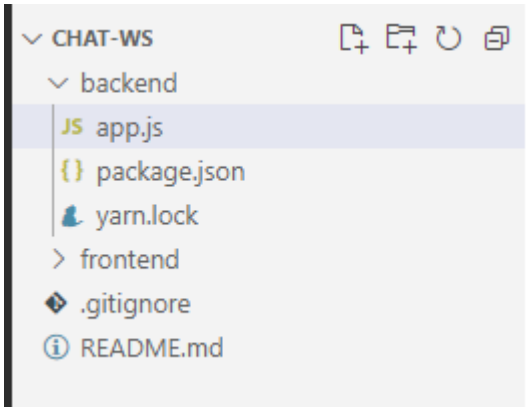
ДОДАТОК А – КОД СЕРВЕРНОЇ ЧАСТИНИ ДОДАТКУ НА МОВІ JAVASCRIPT З ЗАСТОСУВАННЯМ NODEJS



app.js

backend > app.js > ...

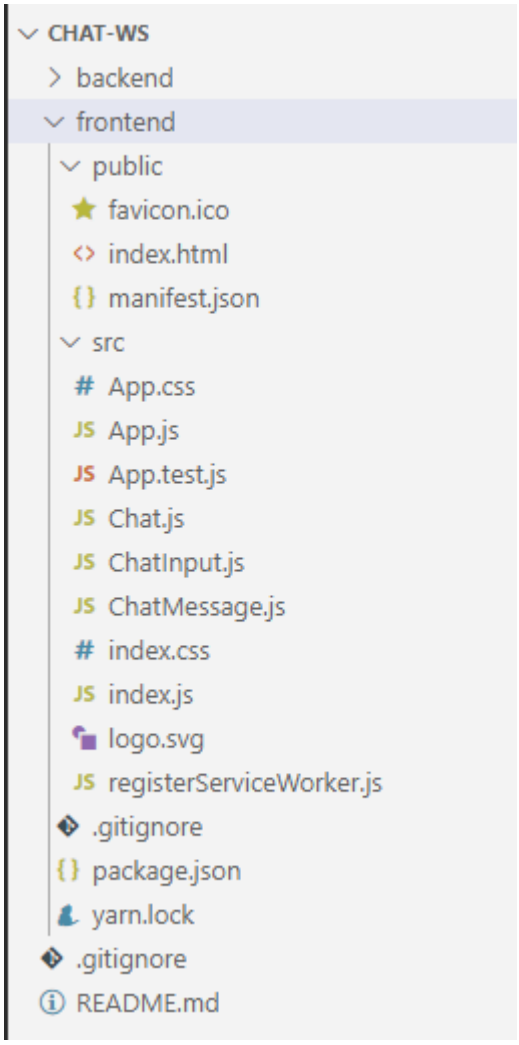
```
1   const WebSocket = require('ws');
2
3   const wss = new WebSocket.Server({ port: 3030 });
4
5   wss.on('connection', function connection(ws) {
6     ws.on('message', function incoming(data) {
7       wss.clients.forEach(function each(client) {
8         if (client !== ws && client.readyState === WebSocket.OPEN) {
9           client.send(data);
10        }
11      });
12    });
13  });
```

package.json

backend > package.json > ...

```
1   {
        Debug
2     "scripts": {
3       "start": "node app.js"
4     },
5     "dependencies": {
6       "ws": "^6.0.0"
7     }
8   }
9
```

ДОДАТОК Б – КОД КЛІЄНТСЬКОЇ ЧАСТИНИ НАПИСАНИЙ НА МОВІ JAVASCRIPT З ЗАСТОСУВАННЯМ REACTJS



package.json

frontend > package.json > dependencies > react-dom

```
{
  "name": "frontend",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "prop-types": "^17.1.0",
    "react": "^17.1.0",
    "react-dom": "^17.1.0"
  },
  Debug
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test --env=jsdom",
    "eject": "react-scripts eject"
  }
}
```

index.js

frontend > src > index.js

```
1   import React from 'react';
2   import ReactDOM from 'react-dom';
3   import './index.css';
4   import App from './App';
5   import registerServiceWorker from './registerServiceWorker';
6
7   ReactDOM.render(<App />, document.getElementById('root'));
8   registerServiceWorker();
9
```

App.js

frontend > src > App.js > App

```
1   import React, { Component } from 'react'
2   import logo from './logo.svg'
3   import './App.css'
4   import Chat from './Chat'
5
6   class App extends Component {
7     render() {
8       return (
9         <div className="App">
10          <header className="App-header">
11            <img src={logo} className="App-logo" alt="logo" />
12            <h1 className="App-title">Welcome to React</h1>
13          </header>
14          <Chat />
15        </div>
16      )
17    }
18  }
19
20  export default App
21
```

JS Chat.js × Settings

frontend > src > JS Chat.js > ...

```
1   import React, { Component } from 'react'
2   import ChatInput from './ChatInput'
3   import ChatMessage from './ChatMessage'
4
5   const URL = 'ws://localhost:3030'
6
7   class Chat extends Component {
8     state = {
9       name: 'Bob',
10      messages: [],
11    }
12
13    ws = new WebSocket(URL)
14
15    componentDidMount() {
16      this.ws.onopen = () => {
17        // on connecting, do nothing but log it to the console
18        console.log('connected')
19      }
20
21      this.ws.onmessage = evt => {
22        // on receiving a message, add it to the list of messages
23        const message = JSON.parse(evt.data)
24        this.addMessage(message)
25      }
26
27      this.ws.onclose = () => {
28        console.log('disconnected')
29        // automatically try to reconnect on connection loss
30        this.setState({
31          ws: new WebSocket(URL),
32        })
33      }
34    }
35
36    addMessage = message =>
37      this.setState(state => ({ messages: [message, ...state.messages] }))
38
39    submitMessage = messageString => {
40      // on submitting the ChatInput form, send the message, add it to the list and reset the input
41      const message = { name: this.state.name, message: messageString }
42      this.ws.send(JSON.stringify(message))
43      this.addMessage(message)
44    }
45
46    render() {
47      return (
48        <div>
49          <label htmlFor="name">
50            Name: 
51            <input
52              type="text"
53              id={'name'}
54              placeholder={'Enter your name...'}
55              value={this.state.name}
56              onChange={e => this.setState({ name: e.target.value })}
57            />
58          </label>
59          <ChatInput
60            ws={this.ws}
61            onSubmitMessage={messageString => this.submitMessage(messageString)}
62          />
63          {this.state.messages.map((message, index) =>
64            <ChatMessage
65              key={index}
66              message={message.message}
67              name={message.name}
68            />,
69          )}
70        </div>
71      )
72    }
73  }
74
75  export default Chat
76
```

JS ChatInput.js ×

frontend > src > JS ChatInput.js > ...

```
import React, { Component } from 'react'
import PropTypes from 'prop-types'

class ChatInput extends Component {
  static propTypes = {
    onSubmitMessage: PropTypes.func.isRequired,
  }
  state = {
    message: '',
  }

  render() {
    return (
      <form
        action="."
        onSubmit={e => {
          e.preventDefault()
          this.props.onSubmitMessage(this.state.message)
          this.setState({ message: '' })
        }}
      >
        <input
          type="text"
          placeholder={'Enter message...'}
          value={this.state.message}
          onChange={e => this.setState({ message: e.target.value })}
        />
        <input type="submit" value={'Send'} />
      </form>
    )
  }
}

export default ChatInput

```

JS ChatMessage.js ×

frontend > src > JS ChatMessage.js > ...

```
import React from 'react'

export default ({ name, message }) =>
  <p>
    <strong>{name}</strong> <em>{message}</em>
  </p>

```

index.css

frontend > src > # index.css > body

```css
body {
  margin: 0;
  padding: 0;
  font-family: sans-serif;
}

```

App.css

frontend > src > # App.css > .App

```css
.App {
  text-align: center;
}

.App-logo {
  animation: App-logo-spin infinite 20s linear;
  height: 80px;
}

.App-header {
  background-color: #222;
  height: 150px;
  padding: 20px;
  color: white;
}

.App-title {
  font-size: 1.5em;
}

.App-intro {
  font-size: large;
}

@keyframes App-logo-spin {
  from { transform: rotate(0deg); }
  to { transform: rotate(360deg); }
}

```

JS App.test.js

frontend > src > JS App.test.js > ...

```javascript
import React from 'react';
import ReactDOM from 'react-dom';
import App from './App';

it('renders without crashing', () => {
  const div = document.createElement('div');
  ReactDOM.render(<App />, div);
  ReactDOM.unmountComponentAtNode(div);
});

```

ДОДАТОК В – ТЕЗИ ДОПОВІДІ ДО 25-ГО МІЖНАРОДНОГО МОЛОДІЖНОГО ФОРУМУ «РАДІОЕЛЕТРОНІКА І МОЛОДЬ У XXI СТОЛІТТІ»

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЛЕКТРОНІКИ

МАТЕРІАЛИ 25-го МІЖНАРОДНОГО МОЛОДІЖНОГО ФОРУМУ

«РАДІОЕЛЕТРОНІКА І МОЛОДЬ У XXI СТОЛІТТІ»

20 – 22 квітня 2021 р.

Том 4

КОНФЕРЕНЦІЯ

«ПЕРСПЕКТИВИ РОЗВИТКУ ІНФОКОМУНІКАЦІЙ ТА ІНФОРМАЦІЙНО-ВИМІРЮВАЛЬНИХ ТЕХНОЛОГІЙ»

Харків 2021

УДК 004:[621.317+621.391](06)

25-й Міжнародний молодіжний форум «Радіоелектроніка та молодь у XXI столітті». Зб. Матеріалів форуму. Т.4. – Харків: ХНУРЕ. 2021. – 176 с.

В збірник включені матеріали 25-го Міжнародного молодіжного форуму «Радіоелектроніка та молодь у XXI столітті».

Видання підготовлено факультетом інфокомунікацій
Харківського національного університету радіоелектроніки

61166 Україна, Харків, прос. Науки, 14
тел./факс.: (057) 7021397

E-mail: mref21@nure.ua

Харківський національний університет
радіоелектроніки (ХНУРЕ), 2021

ДОСЛІДЖЕННЯ ОСОБЛИВОСТЕЙ ОПТИМІЗАЦІЇ ТА НАДІЙНОСТІ ВЗАЄМОДІЇ СЕРВЕРНОЇ І КЛІЄНТСЬКОЇ ЧАСТИН МЕРЕЖНИХ WEB-ДОДАТКІВ

Лялічев В. Д.
Науковий керівник – к.т.н. доцент Бондар Д. В.
Харківський національний університет радіоелектроніки
(61166, Харків, пр. Науки,14, каф. Інформаційно-мережна інженерія,
тел. (057) 702-13-06)
e-mail: vladyslav.lialichev@nure.ua , +380669148523

The spread of information systems is constantly increasing, but they are becoming more complex. Over time, computer resources began to evolve rapidly into ideas and mechanisms that began to grow and evolve. It was to ensure the maximum benefit and performance of information systems that the client-server architecture was invented, which helped solve many problems at the time.

As the number of these systems continues to grow, so do their requirements. The complexity of designing and developing such systems requires a lot of time and effort, and the methods and tools used to implement such projects differ from the development of standard systems.

Поширення інформаційних систем постійно збільшується, але вони стають дедалі складнішими. З часом комп'ютерні ресурси почали швидко еволюціонувати в ідеї та механізми, які почали рости та розвиватися.[1]

Саме для забезпечення максимальної вигоди та продуктивності інформаційних систем була винайдена архітектура клієнт-сервер, яка допомогла вирішити багато проблем на той час.

Термін "клієнт-сервер" відноситься до такого архітектурного програмного комплексу, в якому його функціональні частини взаємодіють з найпростішою схемою, клієнт дає попит, сервер дає відповідь.

Якщо ми розглянемо кожну частину взаємодії з цього комплексу, один з них, конкретний клієнт, робить активну роботу, тобто вони утворюють певні вимоги, а інший, сервер, відповідає.

Як розвиток інформаційних систем, ці завдання можуть відрізнятися, наприклад, розробка блоків одночасно для виконання функцій сервера та функцій клієнтів у порівнянні з іншими блоками.

Щоб стати сучасною архітектурою, взаємодія клієнт-сервер пройшло довгий шлях, починаючи з централізованої системи архітектурних додатків, які були популярні в 70-х роках минулого століття. Згодом такі системи перейшли на новий рівень, рівень персональних комп'ютерів і локальних задач на цих машинах.

З розвитком персональних машин на новий етап перейшли локальні мережі з розвиненою файлово-серверної архітектурою. Так з'явилися перші однорангові мережі, і з розвитком цих мереж почалося перше поділ комп'ютерів на клієнтів і сервери.[3] А з розвитком технологій і їх

вдосконаленням була створена архітектура клієнт-сервер, яка сьогодні займає високі позиції.

Але в наш час такі системи зростають і стають настільки складними, що набувають глобального характеру, і діяльність великої кількості людей почала залежати від їх правильної та надійної роботи. Такі системи часто мають дуже складну архітектуру, що складається з великого набору компонентів, кожен з яких працює на окремому пристрої. [4]

Клієнт-сервер є класична архітектура, що має розподіляти три основні частини програми на двох фізичних модулях. Зазвичай дані зберігання знаходяться на інформаційному сервері, таких як певний сервер баз даних, користувальницький інтерфейс - на стороні клієнта та обробка даних розподіляється між частинами клієнта та сервером.

Оскільки кількість цих систем продовжує зростати, зростають і їх вимоги.[2] Складність проектування та розробки таких систем вимагає багато часу та зусиль, а методи та засоби, що використовуються для реалізації таких проектів, відрізняються від розробки стандартних систем.

Література:

10. Многоуровневые системы клиент-сервер Валерий Коржов. [Текст]. Режим доступу : https://www.osp.ru/nets/1997/06/142618. /. –Дата доступу: березень 2021.
11. Компоненты сетевого приложения. [Електронний ресурс]. Режим доступу : http://www.4stud.info/networking. –Дата доступу: березень 2021
12. Архитектура клиент-сервер [Електронний ресурс]. Режим доступу : https://sergeygavaga.gitbooks.io/–Дата доступу: травень 2020.
13. Клиент-сервер. [Електронний ресурс]. Режим доступу : https://developer.mozilla.org/ Дата доступу: березень 2021.
14. What is a Thin Client [Електронний ресурс]. Режим доступу : https://www.clearcube.com/posts/what-is-a-thin. –Дата доступу: березень 2021.

АЛФАВІТНИЙ ПЕРЕЛІК

ДОДАТОК Г – СЛАЙДИ ПРЕЗЕНТАЦІЇ

Харківський національний університет радіоелектроніки
Міністерство освіти і науки України
Кафедра Інформаційно-мережної інженерії

Кваліфікаційна робота магістра на тему:

Дослідження особливостей оптимізації та надійності взаємодії серверної і клієнтської частин мережних WEB-додатків

Виконав:

Студент 2-го курсу	Лялічев Владислав
Група	ІМІм-19-2
Керівник	Бондар Д.В.

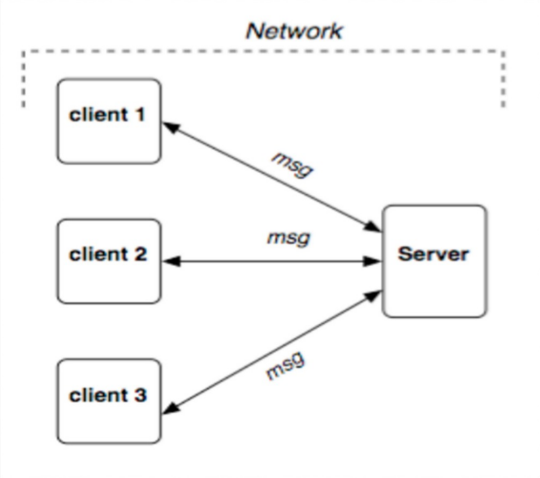
ВСТУП

З розвитком Інтернет-технологій Інтернет стає все більш важливим у нашому житті, так що він навіть стає важливим елементом. У той же час застосування Мережі ніколи не обмежувалось лише комп'ютерами.

Архітектура Інтернету - це модель клієнт-сервер. Як результат, спілкування між сервером і клієнтом - це перше, що нас повинно турбувати. Тільки на основі успішного та безперебійного спілкування веб-технологія може рухатись вперед, а веб-програми застосовуватимуться до всіх типів пристроїв, щоб вони могли допомогти людям.

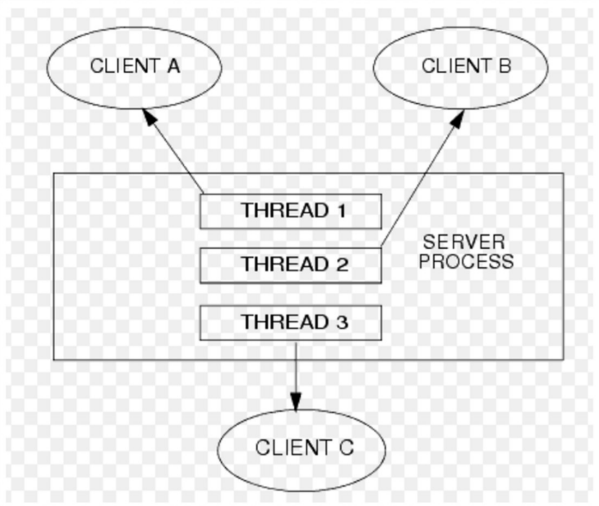
2

МОДЕЛЬ КЛІЄНТ-СЕРВЕР



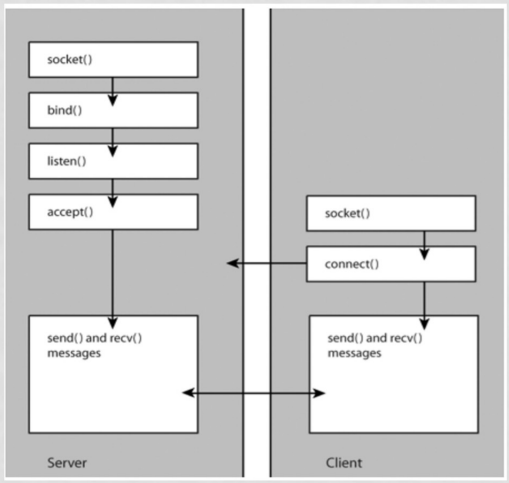
3

БАГАТОПОТОКОВА ОБРОБКА РІЗНИХ КЛІЄНТІВ



4

TCP-З'ЄДНАННЯ



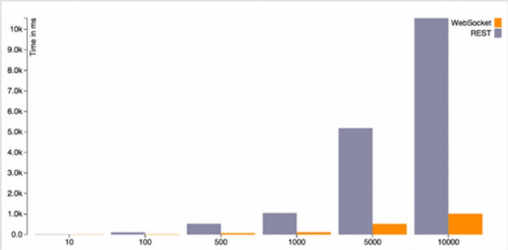
5

ТЕХНОЛОГІЇ ЗВ'ЯЗКУ МІЖ КЛІЄНТОМ ТА СЕРВЕРОМ

- HTTP;
- REST;
- HTTP Polling;
- HTTP Streaming;
- Server Sent Events (SSE);
- HTTP/2 Server Push;
- WebSockets.

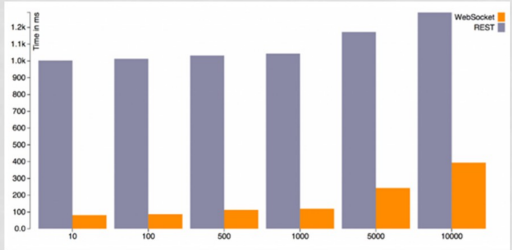
6

REST ПРОТИ WEBSOCKETS



Messages	REST (in ms)	WebSocket (in ms)	x times
10	17	13	1.31
100	112	20	5.60
500	529	68	7.78
1000	1050	115	9.13
5000	5183	522	9.93
10000	10547	1019	10.35

Результати збільшення кількості повідомлень



Payload (in bytes)	REST (in ms)	WebSocket (in ms)	x times
10	1003	81	12.38
100	1013	87	11.64
500	1032	113	9.13
1000	1044	119	8.77
5000	1173	243	4.83
10000	1289	394	3.27

Результати збільшення корисного навантаження

7

ПРИНЦИПИ ОПТИМІЗАЦІЇ

- Виконання пошуку DNS;
- Встановлення зв'язку;
- Переговори щодо рукостискання SSL;
- Надсилання запиту;
- Час очікування;
- Отримання відповіді.

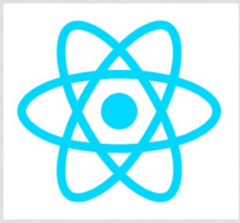
8

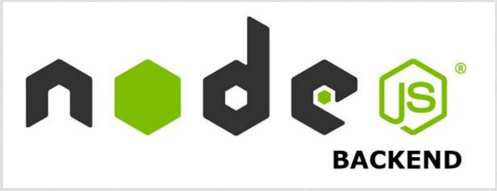
НАДІЙНІСТЬ В СУЧАСНИХ WEB-ДОДАТКІВ

- Підтвердження операції (ACK);
- Час очікування;
- Порядкові номери (SEQ);
- Протокол конвеєризації;
- Протокол Go-Back-N (GBN);
- Вибіркове повторення (SR).

9

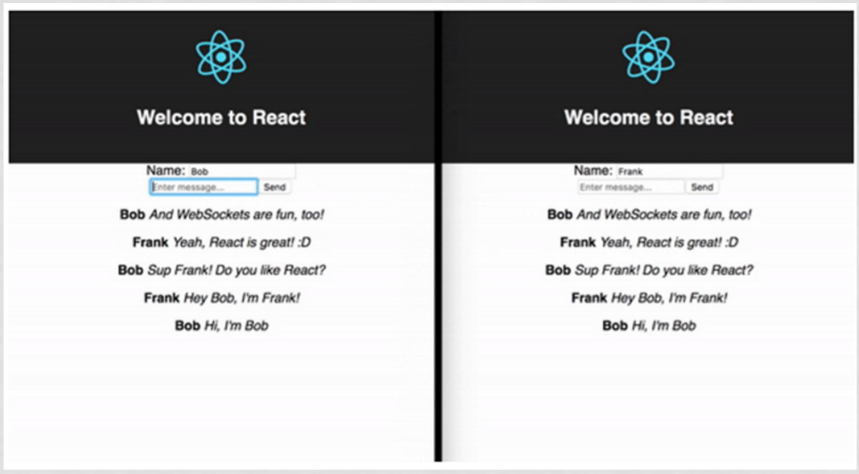
ВИБІР ТЕХНОЛОГІЙ ДЛЯ СУЧАСНОГО WEB-ДОДАТКУ





10

ЗОВНІШНІЙ ВИГЛЯД WEB-ДОДАТКУ



11

ВИСНОВКИ

У зв’язку з активним поширенням інтернет–технологій та розповсюдженням веб–сторінок з’являється необхідність у забезпеченні оптимізації та надійності зв’язку. Саме таку оптимізацію і надійність надають сучасні технології обміну інформацією, більшість із яких існує у відкритому доступі.

Завдяки багатої безлічі існуючих технологій, на поточний момент, кожен може обрати, що використовувати для досягнення особистої мети. При вдалому виборі кожен додаток або програма стає швидше за рахунок швидкого обміну збереженої інформації та додаванню нової.

Як показало дослідження сучасного ринку – саме realtime технології знайшли поширене використання за рахунок швидкості передачі даних, та оптимізації, що гарантує безпеку збереженості даних. Також у кваліфікаційній роботі було розглянуто багато різних технологій, виявлено переваги та недоліки.

12

Дякую за увагу!

Позначення	Найменування	Дод. відомості
	Текстові документи	
1.	Пояснювальна записка	69 с.
	Графічні документи	
2.	Слайди презентації	13

Змін.	Арк.	Номер докум.	Підпис	Дата	Дослідження особливостей оптимізації та надійності взаємодії серверної і клієнтської частин мережних WEB-додатків Відомість кваліфікаційної роботи	Літ.	Аркуш	Аркушів
Розроб.		Лялічев				У	1	1
Перевір.		Бондар				ХНУРЕ кафедра ІМІ		
Н. контр.		Бондар						
Затв.		Безрук						