ДОДАТОК А

Лістинг програми

```
const axios = require('axios');
const yahooFinance = require('yahoo-finance2').default;
const ss = require('simple-statistics');
const { API_KEY } = require('../config/apiConfig');

// Аналіз зміни ціни за день
async function analyzeDailyStockChange(symbol, threshold = 5) {
  const url = `https://www.alphavantage.co/query?function=TIME_SERIES_DAILY&symbol=${symbol}&apikey=${API_KEY}`;
  try {
    const response = await axios.get(url);
    const data = response.data["Time Series (Daily)"];

    if (!data || response.data["Error Message"]) {
      return `Error: Invalid stock symbol ${symbol}.`;
    }

    const dates = Object.keys(data);
    if (dates.length < 2) {
      return `Not enough data for ${symbol}.`;
    }

    const latestDate = dates[0];
    const previousDate = dates[1];
```

```
        const latestClose = parseFloat(data[latestDate]["4. close"]);
        const previousClose = parseFloat(data[previousDate]["4. close"]);

        const changePercentage = ((latestClose - previousClose) / previousClose) * 100;

        if (Math.abs(changePercentage) > threshold) {
          return `Alert! The stock price for ${symbol} changed by ${changePercentage.toFixed(2)}% on ${latestDate}.`;
        } else {
          return `No significant change for ${symbol}.`;
        }
      } catch (error) {
        console.error('Error fetching stock data:', error);
        return `Error fetching stock data for ${symbol}. Please try again later.`;
      }
    }

    // Аналіз середньої ціни протягом тижня
    async function analyzeWeeklyAverage(symbol) {
      const url = `https://www.alphavantage.co/query?function=TIME_SERIES_DAILY&symbol=${symbol}&apikey=${API_KEY}`;
      try {
        const response = await axios.get(url);
        const data = response.data["Time Series (Daily)"];

        if (!data || response.data["Error Message"]) {
          return `Error: Invalid stock symbol ${symbol}.`;
```

```
        }

        const dates = Object.keys(data);
        let sum = 0;
        const days = Math.min(7, dates.length); // Якщо даних менше 7 днів, беремо стільки, скільки є

        for (let i = 0; i < days; i++) {
          const closePrice = parseFloat(data[dates[i]]["4. close"]);
          sum += closePrice;
        }

        const average = sum / days;
        return `The 7-day average closing price for ${symbol} is ${average.toFixed(2)} USD.`;
      } catch (error) {
        console.error('Error fetching stock data:', error);
        return `Error fetching stock data for ${symbol}. Please try again later.`;
      }
    }

    // Аналіз місячних максимумів та мінімумів
    async function analyzeMonthlyHighLow(symbol) {
      const url = `https://www.alphavantage.co/query?function=TIME_SERIES_DAILY&symbol=${symbol}&apikey=${API_KEY}`;
      try {
        const response = await axios.get(url);
        const data = response.data["Time Series (Daily)"];
```

```
        if (!data || response.data["Error Message"]) {
          return `Error: Invalid stock symbol ${symbol}.`;
        }

        const dates = Object.keys(data);
        let maxPrice = -Infinity;
        let minPrice = Infinity;
        const days = Math.min(30, dates.length); // Беремо останні 30 днів

        for (let i = 0; i < days; i++) {
          const closePrice = parseFloat(data[dates[i]]["4. close"]);
          if (closePrice > maxPrice) {
            maxPrice = closePrice;
          }
          if (closePrice < minPrice) {
            minPrice = closePrice;
          }
        }

        return `The highest closing price for ${symbol} in the last 30 days was ${maxPrice.toFixed(2)} USD, and the lowest was ${minPrice.toFixed(2)} USD.`;
      } catch (error) {
        console.error('Error fetching stock data:', error);
        return `Error fetching stock data for ${symbol}. Please try again later.`;
      }
    }

    // Перевірка зростаючого тренду
    async function checkRisingTrend(symbol, days = 3) {
```

```
    const url = `https://www.alphavantage.co/query?function=TIME_SERIES_DAILY&symbol=${symbol}&apikey=${API_KEY}`;
    try {
      const response = await axios.get(url);
      const data = response.data["Time Series (Daily)"];

      if (!data || response.data["Error Message"]) {
        return `Error: Invalid stock symbol ${symbol}.`;
      }

      const dates = Object.keys(data);
      if (dates.length < days) {
        return `Not enough data for ${symbol}.`;
      }

      let isRising = true;
      for (let i = 0; i < days; i++) {
        const currentClose = parseFloat(data[dates[i]]["4. close"]);
        const previousClose = parseFloat(data[dates[i + 1]]["4. close"]);

        if (currentClose <= previousClose) {
          isRising = false;
          break;
        }
      }

      if (isRising) {
```

```
        return `The stock price for ${symbol} has been rising for the last ${days} days.`;
      } else {
        return `The stock price for ${symbol} is not showing a rising trend.`;
      }
    } catch (error) {
      console.error('Error fetching stock data:', error);
      return `Error fetching stock data for ${symbol}. Please try again later.`;
    }
  }

  // Аналіз зміни ціни протягом місяця
  async function analyzeMonthlyStockChange(symbol) {
    const url = `https://www.alphavantage.co/query?function=TIME_SERIES_DAILY&symbol=${symbol}&apikey=${API_KEY}`;
    try {
      const response = await axios.get(url);
      const data = response.data["Time Series (Daily)"];

      if (!data || response.data["Error Message"]) {
        return `Error: Invalid stock symbol ${symbol}.`;
      }

      const dates = Object.keys(data);
      if (dates.length < 30) {
        return `Not enough data for ${symbol}.`;
      }
```

```
        const latestDate = dates[0];
        const monthAgoDate = dates[29];

        const latestClose = parseFloat(data[latestDate]["4. close"]);
        const monthAgoClose = parseFloat(data[monthAgoDate]["4. close"]);

        const changePercentage = ((latestClose - monthAgoClose) / monthAgoClose) * 100;

        return `The stock price for ${symbol} changed by ${changePercentage.toFixed(2)}% over the last month.`;
      } catch (error) {
        console.error('Error fetching stock data:', error);
        return `Error fetching stock data for ${symbol}. Please try again later.`;
      }
    }

    // Перевірка волатильності за останні 10 днів
    async function analyzeVolatility(symbol) {
      const url = `https://www.alphavantage.co/query?function=TIME_SERIES_DAILY&symbol=${symbol}&apikey=${API_KEY}`;
      try {
        const response = await axios.get(url);
        const data = response.data["Time Series (Daily)"];

        if (!data || response.data["Error Message"]) {
          return `Error: Invalid stock symbol ${symbol}.`;
        }
```

```
        const dates = Object.keys(data);
        let totalVolatility = 0;
        const days = Math.min(10, dates.length);

        for (let i = 0; i < days - 1; i++) {
          const closeToday = parseFloat(data[dates[i]]["4. close"]);
          const closeYesterday = parseFloat(data[dates[i + 1]]["4. close"]);
          totalVolatility += Math.abs(closeToday - closeYesterday);
        }

        const averageVolatility = totalVolatility / (days - 1);
        return `The average daily volatility for ${symbol} over the last 10 days is ${averageVolatility.toFixed(2)} USD.`;
      } catch (error) {
        console.error('Error fetching stock data:', error);
        return `Error fetching stock data for ${symbol}. Please try again later.`;
      }
    }

    // Аналіз найбільших коливань за 2 тижні
    async function analyzeBiggestSwings(symbol) {
      const url = `https://www.alphavantage.co/query?function=TIME_SERIES_DAILY&symbol=${symbol}&apikey=${API_KEY}`;
      try {
        const response = await axios.get(url);
        const data = response.data["Time Series (Daily)"];

        if (!data || response.data["Error Message"]) {
```

```
        return `Error: Invalid stock symbol ${symbol}.`;
      }

      const dates = Object.keys(data);
      let maxSwing = 0;
      const days = Math.min(14, dates.length);

      for (let i = 0; i < days - 1; i++) {
        const closeToday = parseFloat(data[dates[i]]["4. close"]);
        const closeYesterday = parseFloat(data[dates[i + 1]]["4. close"]);
        const swing = Math.abs(closeToday - closeYesterday);
        if (swing > maxSwing) {
          maxSwing = swing;
        }
      }

      return `The biggest daily price swing for ${symbol} over the last 14 days was ${maxSwing.toFixed(2)} USD.`;
    } catch (error) {
      console.error('Error fetching stock data:', error);
      return `Error fetching stock data for ${symbol}. Please try again later.`;
    }
  }

  // Аналіз тижневого зростання
  async function analyzeWeeklyGrowth(symbol) {
    const url = `https://www.alphavantage.co/query?function=TIME_SERIES_DAILY&symbol=${symbol}&apikey=${API_KEY}`;
```

```
        try {
          const response = await axios.get(url);
          const data = response.data["Time Series (Daily)"];

          if (!data || response.data["Error Message"]) {
            return `Error: Invalid stock symbol ${symbol}.`;
          }

          const dates = Object.keys(data);
          if (dates.length < 7) {
            return `Not enough data for ${symbol}.`;
          }

          const latestDate = dates[0];
          const weekAgoDate = dates[6];

          const latestClose = parseFloat(data[latestDate]["4. close"]);
          const weekAgoClose = parseFloat(data[weekAgoDate]["4. close"]);

          const growthPercentage = ((latestClose - weekAgoClose) / weekAgoClose) * 100;

          return `The stock price for ${symbol} grew by ${growthPercentage.toFixed(2)}% over the last week.`;
        } catch (error) {
          console.error('Error fetching stock data:', error);
          return `Error fetching stock data for ${symbol}. Please try again later.`;
        }
      }
```

```
// Перевірка збільшення обсягів торгів протягом останніх 5 днів
async function analyzeVolumeSpike(symbol) {
 const url = `https://www.alphavantage.co/query?function=TIME_SERIES_DAILY&symbol=${symbol}&apikey=${API_KEY}`;
 try {
  const response = await axios.get(url);
  const data = response.data["Time Series (Daily)"];

  if (!data || response.data["Error Message"]) {
   return `Error: Invalid stock symbol ${symbol}.`;
  }

  const dates = Object.keys(data);
  let maxVolume = 0;
  const days = Math.min(5, dates.length);

  for (let i = 0; i < days; i++) {
   const volume = parseFloat(data[dates[i]]["5. volume"]);
   if (volume > maxVolume) {
    maxVolume = volume;
   }
  }

  return `The highest trading volume for ${symbol} in the last 5 days was ${maxVolume.toFixed(0)}.`;
 } catch (error) {
  console.error('Error fetching stock data:', error);
  return `Error fetching stock data for ${symbol}. Please try again later.`;
```

```
  }
 }

 // Аналіз денної волатильності протягом місяця
 async function analyzeDailyVolatility(symbol) {
  const url = `https://www.alphavantage.co/query?function=TIME_SERIES_DAILY&symbol=${symbol}&apikey=${API_KEY}`;
  try {
   const response = await axios.get(url);
   const data = response.data["Time Series (Daily)"];

   if (!data || response.data["Error Message"]) {
    return `Error: Invalid stock symbol ${symbol}.`;
   }

   const dates = Object.keys(data);
   let totalVolatility = 0;
   const days = Math.min(30, dates.length);

   for (let i = 0; i < days - 1; i++) {
    const closeToday = parseFloat(data[dates[i]]["4. close"]);
    const closeYesterday = parseFloat(data[dates[i + 1]]["4. close"]);
    totalVolatility += Math.abs(closeToday - closeYesterday);
   }

   const averageVolatility = totalVolatility / (days - 1);
   return `The average daily volatility for ${symbol} over the last 30 days is ${averageVolatility.toFixed(2)} USD.`;
```

```
  } catch (error) {
    console.error('Error fetching stock data:', error);
    return `Error fetching stock data for ${symbol}. Please try again later.`;
  }
}

// Функція отримання історичних даних
async function getHistoricalData(symbol) {
  try {
    const queryOptions = { period1: '2020-01-01', interval: '1d' };
    const data = await yahooFinance.historical(symbol, queryOptions);
    return data;
  } catch (error) {
    console.error('Помилка отримання даних:', error);
    return null;
  }
}

// Прогнозування на тиждень уперед за допомогою лінійної регресії
function predictNextWeekPrices(data, weeksAhead = 52) {
  const prices = data.map((day, index) => [index, day.close]);
  const regression = ss.linearRegression(prices);
  const regressionLine = ss.linearRegressionLine(regression);

  const predictions = [];
  const lastDate = new Date(data[data.length - 1].date); // Остання відома дата
  let lastPrice = data[data.length - 1].close; // Остання відома ціна
```

```
        // Розраховуємо волатильність та обмежуємо її діапазон
        const historicalStdDev = Math.sqrt(
          ss.variance(data.map(day => day.close)) || 1
        ) * 0.05; // Мінімальний вплив волатильності

        for (let i = 1; i <= weeksAhead; i++) {
          // Прогнозована ціна з урахуванням мінімального тренду
          const trendAdjustment = (regressionLine(data.length + i * 7) - regressionLine(data.length)) * 0.05; // Слабый тренд
          const volatilityAdjustment = (Math.random() - 0.5) * 2 * historicalStdDev * 0.5; // Ще менше коливань

          // Прогнозована ціна з додаванням тренду та волатильності до останньої відомої ціни
          lastPrice = lastPrice + trendAdjustment + volatilityAdjustment;

          predictions.push({
            date: new Date(lastDate).setDate(lastDate.getDate() + i * 7), // Зрушуємо дату на тиждень
            close: lastPrice,
          });
        }

        return predictions;
      }

      module.exports = {
        analyzeDailyStockChange,
        analyzeWeeklyAverage,
```

```
  analyzeMonthlyHighLow,
  checkRisingTrend,
  analyzeMonthlyStockChange,
  analyzeVolatility,
  analyzeBiggestSwings,
  analyzeWeeklyGrowth,
  analyzeVolumeSpike,
  analyzeDailyVolatility,
  getHistoricalData,
  predictNextWeekPrices,
};
```

ДОДАТОК Б

Апробація результатів наукових досліджень

Ministry of Education and Science of Ukraine
Odessa National University of Technology
Vinnytsia National Technical University
P.N. Platonov Institute of Computer Engineering, Automation, Robotics and Programming

INFORMATION TECHNOLOGIES AND AUTOMATION– 2024

PROCEEDINGS
OF THE XVII INTERNATIONAL SCIENTIFIC AND PRACTICAL CONFERENCE

OCTOBER 31 - NOVEMBER 1, 2024

Odesa

ПРЕЗИДІЯ ТА ОРГКОМІТЕТ КОНФЕРЕНЦІЇ
PRESIDIUM AND ORGANIZING COMMITTEE OF THE CONFERENCE

ГОЛОВА ПРЕЗИДІЇ
CHAIRMAN OF THE PRESIDIUM

Богдан Єгоров, Президент ОНТУ, академік НААН України, д.т.н., професор

ЧЛЕНИ ПРЕЗИДІЇ
MEMBERS OF THE PRESIDIUM

Надія Дец, к.т.н., доцент, в.о.ректора Одеського національного технологічного університету

Ольга Ольшевська, к.т.н., доцент, проректор з наукової роботи і міжнародних зв'язків Одеського національного технологічного університету.

ГОЛОВА ОРГКОМІТЕТУ
CHAIRMAN OF THE ORGANIZING COMMITTEE

Сергій Котлик, к.т.н., доц. каф. ІТтаКБ, ОНТУ

ЗАСТУПНИК ГОЛОВИ ОРГКОМІТЕТУ
DEPUTY CHAIRMAN OF THE ORGANIZING COMMITTEE

Виктор Хобін – д.т.н., професор кафедри АТПтаРС ОНТУ

ЧЛЕНИ ОРГКОМІТЕТУ
MEMBERS OF THE ORGANIZING COMMITTEE

Panagiotis Tzionas, prof. (Thessaloniki, Greece)
Qiang Huang, prof. (Los Angeles C.A., USA)
Yangmin Li, prof (Macao, China)
Артеменко С.В., проф., (Одеса, Україна)
Романюк О.Н., проф. (Вінниця, Україна)
Грабко В.В., проф. (Вінниця, Україна)
Жученко А.І., проф. (Київ, Україна)
Ладанюк А.П., проф. (Київ, Україна)
Лисенко В.Ф., проф. (Київ, Україна)
Любчик Л.М., проф. (Харків, Україна)
Палов І., проф. (Русе, Болгарія)
Стовкова В.Д., доц. (Тракия, Болгарія)
Суслов В., доц. (Кошалін, Польща)
Артем'єв П., проф. (Ольштин, Польща)
Судацевські В., доц. (Кишинів, Молдова)
Аманжолова С., доц. (Алмати, Казахстан)

УДК 004.01/08

Інформаційні технології і автоматизація – 2024 / Матеріали XVII міжнародної науково-практичної конференції. Одеса, 31 жовтня - 1 листопада 2024 р. - Одеса, Видавництво ОНТУ, 2024 р. – 847 с.

Збірник включає матеріали доповідей учасників конференції, які об'єднані за тематичними напрямками конференції.

Збірник буде корисним як для фахівців і працівників фірм, зайнятих в області IT та автоматизації, так і для викладачів, магістрів і студентів вищих навчальних закладів, які навчаються за напрямами і спеціальностями програмного забезпечення, обчислювальної техніки і автоматизованих систем, прикладної математики та обробки інформації, буде корисним професіоналам з комп'ютерного моделювання та розробки комп'ютерних ігор.

Результати досліджень у збірнику представляють собою своєрідний зріз сучасного стану справ в перерахованих галузях знань, який може допомогти як фахівцям, так і студентам університетів скласти загальну картину розвитку інформаційних технологій та пов'язаних з ними питань.

Наукові праці згруповані за напрямками роботи конференції та наведені в алфавітному порядку прізвищ авторів.

Матеріали (тези доповідей) друкуються в авторській редакції. Відповідальність за якість та зміст публікацій несе автор.

Матеріали подано українською та англійською мовами.

Головний редактор збірника Сергій Котлик

УДК 004.428

АВТОМАТИЗАЦІЯ РОЗСИЛКИ EMAIL-ПОВІДОМЛЕНЬ ТА ПОВІДОМЛЕНЬ В МЕСЕНДЖЕРИ НА ОСНОВІ АНАЛІЗУ ПОДІЙ. МІКРОСЕРВІСНИЙ ПІДХІД

Лебідь Георгій, Іванов Леонід
(heorhii.lebid@nure.ua, leonid.ivanov@nure.ua)
Харківський національний університет радіоелектроніки (Україна)

У роботі розглянуто питання автоматизації систем повідомлень шляхом інтеграції з сучасними комунікаційними платформами. Представлено переваги мікросервісної архітектури для забезпечення масштабованості і гнучкості системи, технічні аспекти інтеграції, включаючи роботу з API, шифрування даних і механізми автентифікації. Особливу увагу приділено викликам, які виникають під час інтеграції. Розглянуто сучасні тенденції впровадження штучного інтелекту для підвищення ефективності комунікацій та адаптування їх до потреб користувачів.

Актуальність даної розробки обумовлена швидким розвитком технологій і збільшенням обсягів даних, які потребують оперативної обробки та передачі. Для бізнесів критично важливо миттєво й ефективно інформувати клієнтів, співробітників або партнерів про різні події - зміни статусів замовлень, активацію нових функцій, попереджати про технічні збої тощо.

Особливе місце у цьому процесі займає автоматизація розсилок електронних повідомлень та інтеграція з популярними месенджерами. Автоматизовані системи розсилки повідомлень дозволяють значно підвищити ефективність роботи за рахунок зниження кількості механічних помилок, що забезпечить доставку повідомлень точно і своєчасно. Це неможливо без впровадження автоматизованих рішень.

Автоматизована система розсилки повідомлень також дозволяє оптимізувати внутрішні процеси в компаніях, скоротивши час на комунікацію; це забезпечує підвищену гнучкість функціонування бізнесу та масштабованість системи в умовах динамічного розвитку компаній і збільшення навантаження на IT-інфраструктуру.

Розробка програмного забезпечення для автоматизованої розсилки з використанням мікросервісної архітектури є перспективним напрямком, оскільки така архітектура дозволяє легко масштабувати рішення, модернізувати окремі компоненти системи без необхідності зупинки всієї служби, а також забезпечує більш високу надійність та ефективність роботи за рахунок спрощення інтеграції з різними платформами та API, що є важливим для забезпечення багатоканальної комунікації [1].

Методи вирішення проблеми

На сьогоднішній день існує багато платформ для комунікацій, кожна з яких має свої особливості та переваги. Основними способами офіційної комунікації є електронна пошта та месенджери, що забезпечують швидкий та менш формальний обмін повідомленнями. Ці платформи дозволяють забезпечити багатоканальну комунікацію, адаптовану до вподобань користувачів. Завдяки інтеграції автоматизованих систем сповіщень із різними платформами можливо охопити всі ці канали, забезпечивши гнучкість комунікації [2].

Перевагами такої інтеграції є миттєве отримання повідомлень, що особливо важливо для критичних подій; обробка великих обсягів інформації без втрати продуктивності; зменшення ручної обробки; централізоване керування повідомленнями та контролювання з одного джерела, що забезпечить єдиний підхід до інформаційної політики.

Технічно інтеграція автоматизованих систем сповіщень з комунікаційними платформами здійснюється через програмні інтерфейси (API).

Для email-розсилок використовуються стандартні протоколи, такі як SMTP (Simple Mail Transfer Protocol), або спеціалізовані API сторонніх сервісів, таких як SendGrid чи Mailgun. Використання API дозволяє гнучко керувати відправкою повідомлень, динамічно змінювати контент, аналізувати доставку та вести статистику відкриття та кліків. Різні платформи можуть використовувати різні формати передачі даних, зокрема JSON або XML, які потрібно коректно обробляти на стороні сервера [3].

Автентифікація та авторизація при роботі з API забезпечують безпеку переданих даних і запобігають несанкціонованому доступу. Більшість сучасних API використовують метод автентифікації OAuth 2.0 або API-токени, які потрібно передавати з кожним запитом. OAuth 2.0 забезпечує підвищену безпеку, оскільки дозволяє системам отримати доступ до сервісів від імені користувачів без необхідності зберігати їхні паролі. Крім того, API часто мають обмеження на кількість запитів у хвилину (rate limits), що вимагає належної оптимізації системи для уникнення перевантаження та недоступності сервісів. Не менш важливою є валідація (перевірка дійсності) запитів і захист від потенційних загроз, таких як атаки типу «людина посередині» (MITM), для чого використовується шифрування SSL/TLS.

Організація обробки подій та чергування повідомлень є ще одним критичним аспектом інтеграції. Для цього часто використовуються механізми черг повідомлень, такі як RabbitMQ, Kafka чи Amazon SQS, які дозволяють розподіляти навантаження між кількома серверами, забезпечуючи паралельну обробку подій у режимі реального часу: системи черг дозволяють обробляти події у порядку їхнього надходження, розподіляти навантаження на різні компоненти системи, забезпечувати високу продуктивність навіть при пікових навантаженнях. Система має можливість контролювати кожен із цих етапів [4].

Критично важливим аспектом є ретельний захист від кібератак, витоків даних, несанкціонованого доступу та забезпечення конфіденційності. У сучасному цифровому середовищі будь-які збої в безпеці можуть призвести до серйозних репутаційних наслідків, юридичних претензій або втрати довіри клієнтів. Одним із ключових елементів безпеки є шифрування даних під час передачі між системою та комунікаційними платформами. Шифрування SSL/TLS (Secure Socket Layer/Transport Layer Security) захищає дані від атак типу «людина посередині» (MITM) і гарантує, що дані будуть передаватися через захищене з'єднання, яке унеможливлює перехоплення повідомлень третіми сторонами. Крім того, важливо забезпечити шифрування і на стороні сервера. Для цього використовуються методи шифрування баз даних, які забезпечують високий рівень захисту і збереження персональних даних користувачів (AES - Advanced Encryption Standard).

Контроль доступу та автентифікація є наступним критично важливим аспектом безпеки. Сучасні API комунікаційних платформ використовують методи автентифікації (OAuth 2.0) без необхідності використовувати паролі користувачів, що дозволяє передавати доступ безпосередньо через безпечні маркери доступу (access tokens). OAuth 2.0 дає можливість системам взаємодіяти від імені користувачів, контролюючи доступ до ресурсів і обмежуючи його певними дозволами. Ключі API, що використовуються для автентифікації, повинні бути надійно захищені і не зберігатися у відкритому вигляді. Крім того, важливо впроваджувати багаторівневу

автентифікацію, наприклад, двофакторну автентифікацію (2FA), щоб підвищити рівень захисту акаунтів [5].

Для забезпечення безпеки необхідно реалізувати чітку політику доступу до різних компонентів системи, обмежуючи права користувачів і системних компонентів лише необхідними для їхньої роботи. Сегментація прав доступу зменшує можливі ризики у разі компрометації окремих компонентів системи. Регулярний аудит прав доступу має запобігти потенційним загрозам.

Незважаючи на численні переваги, інтеграція з різними платформами має свої труднощі, пов'язані з технічними, безпековими та операційними аспектами, а саме:

– кожна комунікаційна платформа має свої специфічні вимоги та можливості API, що ускладнює інтеграцію, оскільки для кожної платформи потрібно налаштовувати окремі процеси. Різні обмеження щодо обсягу даних, структури запитів або обмеження частоти викликів (rate limits) API кожної платформи можуть вплинути на продуктивність системи під час великого навантаженням. Потрібно забезпечити сумісність і стабільність роботи системи на всіх платформах, адаптуючи їх специфічні особливості;

– забезпечення масштабованості системи у режимі реального часу може призвести до збільшення часу доставки повідомлень або навіть до їх втрати. Для вирішення цієї проблеми використовують системи черг повідомлень (RabbitMQ або Kafka), але їхнє впровадження додає складнощів у керуванні чергами та гарантіях доставки. У разі збою чи перевантаження одного з компонентів система повинна вміти автоматично перенаправляти або повторно надсилати повідомлення, що також потребує додаткового налаштування і тестування;

– унікальні вимоги до регуляції роботи з даними (правові, юридичні) на різних платформах та у різних країнах. Для глобальних проєктів, які мають справу з великою кількістю міжнародних користувачів, це створює додаткові труднощі;

– постійне відслідковування оновлення API комунікаційних платформ. Зміни чи додавання функції, модифікування механізмів автентифікації або обмеження частоти запитів змушує розробників регулярно оновлювати свої системи, щоб підтримувати інтеграцію в актуальному стані. Недотримання цих вимог може призвести до того, що деякі функції перестануть працювати або інтеграція взагалі стане неможливою. Це вимагає постійного технічного супроводу системи та додаткових ресурсів для адаптації до нових умов.

Активне впровадження штучного інтелекту (AI) і машинного навчання (ML) у системи сповіщень - сучасна тенденція розвитку інтеграційних технологій. AI здатний автоматично адаптувати контент під конкретну аудиторію, підвищуючи персоналізацію й ефективність комунікації. Інтелектуальні алгоритми можуть аналізувати поведінку користувачів, прогнозувати, коли і яким чином краще надіслати повідомлення для підвищення залученості, що робить систему більш автономною і продуктивною.

Висновки

Автоматизація систем сповіщень на основі інтеграції з сучасними комунікаційними платформами з метою забезпечення ефективної та швидкої комунікації в умовах сучасного інформаційного середовища є нагальною потребою, реалізація якої дозволить досягти високої гнучкості у взаємодії бізнесу з користувачами, підвищити швидкість обробки запитів та значно скоротити ручну обробку даних. Використання мікросервісної архітектури, систем черг повідомлень та API дозволить масштабувати рішення і адаптувати їх до потреб користувачів. Проте, інтеграція з такими системами стикається з викликами – від технічних аспектів роботи API до проблем із забезпеченням безпеки та відповідності міжнародним регуляціям.

Список використаної літератури

[1] Yevsieiev V. Researching Cyberattacks Methods in Industrial Internet of Things / V. Yevsieiev, N. Demska // Виробництво & Мехатронні Системи 2021 : матеріали V-ої Міжнар. конф., 21-22 жовтня 2021 р.: тези доп. – Харків: [електронний друк], 2021. – С. 18–20.

[2] Features of Wave Algorithm Application in Warehouse Logistics Transport Systems / I. Nevliudov, V. Yevsieiev, S. Maksymova S. et all // Information systems in project and program managemen: collective monograph / ed. by I. Linde; European University Press. – Riga: ISMA, 2023. – P. 251-261.

[3] Yevsieiev, V., & Gurin, D. (2023). COMPARATIVE ANALYSIS OF THE BASIC METHODS USED IN INDUSTRY 4.0 AND INDUSTRY 5.0. Collection of Scientific Papers «ΛΌГОΣ», (September 29, 2023; Bologna, Italy), 113–115. https://doi.org/10.36074/logos-29.09.2023.31.

[4] Ніколаєнко, І. В. Сутність CRM як категорії в маркетинговій діяльності / І. В. Ніколаєнко // Theory and Practice of Science : with the Proceedings of the 5 th International Scientific and Practical Conference, November 7-8. – Rome, Italy : Dana, 2021. – P. 97–109.

[5] Круглик В. С., Астаф'єв В. Ю. Особливості реалізації семантичної нейронної мережі створення генератора навчальних кросвордів. Вісник Кременчуцького національного університету імені Михайла Остроградського. Кременчук : КрНУ, 2021. Випуск 2(127), С. 81–88.

ДОДАТОК В

Демонстраційний матеріал

Формат	Позиція	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість	Примітки
			Текстові документи		
А4	1	ГЮІК 501340.009 ПЗ	Пояснювальна	85	с.
			записка		
			Інші документи		
	2		Додаток А	15	с.
			Лістинг програми		
	3		Додаток Б	8	с.
			Апробація результатів		
			наукових досліджень		
	4		Додаток В	16	с.
			Демонстраційний матеріал		

					ГЮІК 501340.009 ПЗ		
Змін	Арк.	№ докум.	Підпис	Дата			
Розробив		Лебідь Г.Т.			Розроблення комп’ютеризованої системи для автоматизації процесу розсилки email-повідомлень за результатами аналізу подій	Літера	Аркуш	Аркуш
Перевірив		Іванов Л.С.				Н	1	1
						Кафедра КІТАР ХНУРЕ		
Норм. контроль		Стародубцев М.Г.						
Затв.		Невлюдов І.Ш.						