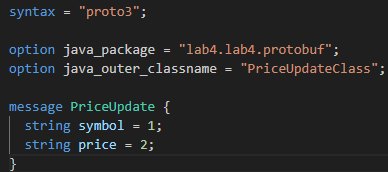


Рис. 2. Визначення модель для Protobuf

В файлі pom.xml були додані плагіни для автоматичного генерування вихідних файлів з .proto теплейту:

<plugin>

<groupId>org.xolstice.maven.plugins</groupId>

<artifactId>protobuf-maven-plugin</artifactId>

<version>0.6.1</version>

<configuration>

<protocArtifact>com.google.protobuf:protoc:4.31.0:exe:${os.detected.classifier}</protocArtifact>

<outputDirectory>${project.basedir}/src/main/java</outputDirectory>

<clearOutputDirectory>false</clearOutputDirectory>

<protoSourceRoot>${project.basedir}/src/client/proto</protoSourceRoot>

</configuration>

<executions>

<execution>

<phase>generate-sources</phase>

<goals>

<goal>compile</goal>

</goals>

</execution>

</executions>

</plugin>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-resources-plugin</artifactId>

<executions>

<execution>

<id>copy-protobuf-helper</id>

<phase>generate-resources</phase>

<goals>

<goal>copy-resources</goal>

</goals>

<configuration>

<outputDirectory>${project.basedir}/src/main/resources/static/js</outputDirectory>

<resources>

<resource>

<directory>${project.basedir}/src/client</directory>

<includes>

<include>protobuf-helper.js</include>

</includes>

</resource>

</resources>

</configuration>

</execution>

</executions>

</plugin>

<plugin>

<groupId>org.codehaus.mojo</groupId>

<artifactId>build-helper-maven-plugin</artifactId>

<version>3.4.0</version>

<executions>

<execution>

<id>add-source</id>

<phase>generate-sources</phase>

<goals>

<goal>add-source</goal>

</goals>

<configuration>

<sources>

<source>${project.basedir}/src/main/java</source>

</sources>

</configuration>

</execution>

</executions>

</plugin>

<plugin>

<groupId>org.codehaus.mojo</groupId>

<artifactId>exec-maven-plugin</artifactId>

<version>3.1.0</version>

<executions>

<execution>

<id>generate-js-proto</id>

<phase>generate-resources</phase>

<goals>

<goal>exec</goal>

</goals>

<configuration>

<workingDirectory>${project.basedir}/src/client</workingDirectory>

<executable>npm</executable>

<arguments>

<argument>run</argument>

<argument>gen</argument>

</arguments>

</configuration>

</execution>

</executions>

</plugin>

</plugins>

</build>

Також для клієнту було створено окрему директорію з npm-модулями. Файл package.json, в якому додатково визначено команду "proto:gen":

{

"name": "frontend",

"version": "1.0.0",

"main": "index.js",

"scripts": {

"gen": "npx buf generate --output=protobuf"

},

"keywords": [],

"author": "",

"license": "ISC",

"description": "",

"dependencies": {

"@bufbuild/buf": "^1.54.0",

"@bufbuild/protobuf": "^2.5.0"

}

}

На фронтенді надалі просто імпортуємо потрібний метод і використовуємо його:

<script type="module">

import { decodePriceUpdate } from "./js/[protobuf-helper.js](http://protobuf-helper.js)";

function processMessage(data) {

if (data instanceof ArrayBuffer) {

const uint8Array = new Uint8Array(data);

try {

const priceUpdate = decodePriceUpdate(uint8Array);

updatePrice(priceUpdate);

} catch (e) {

console.error("Failed to decode binary data:", e);

}

} else {

console.warn("Received non-binary data:", data);

}

}

Тепер подивимось як це виглядає:

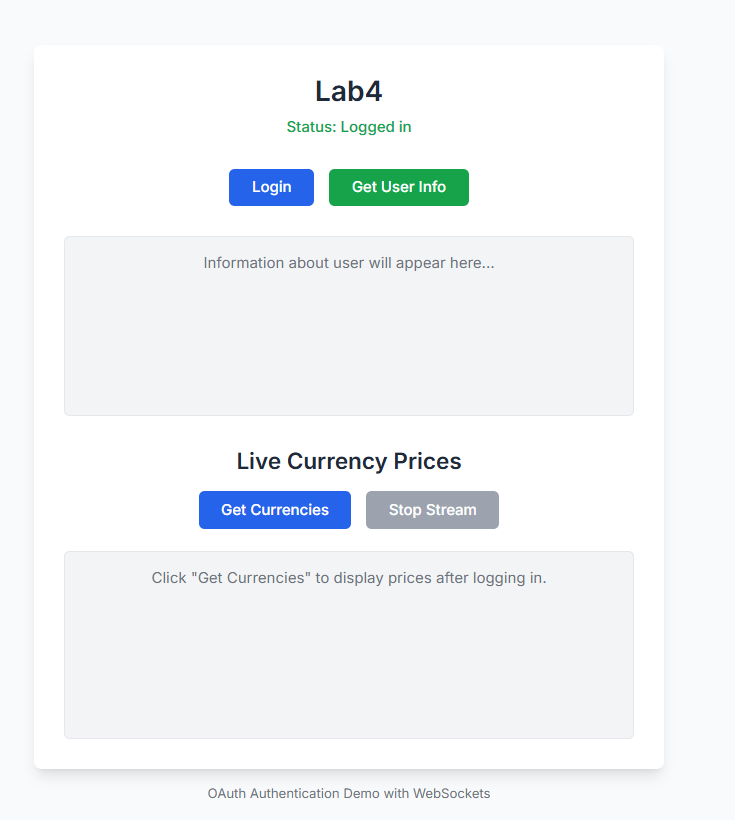
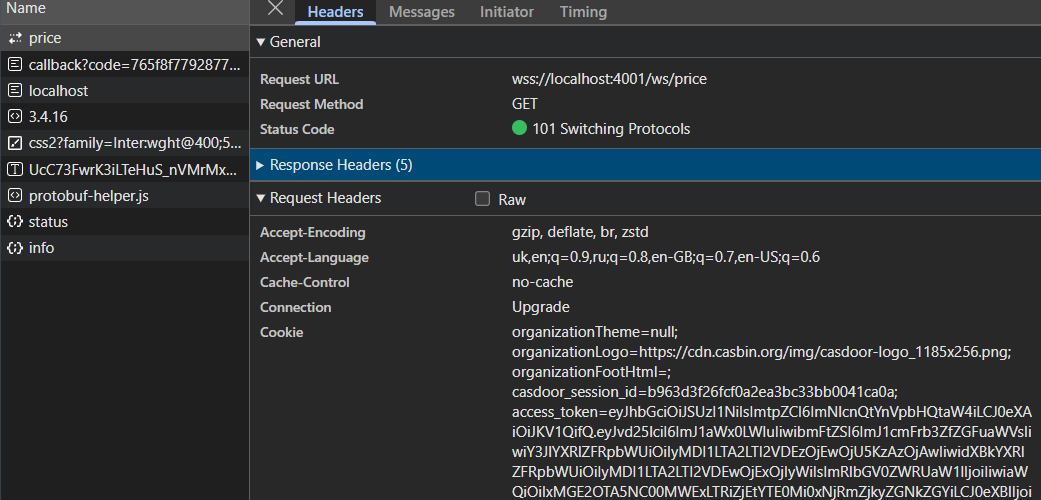


Рис. 3. Інтерфейс

devTools:



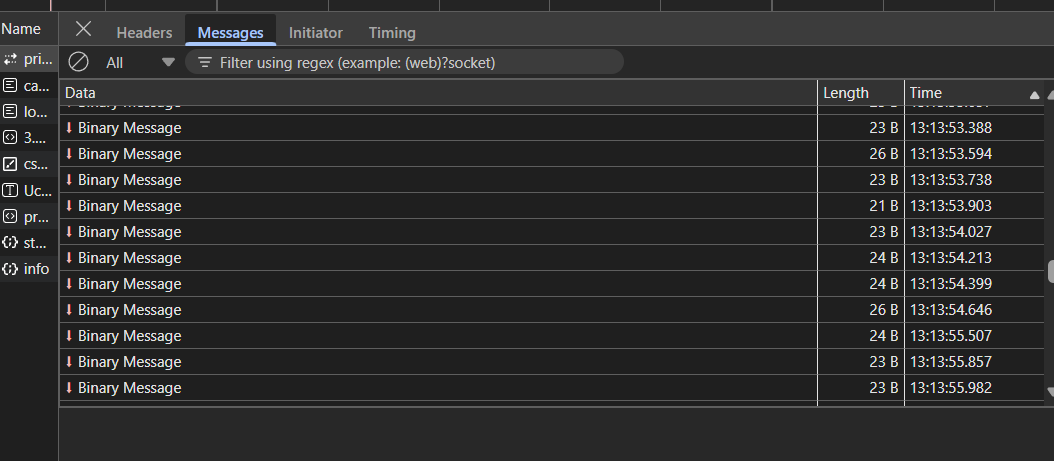


Рис. 4. Запити при натисканні кнопки Get Currencies

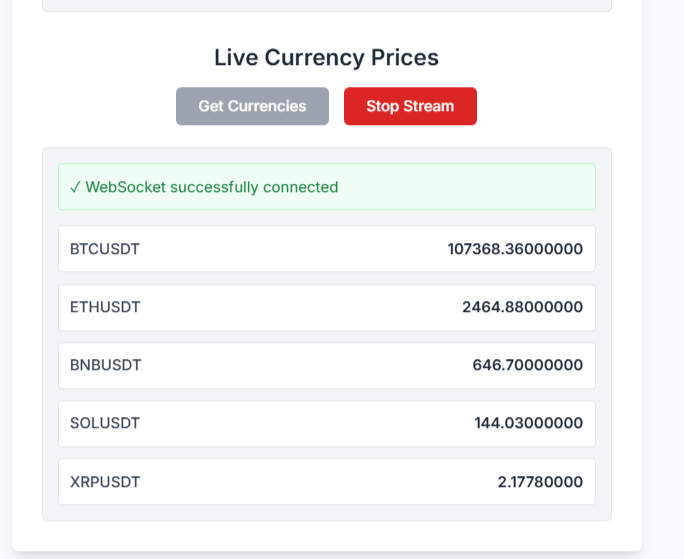


Рис. 5. Отримання валюти у псевдо постійному часі

Оглянемо на трафік через WireShark:

Дані через Websocket:

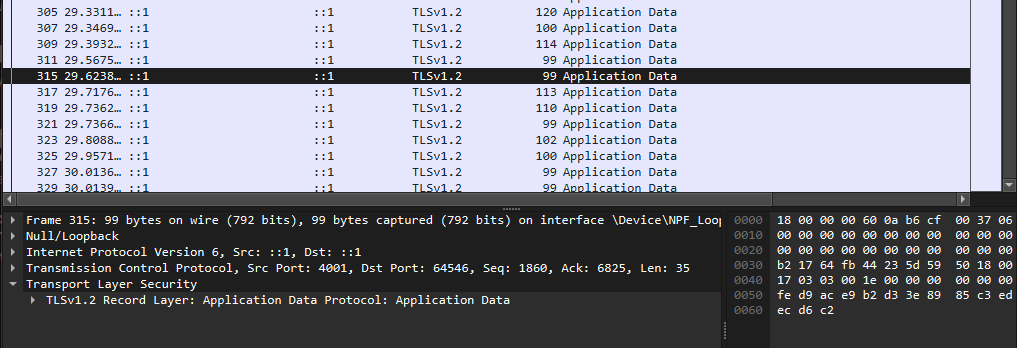


Рис. 6. Зашифровані дані через Websocket у Protobuf форматі

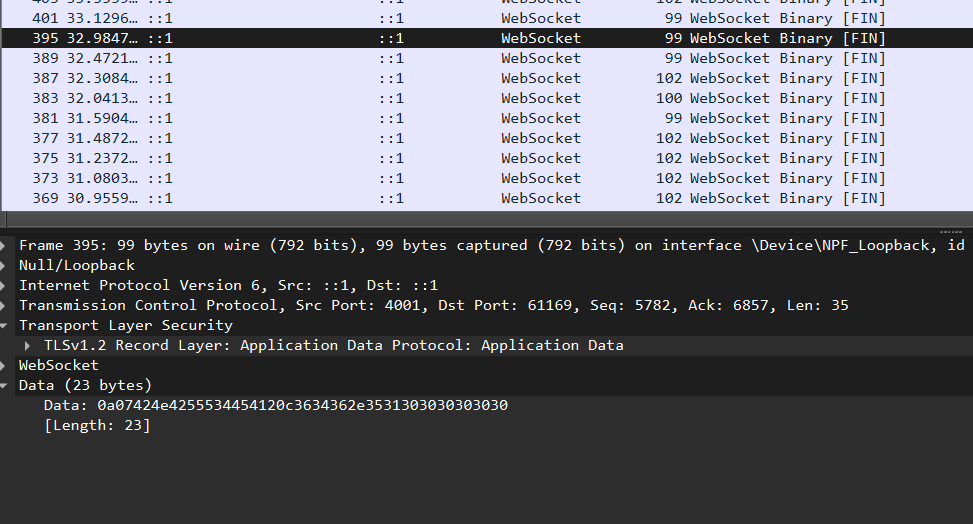


Рис. 7. Розшифровані дані через Websocket

**Висновки:**

У результаті лабораторної роботи я покращив навички отримання даних через TCP-з’єднання, їхню передачу клієнту у форматі Protobuf та подальше декодування на клієнтській стороні. Також я освоїв як розшифровувати та аналізувати WSS-запити за допомогою Wireshark.