Міністерство освіти і науки України

Національний університет «Києво-Могилянська академія»

Факультет інформатики

Катедра інформатики

Кваліфікаційна робота

освітній ступінь – бакалавр

на тему: «**ПОБУДОВА АВТОМАТИЗОВАНОЇ СИСТЕМИ ВИЯВЛЕННЯ АНОМАЛІЙ В БІЗНЕС-ДАНИХ**»

Виконав: студент 4-го року навчання,

Освітньої програми «Інженерія програмного забезпечення», 121

Постніков Михайло Андрійович

Керівник Гороховський С.С.

кандидат фіз.-мат. наук, доцент

Рецензент

_____

(прізвище та ініціали)

Кваліфікаційна робота захищена з оцінкою _____

Секретар ЕК

_____

«____» _____ 2024 р.

Київ – 2024

КАЛЕНДАРНИЙ ПЛАН ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ

№ п/п	Назва етапу кваліфікаційної роботи	Термін виконання етапу	Примітка
1.	Отримання завдання на кваліфікаційну роботу	до 30.11.2023	
2.	Дослідження предметної області та наявних рішень	20.01.2024	
3.	Огляд і вивчення наявних літературних джерел та досліджень	10.02.2024	
4.	Формування технічного завдання до програмної частини	20.02.2024	
5.	Написання програмної частини роботи	20.03.2024	
6.	Написання теоретичної частини роботи	10.04.2024	
7.	Погодження роботи з науковим керівником, внесення правок	20.04.2024	
8.	Підготовка матеріалів до захисту	07.05.2024	
9.	Захист кваліфікаційної роботи	до 31.05.2024	згідно з розкладом роботи ЕК

Тема роботи: Розробка автоматизованої системи виявлення аномалій в бізнес-даних

Студент Постніков М.А.

Керівник Гороховський С.С.

«___» _____ 2023 р

Міністерство освіти і науки України

Національний університет «Києво-Могилянська академія»

Факультет інформатики

Катедра інформатики

ЗАТВЕРДЖУЮ
Зав. катедри інформатики
_____ Гороховський С.С.
«__» _____ 2023 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ
на кваліфікаційну роботу студенту Постнікову Михайлу Андрійовичу
факультету інформатики 4 курсу бакалаврської програми

ТЕМА: Побудова автоматизованої системи виявлення аномалій в бізнес-даних

Зміст ТЧ до кваліфікаційної роботи:

- Вступ
- Розділ І. Аналіз предметної області
- Розділ ІІ. Вимоги до програмної реалізації
- Розділ ІІІ. Програмна реалізація системи
- Висновки
- Список літератури
- Додатки (за необхідністю)

Керівник
_____ Гороховський С.С.
(підпис)

Дата видачі
«__» _____ 2023 р.

Завдання отримав
_____ Постніков М.А.
(підпис)

ЗМІСТ

АНОТАЦІЯ

У роботі здійснено аналіз процесу виявлення аномалій, проведено огляд і систематизацію літератури за темою, наведено основні класифікації аномалій та технік їхнього визначення. Також проаналізовано поняття бізнес-даних, актуальної проблематики аналізу даних в сфері бізнесу.

Для розв’язання цих проблем було розроблено автоматизовану програмну систему виявлення аномалій. Було проведене дослідження існуючих програмних рішень та проаналізовано їхні переваги і недоліки, та сформульовано вимоги до системи. З урахуванням цих вимог і сучасних трендів в галузі було обрано технології і спроєктовано архітектуру, що лягли в основу програмної реалізації системи. Був даний опис отриманій системі і наведені перспективи її розвитку.

Мета роботи: побудова автоматизованої системи виявлення аномалій в бізнес-даних.

Ключові слова: виявлення аномалій, програмна система, вимоги до системи, архітектура, технології, веб-додаток, FastAPI, Celery, REST API, Pandas.

ВСТУП

«Дані – нафта 21 століття» - ці слова, сказані британським математиком Клівом Гамбі у 2006 році [1] *(тут і далі переклад наш – М. А.)* ознаменували собою головну характеристику сучасного цифрового світу. Дані сьогодні є найбільш цінним і затребуваним ресурсом для будь-якої компанії на ринку. Вони використовуються, щоб досліджувати поведінку користувачів, проводити A/B тестування, робити прогнози і вирішувати подальший хід цифрових продуктів . Але дані не несуть прямої користі в тому вигляді, в якому вони є – для застосування вони потребують обробки і інтерпретації [1].

Згідно з даними ресурсу Big Data Analytics News, у 2024 році ринок великих даних (big data) досягне об'єму у $84 млрд, а об'єм даних, що генерується щорічно, з 2023 до 2025 року зросте майже наполовину, і становитиме 181 зетабайт [2]. Згідно з даними цього ж ресурсу, 70% всіх цих даних були згенеровані користувачами. Класична характеристика великих даних – «об'єм, варіативність та швидкість» [3] *(тут і далі переклад наш – М. А.)* – чудово описують тенденції, що панують в цій області роками: об'єм і варіативність даних збільшується з кожним днем, і швидкість цього приросту з часом тільки зростає.

Таким чином, для забезпечення своєї діяльності компанії мусять базувати свою аналітику на даних, що стає дедалі важчим завданням через постійне зростання їхньої варіативності та об'єму. Головними проблемами тут є надійність даних (data reliability) та швидка ідентифікація зміни трендів. Проблема надійності даних є наріжною в індустрії, оскільки сучасні бізнеси оперують даними, що походять з різноманітних джерел (бухгалтерські системи, системи звітності, продажів, CRM системи, дані активності соціальних мереж, дані систем обліку співробітників і т.п.). Такі дані часто не є придатними до використання одразу, оскільки містять помилки та неточності. Знаходження таких помилок і неточностей є класичною проблемою аналізу даних. Також бізнес стикається з потребою швидко ідентифікувати незвичну поведінку даних,

що є індикатором певних суттєвих змін, які вносять корективи до бізнес-процесів. Вирізняти такі випадки серед великих обсягів даних швидко – складна задача, що складно вирішується за допомогою звичних практик аналізу даних.

Саме тут багато бізнесів звертаються до техніки виявлення аномалій (outlier detection), що допомагає визначати одиниці даних, які не слідують патерну поведінки інших даних. Ці методи широко застосовуються для вирішення проблем на кшталт оптимізації витрат, виявлення випадків шахрайства, тощо, і можуть бути використані для вирішення наведених вище проблем.

Проведене автором дослідження показало, що серед відкритого програмного забезпечення на сьогодні немає системи, яка могла б задовольнити потребу бізнесу у швидкому виявленні аномалій в найбільш поширеній формі бізнес-даних, а саме – в часових рядах. Ця проблема і стала поштовхом для написання даної роботи.

Об'єкт дослідження: виявлення аномалій та особливості цього підходу для аналізу бізнес-даних (часових рядів), існуючі системи виявлення таких аномалій, актуальні технології та програмна реалізація системи.

Мета дослідження: провести аналіз і систематизувати дослідження для процесу виявлення аномалій, описати особливості виявлення аномалій в даних бізнесу, для вирішення поставленої проблеми на основі досвіду існуючих розробок сформулювати вимоги до програмної системи, провести проєктування архітектури і вибір технологій, та розробити систему автоматизованого виявлення аномалій відповідно до цих вимог.

РОЗДІЛ І. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Загальна інформація про виявлення аномалій

Станом на сьогодні можна сказати, що виявлення аномалій (outlier detection, також anomaly detection) – це усталена галузь науки про дані (data science), що досліджує підходи і методи вирішення проблеми визначення аномалій в даних. Однією з перших відомих наукових робіт на цю тему прийнято вважати статтю викладача Королівської школи Лондона Ф. Еджворс (Francis Edgeworth) «On discordant observations» [5], де автор визначає дискордантні спостереження як ті, що «...справляють враження відмінності від інших спостережень відносно до закону, за яким розподілені їхні частоти» *(тут і далі переклад наш – М. А.)*. З плином часу дослідженням цієї проблеми приділялось дедалі більше уваги.

Перші ґрунтовні напрацювання щодо визначення аномалій були зроблені дослідниками в галузі статистики. Так, класичне визначення аномалії тут наводить Ф. Ґрабс (Frank Grubbs): «Аномальним спостереженням, або “аномалією”, є те, що помітно виділяється серед інших членів вибірки, де воно зустрічається» [6, с. 1] *(тут і далі переклад наш – М. А.)*. Автор виділяє два типи аномальних спостережень: ті, «...що є серйозним свідченням про варіативність даних випадкового характеру...» [6, с. 1], та ті, «...що є результатом серйозних відхилень від процедури [статистичного] експерименту або помилки в підрахунках чи записах числового значення» [6, с. 1]. Дослідник наводить опис розроблених ним критеріїв визначення аномалій в статистичних даних, та зазначає, що «...майже всі [розроблені] критерії для аномалій ґрунтуються на передбаченні про нормальний (Ґаусовий) розподіл або популяцію, що лежить в основі [даних]» [6, с. 3]. Додатково автор підкреслює, що знаходження аномалій служить насамперед індикатором потреби подальшого дослідження і виявлення причини їхньої появи, оскільки наявність аномалій не обов’язково означає, що такі спостереження потребують особливого поводження чи мають бути відкинуті [6, с. 20-21].

Ґрунтовним дослідженням визначення аномалій в статистичних даних стала монографія Д. Гокінза (Douglas Hawkins) «Identification of Outliers» [7]. За Гокінзом, аномалія – це «...спостереження, що настільки сильно відхиляється від інших спостережень, що виникає підозра, що воно було згенероване іншим механізмом» *(тут і далі переклад наш – М. А.)* [7, с. 1]. Автор виділяє два основних механізми походження аномалій: коли дані походять від розподілу з «важкими хвостами» (heavy-tailed distributions), та коли дані походять від двох розподілів, один з яких генерує нормальні дані, а інший – аномалії [7, с. 1-2].

Найбільш значимою роботою останніх років у сфері стало дослідження науковців з Університету Мінесоти В. Чандола (Varun Chandola), А. Банерджі (Arindam Banerjee) та В. Кумара (Vipin Kumar), опубліковане в 2009 році [8], де було проаналізовано понад сотню досліджень і підсумовано наявні звершення в цій галузі. Так, дослідники дають визначення аномаліям як «...патернам в даних, що не підкорюються поняттю про нормальну поведінку» [8, с. 2].

Ілюстрація прикладу аномалій серед двовимірних даних наведена на Рис. 1.1.1., де області N1, N2 – кластери нормальних значень, кластер A1 і точка A2 – аномальні значення.

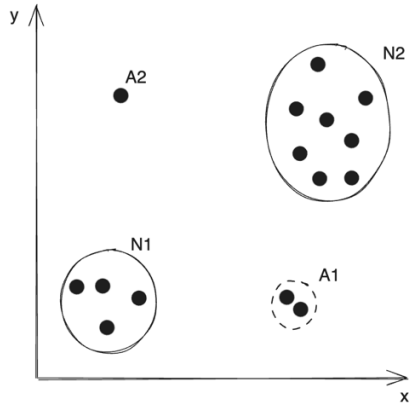

Рис. 1.1.1. Приклад аномалій в двовимірних даних

Таким чином, можна зробити висновок, що всі дослідники дають схожі визначення аномаліям – це ті дані, що *помітно* відрізняються серед інших схожих даних. Наявність аномалій сама по собі говорить лише про наявність цієї незвичності, і не є результатом, що несе практичний сенс, оскільки потребує додаткового дослідження причин таких аномалій.

1.2. Застосування виявлення аномалій

Через те, що аномалії – широке поняття, техніки виявлення аномалій мають різноманітне практичне застосування. Наведемо основні приклади:

- *Виявлення шкідливої поведінки*. Незвична активність, неочікувані зміні у поведінці людей або комп'ютерних систем можуть вказувати на злий умисел таких дій, і за допомогою визначення аномалій можна виявляти таку активність. Наприклад, банки таким чином виявляють випадки зламу платіжних карток або подання неправдивої інформації, соціальні мережі – випадки злому акаунтів, адміністратори комп'ютерних мереж – випадки несанкціонованого проникнення в систему [8, с. 13; 10, с. 1; 11, с. 87];
- *Виявлення дефектів*. Аномалії у будові матеріалів пристроїв або характеристик готових виробів можуть означати проблеми у виробництві або експлуатації. Так, визначення аномалій використовується при обстеженні літаків за допомогою іонізуючого сканування, тут аномалії позначатимуть структурні зміні у металі. Виявлення аномалій в параметрах виробів серед допустимих відхилень від еталонних може свідчити про проблеми на виробничій лінії. Аномальний дефект у тканинах організму може означати початок запального або онкологічного процесу у тканинах [8, с. 16; 11, с. 87];
- *Виявлення новизни (novelty detection)*. Важливо наголосити, що виявлення новизни є окремою самостійною темою, але пов'язаною з виявленням аномалій, тому методи виявлення аномалій можуть бути використані і тут. Приклади такого використання: виявлення неочікуваних даних серед наборів даних для машинного навчання для їхньої фільтрації, аномалії в концентрації або пересуванні військ свідчать про наміри противника, а зміни в поведінці користувачів можуть означати новий тренд, а зміна в поведінці часового ряду означатиме вплив неочікуваних факторів [8, с. 19; 10, с. 5; 11, с. 87].

Загалом, виявлення аномалій може бути застосоване у будь-якій області, де присутня потреба відстежувати зміни в даних. Особливо корисною вона буде в тих випадках, де на аналіз усіх потрібних даних не вистачає ресурсів. Виявлення аномалій може бути «на сторожі» певних важливих параметрів, сповіщаючи лише тоді, коли дійсно потрібна додаткова увага. Ключовим є те, що виявлення аномалій не є інтерпретацією даних: аномальні дані не завжди є чимось добрим або поганим, вони не завжди означають, що щось пішло не так, як планувалось. Вони дають знати про критичну зміну в поведінці певних показників. Як і з будь-яким моделюванням, аномалії не є стовідсотково точними: так, можливість хибнопозитивних (false-positive) чи хибнонегативних (false-negative) результатів залежить від якості вхідних даних, їхньої попередньої обробки, вибраної моделі і її параметрів, варіативності і обсягу вибірки.

1.3. Класифікації області визначення аномалій

Зважаючи на розповсюдженість проблеми аномалій, існує багато класифікацій, в яких відображені різні підходи до цієї проблеми. Нижче наведемо основні широко прийняті такі класифікації.

Почати слід з характеристик самих даних. Визначення аномалій відбувається серед певного набору даних, саме тому походження і характеристики вхідних даних мають найбільший вплив на визначення аномалій *(тут і далі поняття одиниця даних, спостереження, експеримент, точка використовуються синонімічно на позначення самостійної одиниці набору певних величин, в якому потрібно виявити аномалії – М. А.)*. Так, за кількістю атрибутів вхідні дані поділяють на *уніваріативні* – мають один атрибут (напр., ціна товару) , та *багатоваріативні* – мають кілька атрибутів (напр., технічні характеристики автомобіля) [8, с. 6; 9, с. 132]. Дані також різняться між собою за відношенням між входженнями: найчастішими прикладами тут можуть бути *вектори*, *послідовності*, *просторові* та *графові* дані [8, с. 7; 10, с. 2]. Велике значення для вибору техніки визначення аномалій має їхній *розподіл*: якщо він відомий (напр., нормальний розподіл, що часто зустрічається серед статистичних

даних), то це робить можливим застосування статичних методів визначення [8, с. 33; 9, с. 132].

За типом пов’язаності виділяють такі категорії аномалій:

- *Точкові* – ті, що є аномаліями незалежно від інших точок [8, с. 7; 10, с. 2]. Такі аномалії є найпростішими для знаходження. Наприклад, розмір яблука можна класифікувати як аномальний на основі відомого розподілу розмірів в популяції, і це не залежатиме від розміру інших яблук в даному наборі;
- *Контекстуальні* – ті, аномальність яких залежить від сусідніх точок [8, с. 7-8; 10, с. 2]. Ці аномалії є куди більш складними для знаходження, оскільки вимагають аналізу одразу кількох спостережень. Тут спостереження часто пов’язані між собою. Найчастішими з контекстуальних аномалій є часові ряди [8, с. 8]: наприклад, аномальність інфляції гривні в певний момент часу буде залежати від аномальності інших точок (попередні та наступні спостереження, сезонність і т.п.);
- *Колективні* – ті, аномальність яких залежить від усієї вибірки [8, с. 9]. Наприклад, аномальність ціни на товар серед конкурентів залежатиме від того, який розподіл має вибірка цін на цей товар, і змінюватиметься відносно того, як змінюватиметься вся вибірка.

За ступенем контролю над визначенням аномалії виділяють такі типи технік:

- *Контрольовані (supervised)* – передбачають наявність вибірки даних, аномальні і нормальні точки в якій вже промарковані [8, с. 10; 10, с. 3]. На основі них робиться моделювання, і ця модель використовується для визначення аномалій. Класичним прикладом тут є нейронні мережі: мережу навчають на певному наборі даних, мережа засвоює патерни поведінки даних, і може робити прогноз стосовно нових спостережень. Це найбільш вимогливий різновид технік визначення аномалій, оскільки вимагає найбільше зусиль через необхідність класифікації, але є дуже точним;

- *Напівконтрольовані (semi-supervised)* – ті, що передбачають необхідність маркування лише для нормальних спостережень [8, с. 10; 10, с. 3]. Тут також використовуються нейронні мережі, але в навчанні беруть участь лише нормальні спостереження. Мережа засвоює патерни поведінки нормальних даних, а будь-яка інша поведінка класифікується як аномалія;
- *Неконтрольовані (unsupervised)* – не передбачають ніякого маркування даних [8, с. 11; 10, с. 3]. Моделювання відбувається на основі некласифікованих даних. Найбільш широко вживаний механізм визначення аномалій через легкість реалізації, але поступається попереднім методам у точності. Часто такі методи застосовуються там, де про вхідні дані заздалегідь нічого не відомо.

За результатом визначення техніки бувають: *дискретними (score)* – на виході дають числову оцінку аномальності, та *категорійними (label)* – точки позначаються лише як аномальні або не аномальні [8, с. 11].

1.4. Техніки визначення аномалій

За історію напрямку напрацьовано багато різних технік виявлення аномалій. Вони різняться передусім за рахунок предметної області застосування, характеру вхідних даних, бажаної точності передбачень та об’ємом зусиль на впровадження. Нижче наведемо основні з них.

Статистичні методи полягають у тому, щоб застосовувати до даних статистичні моделі, припускаючи, що дані можна пояснити за допомогою тих чи інших статистичних технік [10, с. 3]. Припущення полягає в тому, що дані, які згенеровані певною технікою, є нормальними, а ті, що не вдалось згенерувати за допомогою даної техніки – аномальними [8, с. 33]. Наприклад, припускаючи, що дані слідують нормальному розподілу, можна визначити порогові значення, скажімо, 2σ, і класифікувати дані, що лежать за межами цих значень, як аномалії. Окремими частковими випадками статистичних методів є *параметричні* та *непараметричні* методи [8, с. 32; 10, с. 3]. *Параметричні* методи передбачають те, що дані слідують певному розподілу, поведінку якого можна визначати за

допомогою параметрів, які можна дізнатись за допомогою наявних даних. Прикладами таких методів є Ґаусові та регресійні моделі [8, с. 36]. Для їхнього використання потрібно на основі даних припустити, за допомогою якого розподілу вони згенеровані, визначити параметри для цього розподілу, і значення, які не будуть потрапляти в результати розподілу, вважатимуться аномаліями. *Непараметричні* методи передбачають те, що не робиться жодних припущень щодо розподілу даних – натомість, для моделювання даних використовуються статистичні моделі, що не містять параметрів, вони вивчають розподіл даних [8, с. 37; 10, с. 3]. На основі проведених досліджень даних визначаються критерії аномальності. Прикладами таких методів є гістограмні та ядрові методи. *Гістограмний метод* полягає в наступному: за допомогою гістограми будується частотний профіль даних (тільки аномальних або тільки нормальних), він використовується для порівняння з гістограмою тих даних, які ми хочемо аналізувати, і оцінка аномальності випливає з різниці частотних профілів даних [8, с. 37; 10, с. 4]. *Ядрова оцінка густини* (kernel density) використовується дуже схожим чином; можна сказати, що цей метод є гістограмним, який використовує згладжування, за рахунок чого вирішується проблема впливу к-ті стовпчиків гістограми на дані [8, с. 38; 10, с. 4].

Класифікаційні методи використовують модель класифікатора, яку можна натренувати на розмічених тренувальних даних, і яка на основі цього буде класифікувати дані як аномальні або нормальні [8, с. 20]. Практичним застосуванням таких методів є різноманітні варіанти нейронних мереж. Наприклад, тут використовуються мережі, що вивчають правила, за якими розподілені тестові дані (rule-based learning), і застосовують ці правила для оцінки даних. Також тут використовуються баєсові мережі [8, с. 22]. Прикладом таких методів може бути класифікація зображень нейронною мережею: припустімо, що нам потрібно визначати аномалії серед набору фотографій квіток, це будуть види, яких ми не очікуємо побачити. Тоді необхідно підготувати набір розмічених тренувальних даних, які подаються на вхід нейронної мережі, і на основі яких мережа вивчає характеристики квіток

існуючих видів. Після навчання мережа використовується для визначення виду квітки на зображенні, яке ми хочемо класифікувати: якщо мережа не знайшла для квітки існуючий вид з тих видів, які ми використовували в наборі даних, то це вважатиметься аномалією.

Кластеризаційні методи використовують поділ вхідних даних на групи за певними ознаками, а ті дані, що не потрапили до жодної з груп, є аномаліями [8, с. 30; 10, с. 7]. Найчастіше такі методи є неконтрольованими, тобто, не підтримують використання розмічених тренувальних даних для оптимізації алгоритму. Такі методи базуються на припущенні про те, що спостереження є пов'язані між собою за якимись ознаками, і якщо точка лежить далеко від якогось з таких кластерів, це означає її ненормальну природу, тобто, можна вважати її аномалією. Прикладами таких алгоритмів є CLARANS, DBSCAN, BIRCH [8, с. 30; 10, с. 13].

Відстаневі методи визначення аномалій використовують міру відстані між спостереженнями, яка потім використовується для класифікації аномальності [8, с. 25; 10, с. 6]. Ідея полягає в тому, що для даної вибірки нормальні точки будуть мати малу відстань між собою, а аномальні будуть знаходитись на великій відстані від інших точок [8, с. 25]. Цю ідею також часто називають принципом сусідньості (neighborhood) або близькості спостережень. Для використання методу необхідно вибрати функцію дистанції точок, часто для цього використовують Евклідову відстань Найбільш популярними методами тут є техніки kNN (k близьких сусідів) та LOF (локальний фактор аномальності). Найпростіша реалізація техніки kNN використовує припущення про те, що «...оцінка аномальності одиниці даних є відстань до k-того найближчого сусіда до даної одиниці з вибірки» [8, с. 25], але за показник аномальності тут можна брати і більш складні метрики, такі як сума відстаней до k найближчих сусідів.

РОЗДІЛ ІІ. ВИМОГИ ДО ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

2.1. Проблеми аналітики бізнес-даних

Важливість даних для бізнесу зростає з кожним днем. Сьогодні складно уявити бізнес, який не робив би аналітику своїх показників, бо саме це дозволяє бізнесу розуміти, в якому напрямку рухається ринок, чи збігається цей напрямок з напрямком руху компанії, що треба змінювати, щоб «тримати хвилю». Дослідники з Університету Кембриджа повторюють фразу, сказану К. Гамбі у 2006 році, що «дані – це нафта 21 століття» [1, 12] *(тут і далі переклад наш – М. А.)*, і наголошують, що «...капіталізація цього вибуху [великих даних] стає необхідною умовою для бізнесу, щоб лишатись конкурентним. [...] Для бізнесу сьогодні ефективне використання даних розглядається не тільки, як конкурентна перевага, а і як необхідність для виживання» [12]. Автори дослідження говорять про те, що інтерпретація даних для отримання з них користі є основною задачею бізнесу, і це зробити зовсім непросто. Зокрема, одною з проблем тут є якість даних *(поняття «якість даних» тут і далі використовується як на позначення власне якості даних, англ. data quality, так і цілісності даних, англ. data integrity, так і інших термінів, що мають під собою здатність даних бути використаними для отримання правильних висновків – М. А.)*. За даними авторів дослідження, 71% бізнесу стикається з проблемами якості даних, і називають це «бар'єром для досягнення цілі *(отримання користі з даних – М. А)*» [12].

Бізнесу необхідно швидко і правильно реагувати на нові дані, для того, щоб відповідати змінам у сучасному світі. Ця задача з кожним днем стає дедалі складнішою, адже об'єм і варіативність даних тільки зростають. Велика частина цих даних належить до *часових рядів* (time series data). За особистим досвідом автора, більшість потреб бізнесу у даних задовольняють дані, що відносяться до часових рядів: ці дані представляють зміну певних показників з часом, наприклад, це може бути кількість продажів, дохід, кількість активних користувачів, маркетингові показники, такі як retention rate (відношення

користувачів, що лишились користуватись продуктом після певного часу). Цих даних на сьогодні набагато більше, ніж можливостей у більшості бізнесів для їх аналізу. Дослідники з SRM Institute of Science and Technology та VIT Bhopal University виділяють такі характеристики часових рядів: тренд, сезонність, циклічність та випадкову компоненту (шум), і наголошують, що аналіз часових рядів дозволяє бізнесу покращити процес прийняття рішень за рахунок кращого розуміння трендів на ринку [13] *(тут і далі переклад наш – М. А.)*.

Таким чином, особливості бізнес даних можна сформулювати так: *необхідність визначати незвичну інформацію серед великої кількості даних, представлених у вигляді часових рядів.* Тобто, метою програмної системи буде *дати можливість легкого визначення аномалій у часових рядах*. Система має бути інструментом, за допомогою якого можна *швидко* і *легко* налаштувати відслідковування аномалій в даних, які є у бізнесу.

2.2. Дослідження існуючих систем

Дослідження наявних систем дозволить краще зрозуміти можливі шляхи реалізації власної системи, урахувати їхні помилки і переваги, та звернути увагу на моменти, які потребували б ретельного опрацювання. До переліку систем для дослідження були відібрані ті, головна мета яких – відслідковування і сповіщення про аномалії в даних *(позначимо їх терміном «основні системи» - М. А.)*. Також у перелік потрапили системи з числа популярних в спільноті спеціалістів з даних, для яких виявлення аномалій – не головна мета, але вони мають такий функціонал *(позначимо їх терміном «побічні системи» - М. А.)*. Таким чином, у дослідженні розглянуто 2 основних та 1 побічну систему, всього 3 програмних продукти.

ChaosGenius є найновішою з подібних систем. З’явившись в доступі наприкінці 2021 року, цей продукт встиг пережити як свій розквіт, так і занепад. Наразі проєкт не підтримується: з 2022 року не було зроблено ніяких функціональних змін [14]. ChaosGenius є продуктом з відкритим кодом (open source), розміщеним на GitHub, і кожен може долучитись до його розробки.

Проєкт зібрав більше 700 зірочок на GitHub, до його розробки встигли долучитись більше 30 людей [14].

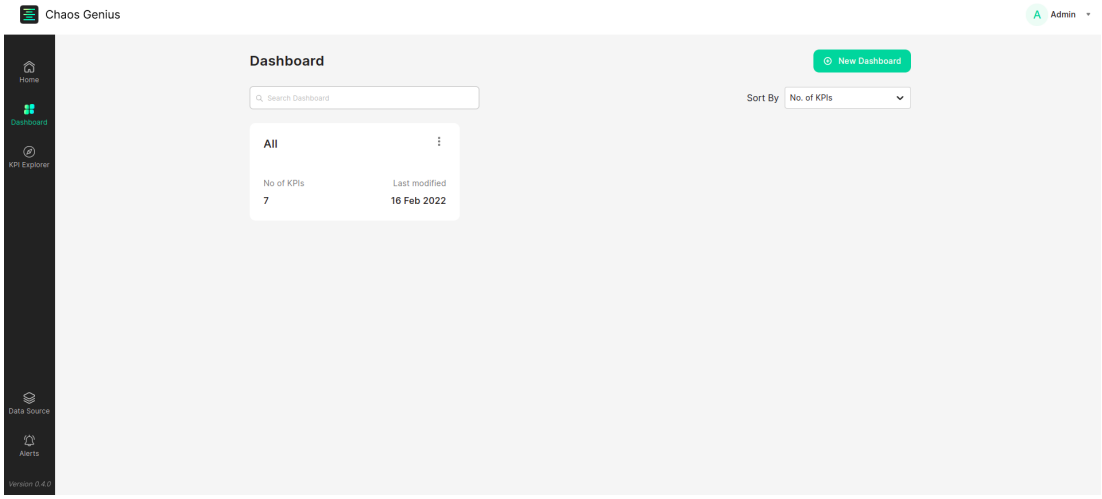

Рис. 2.3.1. Головний екран ChaosGenius [15]

Продукт позиціонується як «система прозорості бізнесу [...] що дозволяє бізнесу відслідковувати свої KPI в багатьох джерелах даних та здійснювати аналіз першопричин *(англ. root cause – М. А.)* відхилень в KPI» [15]. Система підтримує підключення до наступних баз і сховищ даних: Postgres, MySQL, BigQuery, Redshift, Snowflake, Druid, Databricks, Google Sheets, Google Analytics, Shopify, Stripe, Bing Ads, Facebook Marketing, Google Ads та AWS Athena. Відстеження аномалій відбувається через функціонал KPI *(укр. ключовий показник ефективності, англ. key performance indicators; тут і далі переклад наш – М. А.)*. KPI в цій системі – це абстракція на позначення сукупності метрик, що беруться з даних з підтримуваних сховищ. Додавання відбувається через простий графічний інтерфейс, зображений на Рис. 2.3.2: зі списку сховищ вибирається потрібне, в ньому зі списку доступних даних вибирається потрібний набір; наявна також можливість написання власного SQL запиту. Для показнику наявна можливість вибрати агрегацію: середнє, к-ть, або сума. Атрибутами показника є колонка з значенням часу, та колонки вимірів, що є числовими метриками, які потрібно аналізувати. Після додавання для KPI налаштовується відслідковування аномалій: для цього задаються параметри частоти даних (погодинно або поденно), вікна часу (використовується для тренування моделі, від 30 до 90 днів), частоти запуску моделі (раз на день або раз на годину), вибір

самої моделі, чутливість (вибір серед високої, середньої та низької) та година щоденного виконання (розклад) [15].

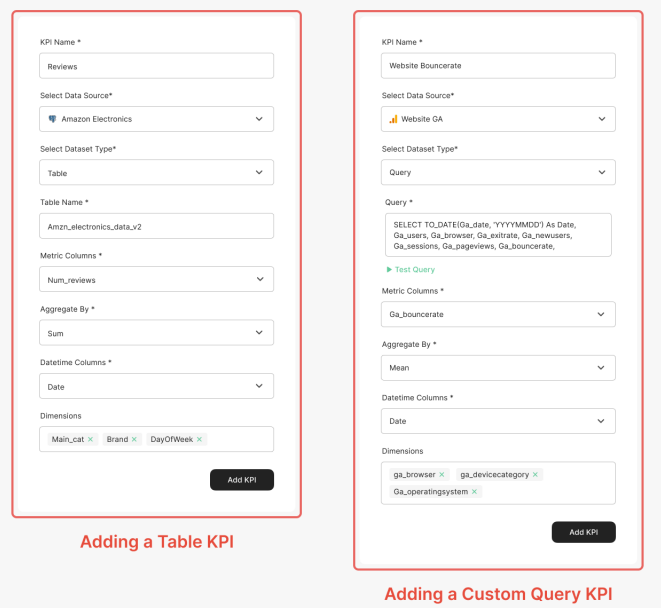

Рис. 2.3.2. Додавання KPI в ChaosGenius

Для визначення аномалій доступні такі моделі: Prophet, просте стандартне відхилення, експоненційно зважене стандартне відхилення (EWSTD), експоненційно зважене ковзке середнє (EWMA). Після налаштування параметрів, система запускає визначення аномалій щодня або щогодини, і якщо аномалії були знайдені, надсилає про це сповіщення в потрібний канал комунікації. Канал комунікації налаштовується і підключається окремо, це може бути або Slack (корпоративний месенджер), або електронна пошта [15].

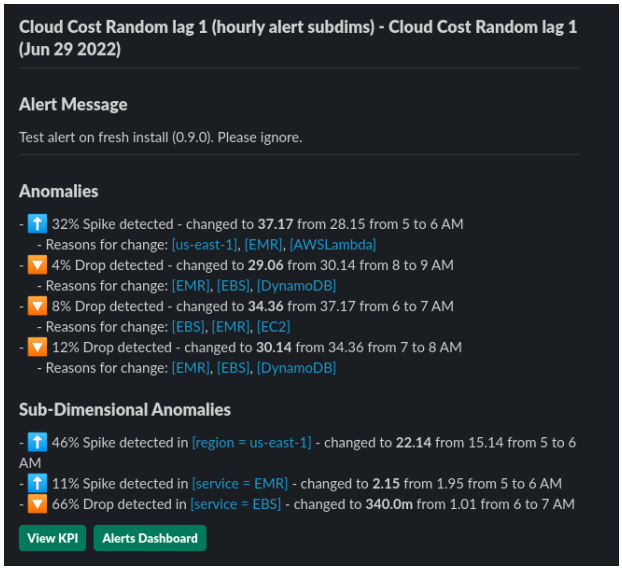

Рис. 2.3.3. Приклад сповіщення ChaosGenius в Slack

CueObserve була представлена публіці у 2021 році. Розповсюджується через GitHub як програмне забезпечення з відкритим кодом, так само як і з першою системою, кожен охочий може долучитись до її розробки. Система зібрала 200 зірочок на GitHub, наразі не підтримується, останні оновлення були зроблені в лютому 2022 року [16]. Позиціонується як система для «виявлення аномалій в часових рядах та аналіз першопричин в даних зібраних з використанням SQL зі сховищ та баз даних» [16].

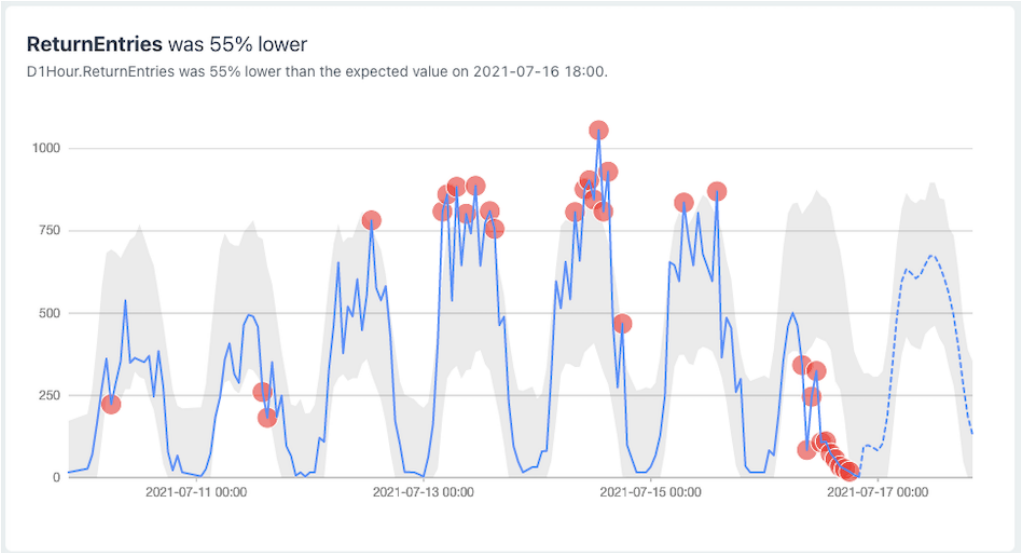

Рис. 2.3.4. Результат виявлення аномалій в CueObserve

CueObserve здійснює визначення аномалій за допомогою періодичного запуску моделі на вибірці бізнес-даних. Для налаштування потрібно вибрати набір даних, виміри (опціонально) і потрібну метрику, вибрати модель і розклад запуску. При використанні вимірів система підтримує фільтрацію значень для них за наступними ознаками: топ N значень у вимірі, мінімальний відсоток значень, що має лишитись після фільтрації, мінімальне середнє відфільтрованих значень. Нетривіальна модель, яку можна вибрати для відстеження аномалій, є тільки одна: Prophet, а інші ж моделі досить примітивні: відсоток зміни величини, чи є значення найбільшим або найменшим за всю історію, та статичне порогове значення [17]. Система підтримує наступні джерела даних: Google BigQuery, Amazon Redshift, Snowflake, PostgreSQL, MySQL, Apache Druid та SQL Server [17]. Доступно використання SQL для вибору метрик. Як і перша система,

CueObserve надає можливість відправки сповіщень при виявленні аномалій в даних. Сповіщення можна відправити через Slack, або через електронну пошту.

Рис. 2.3.5. Налаштування сповіщень CueObserve для Slack

Надалі розгляньмо побічні системи, які не мають за головну мету відслідковування аномалій, проте в них наявний функціонал для цього. *Elementary Data* є продуктом для відслідковування якості даних, які будуються за допомогою DBT (data build tool, популярна бібліотека для трансформації даних в базах даних за допомогою SQL). Цей інструмент з'явився публічно у 2022 році і став популярним серед користувачів DBT: так, на GitHub сторінці проєкту зібрано більше 1.7 тис. зірочок [18]. Відстеження аномалій тут відбувається за допомогою DBT тестів – SQL запитів, які виконуються в базі даних. Інформація про запуск тестів зберігається в цій же БД. Запуск тестів за графіком можна налаштувати за допомогою допоміжного інструменту, який розробники пропонують на своєму сайті. Бібліотекою підтримуються наступні БД: Postgres, Google BigQuery, Amazon Redshift, Databricks, Snowflake, AWS Athena [19]. Відправка сповіщень можлива через Microsoft Teams або Slack [19]. Серед технік для визначення аномалій є лише статистична техніка, яка визначає нормальність величини згідно нормального розподілу [19]. Важливо відзначити, що цей інструмент тісно зв'язаний з DBT, і не буде працювати без використання DBT, що не завжди зручно. Версія системи з відкритим програмним кодом, що включає доповнення для DBT, утиліту для запуску тестів, та веб-дашборд для перегляду результатів, розповсюджується безкоштовно. Є можливість отримати розширені можливості з купівлею хмарної версії [19].

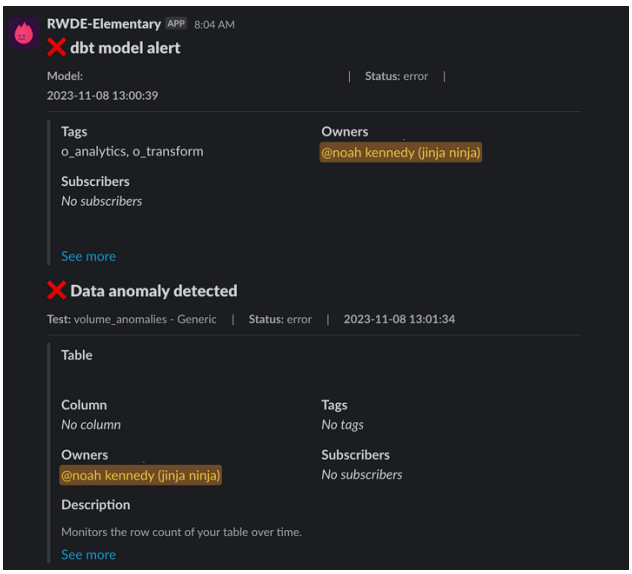

Рис. 2.3.6. Приклад сповіщення Elementary Data в Slack [20]

Серед особливостей всіх продуктів можна виділити наступні:

- Підтримка багатьох баз і сховищ даних. Найпопулярніші: Postgres, Google BigQuery, Amazon Redshift;
- Підтримка інтеграцій з різними системами для відправки сповіщень. Найпопулярніші: Slack та електронна пошта;
- Автоматизоване виявлення аномалій, яке запускається за розкладом. Найчастіше це раз на день;
- Наявність веб-інтерфейсу для перегляду інформації і налаштувань;
- Підтримка різних моделей визначення аномалій;
- Наявність налаштувань для моделей визначення аномалій.

Водночас, серед недоліків даних систем відзначимо наступні:

- Жодна з основних систем більше не підтримується станом на час проведення дослідження;
- Налаштування систем є заплутаним і неочевидним, зокрема, в процесі налаштування бракує інтерактивності і можливості подивитись, як налаштування впливають на результат;
- В системах мала кількість підтримуваних моделей визначення аномалій.

Але головними недоліками цих систем є їхня непристосованість до потреб бізнесу. Ці інструменти не дозволяють швидко і інтерактивно налаштувати відстеження аномалій, їхні налаштування не інтуїтивні і не дозволяють швидко розібратись в системі, або в тому, як найкраще налаштувати параметри визначення враховуючи особливості тих чи інших даних. Бізнес потребує інструмент, який дозволяв би за короткий час налаштувати відстеження аномалій в часових рядах, які періодично оновлюються. Це дозволило б бізнесу без особливих витрат часу і грошей слідкувати за багатьма показниками.

2.3. Вимоги до програмної реалізації

На основі проведеного дослідження, і з врахуванням особистого досвіду роботи автора в сфері даних протягом кількох років, були сформульовані вимоги до програмної реалізації. *Мета системи*: дозволити легко і швидко визначати аномалії в часових рядах за допомогою інтерактивних налаштувань. Нижче описані блоки функціональних та нефункціональних вимог до системи.

2.3.1. *Легкість у налаштуванні*. Система має мати веб-інтерфейс, через який буде відбуватись взаємодія користувача з нею, оскільки веб-інтерфейс є крос-платформеним, а також – стандартом де-факто в індустрії, як показало проведене автором дослідження. В веб-інтерфейсі можна буде дивитись створені конфігурації, запуски перевірок, результати визначення. Окремим важливим елементом веб-інтерфейсу має бути інтерактивний менеджер налаштування відстеження аномалій: він має поетапно супроводжувати користувача через процес налаштування, допомагаючи швидко і зручно додавати нові конфігурації запусків.

2.3.2. *Гнучкість і легкість кастомізації системи*. Як показало проведене дослідження, важливим для бізнесу є підтримка різних баз і сховищ даних, різних систем для надсилання сповіщень. Отже, система має мати базу для такої підтримки. Важливо зазначити, що це має бути передбачено архітектурно: процес додавання підтримки нової інтеграції в систему має бути максимально

простим. Звідси випливає і підтримка різних моделей визначення аномалій, яка теж має бути продумана архітектурно.

2.3.3. *Надійність*. Оскільки бізнес буде покладатись на таку систему у своїй діяльності, система має мати високі показники стійкості до помилок і відмов, і мати змогу обробляти великі масиви даних. Передусім це має бути забезпечено архітектурними рішеннями: наприклад, використання черг задач. Якщо якась перевірка не відбулась через якусь помилку, користувач має про це знати. Якщо є проблема в якомусь каналі комунікації, користувач також повинен про це дізнатись. Система має бути якомога більш чіткою і явною в своїй поведінці.

2.3.4. *Функціональні вимоги.* Система має мати функціонал для налаштувань відстеження аномалій, перегляду цих налаштувань, перегляду запусків відстежень, їхніх результатів та помилок, якщо такі були. Також важливим є зручний перегляд даних, який має бути реалізований у вигляді графіку. Всі повідомлення, що відправлені системою, мають дублюватись у самій системі для можливості відслідковування коректності роботи цього компоненту.

2.3.5. *Сфокусованість реалізації*. Проблема при реалізації будь-яких систем – правильна розстановка пріоритетів, тому окремим пунктом важливо зазначити і додатково підкреслити, що для виконання поставленої мети фокус при реалізації системи має бути зроблений на легкості і швидкості налаштувань. Для цього можна знехтувати точністю системи, можливостями до опрацювання різних вимірів даних (data dimensions), красою дизайну інтерфейсу або елегантністю вигаданих архітектурних рішень.

2.3.6. *MVP (мінімально життєздатний продукт)*. На думку автора роботи, зробити систему, яка чудово задовольняла би усі вимоги, мала б підтримку усіх популярних моделей виявлення, систем сповіщень, баз даних і т.п. – задача, об’єм якої буде неможливо зробити наодинці. Враховуючи, що робота виконувалась автором наодинці, було вирішено результатом програмної реалізації мати мінімально життєздатну версію (англ. MVP) продукту, яка демонструвала б результати проведеного дослідження.

РОЗДІЛ ІІІ. ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ

3.1. Опис вибраних технологій

Для реалізації поставленої мети, з урахуванням функціональних та нефункціональних вимог, було вибрано технології, які лягли в основу програмної системи. В якості мови програмування вибрано мову Python. Згідно опитування сайту StackOverflow, Python посідає 3 за популярністю місце серед усіх мов програмування після JavaScript та HTML/CSS: протягом останніх місяців його використовувало 49% респондентів [21]. Python – інтерпретована мова програмування з динамічною строгою типізацією, що використовується передусім в серверній (backend) розробці [22]. В індустрії даних ця мова є найпопулярнішою і стандартом де-факто галузі, що дозволить користувачам за потреби модифікувати систему.

Для реалізації серверної частини була обрана бібліотека FastAPI: це асинхронний веб-фреймворк, що почав розроблятись в 2018 році, і завоювала популярність серед спільноти. Веб-фреймворком називають бібліотеку, яка дозволяє побудувати на своїй основі веб-застосунок. Фреймворк відрізняється від своїх популярних у світі Python аналогів Django та Flask тим, що він повністю асинхронний та базується на typehints – оголошеннях про типи в Python [23]. Підтримка typehints реалізована через бібліотеку Pydantic, що є найпоширенішою бібліотекою для валідації даних в Python. Асинхронність FastAPI дозволить підвищити продуктивність роботи веб-серверу за рахунок зменшення процесорного часу, що в синхронному режимі витрачається на затратні I/O операції.

FastAPI часто називають мікрофреймворком (micro framework) по аналогії з Flask через те, що він надає «з коробки» небагато можливостей, а весь додатковий функціонал потрібно реалізовувати окремо через підключення сторонніх бібліотек або власноруч написану логіку. На противагу цьому, такі великі фреймворки як, наприклад, Django надають дуже широкий функціонал:

так, Django має вбудовану ORM (систему для трансляції роботи з базою на вбудовані засоби мови програмування), бібліотеку для міграцій бази даних (абстракція, яка дозволяє БД змінювати конфігурацію таблиць за допомогою наборів команд, які переводять БД з попередньої конфігурації в наступну, і навпаки), та рушій серверного рендерингу (форматування та підставлення даних в HTML шаблони сторінок). FastAPI немає вбудованих інструментів для цього, тому разом з фреймворком були використані такі бібліотеки: Psycopg3 в якості рушія для спілкування з БД Postgres, SQLAlchemy як ORM для СКБД Postgres, Jinja2 в якості рушія серверного рендерингу, та Alembic для міграцій БД.

Для маніпуляції даними обрано бібліотеку Pandas, вона є найбільш популярною такою бібліотекою на Python, і також є стандартом де-факто у галузі даних, що також дозволить легко робити зміни в системі іншим людям. Дуже багато людей критикують Pandas за неінтуїтивність і нелогічність API, відсутність розподіленості і порівняно низьку швидкість роботи, але для роботи з помірним розміром даних цього цілком достатньо, тому вибір був зроблений на користь цієї бібліотеки, а не її конкурентів на кшталт Polars, які ще не набули такої популярності.

Правильним є відокремити роботу веб-серверу від роботи з даними (запуску перевірок), тому ці компоненти мають якимось чином спілкуватись між собою. Для цього обрано Celery – чергу задач на Python, яка дозволяє легко передавати задачі на виконання іншим компонентам і зручно слідкувати за їхнім виконанням, станом, успішністю і забирати результат. Оскільки перевірки мають виконуватись за розкладом, потрібен компонент, який би це робив: для цього обрано Celery-Redbeat. В якості черги повідомлень обрано Redis як надшвидку базу даних, що зберігається в оперативній пам’яті: між компонентами системи будуть передаватись повідомлення малого розміру, тому зберігання даних в оперативній пам’яті є допустимим. Для адміністрування Celery було обрано інструмент Flower: він надає можливість через веб-інтерфейс дивитись за станом черг, задач, їхнім виконанням, та можливими помилками.

Замість застосування планувальника Redbeat для Celery також розглядався вибір іншого планувальника або оркестратора задач, наприклад, Airflow або Dagster, але через складність цих інструментів, трудоємність інтеграції та малу утилізацію їхнього потенціалу для поставленої задачі, ці опції були відзначені як надлишкові та відкинуті.

Для демонстрації алгоритмів визначення аномалій було обрано наступні бібліотеки: Statsmodels для виявлення за допомогою STL декомпозиції, та Prophet як бібліотеку аналізу часових рядів за допомогою неконтрольованого навчання.

Для відправки електронних листів було вирішено обрати Gmail як найбільш популярну систему електронної пошти. Аутентифікація в Gmail проводиться через протокол OAuth 2.0. Відправка повідомлень в Slack відбувається за допомогою Slack API. Запити до обидвох систем відправляються через бібліотеку httpx.

Як спосіб реалізації веб-частини системи було вирішено вибрати стратегію змішаного рендерингу: серверний рендеринг відбувається за допомогою Jinja2, а клієнтський рендеринг за допомогою JavaScript бібліотек. Для комунікації між сервером і клієнтом при клієнтському рендерингу обрано концепцію RESTful API. Стисло кажучи, ця концепція полягає в атомарності і незалежності один від одного запитів між клієнтом та сервером: головним є те, що запит до сервера має мати всю необхідну інформацію, щоб його опрацювати, тобто, запит не має контексту.

На стороні клієнту реалізація веб-сторінок була зроблена за допомогою мови розмітки HTML 5 та мови стилів CSS 3, з використанням JavaScript для динамічності сторінок. Інтерфейс зроблений на базі бібліотеки компонентів Tabler: вона постачається у вигляді CSS та JS файлів, які підключаються до потрібного веб-сайту, і надає широкий набір заготовлених компонентів, зовнішній вигляд яких зроблений в одному стилі. Tabler дуже популярний через те, що побудований на базі Bootstrap – CSS-бібліотеки стилів, яка повсюдно використовується в веб-сторінках, оскільки дозволяє максимально швидко розробляти красиві веб-сторінки, адаптовані до різних пристроїв і форматів

екранів. Для клієнтського рендерингу було обрано бібліотеку Alpine.js, яка надає можливість декларативно описувати залежності компонентів, і при зміні цих залежностей компоненти будуть перебудовуватись з новими даними, що позбавляє потреби робити оновлення вмісту і вигляду компонентів вручну.

Керування кодовою базою відбувається через систему контролю версій Git, а в якості віддаленого сховища для Git репозиторію обрано GitHub як стандарт де-факто в індустрії. Опис до Git комітів зроблений за допомогою концепції Conventional Commits.

3.2. Опис програмної реалізації

На виконання поставлених вимог на базі обраних технологій була розроблена програмна система автоматизованого виявлення аномалій. Архітектурним патерном виконання системи є мікросервісна архітектура. Система складається з трьох компонентів:

- *Веб-сервер (web server)*: відповідає за роботу веб-інтерфейсу системи;
- *Планувальник (scheduler)*: планує виконання перевірок аномалій за розкладом;
- *Виконавець (worker)*: виконує задачі, що приходять від планувальника та веб-серверу.

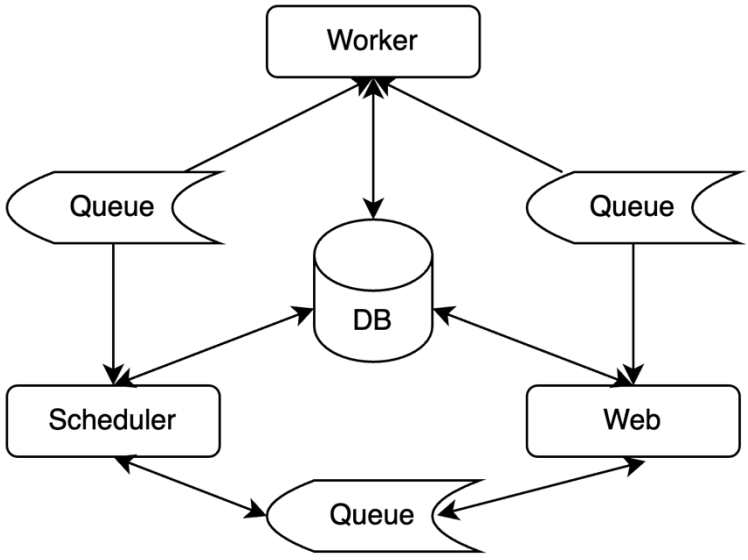

Рис. 3.2.1. Схема взаємодії компонентів системи

Компоненти взаємодіють один з одним за допомогою черги задач (task queue): веб-сервер та планувальник надсилають в цю чергу задачі, які потрібно виконати, виконавець виконує їх і сигналізує про виконання результатом, який надсилає в цю ж чергу. Черга відповідає лише за комунікацію про завдання між компонентами системи; за зберігання даних відповідає БД, яка доступна всім компонентам.

Система складається з п’яти функціональних модулів: конфігурації виявлення, запуски перевірок, підключення до БД, контакти та повідомлення. Для того, щоб налаштувати систему, користувачу потрібно зробити наступне: додати підключення до бази даних, додати контакт для надсилання сповіщень, і створити конфігурацію виявлення.

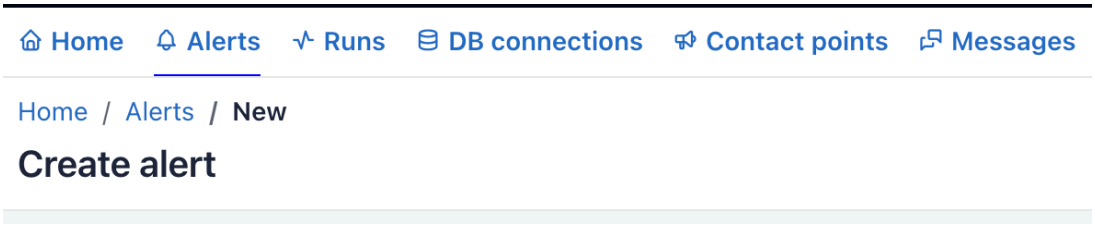

Рис. 3.2.2. Відображення модулів у інтерфейсі

Найпершим слід відзначити головний елемент системи: менеджер налаштування конфігурацій виявлення. Ключовою особливістю системи є її інтерактивність: користувач має в інтерактивному режимі налаштовувати виявлення аномалій. Цю особливість було реалізовано в менеджері налаштування.

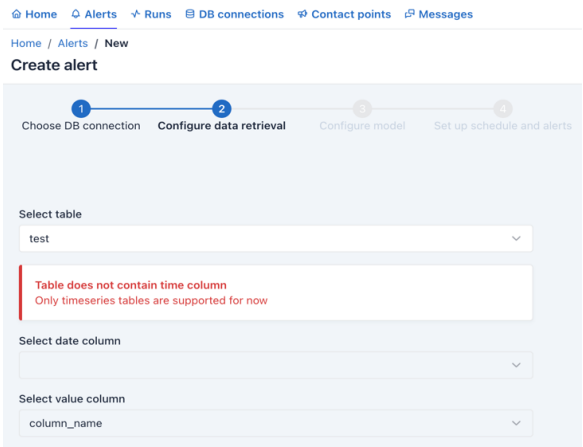

Рис. 3.2.3. Сповіщення про неможливість вибору таблиці в менеджері налаштування відстеження

Налаштування складається з чотирьох кроків, які користувач проходить послідовно:

1. *Вибір підключення до БД*. Користувач обирає попередньо додане ним підключення до бази даних;
2. *Вибір даних для відстеження*. Менеджер сканує обрану базу даних і пропонує користувачу обрати потрібну таблицю та колонки, з яких система братиме дані. Система сповіщає користувача, якщо певну таблицю чи колонку неможливо обрати;
3. *Вибір і налаштування моделі*. На цьому етапі користувач має можливість вибрати модель, налаштувати часові інтервали і параметри моделі та одразу перевірити вибрану конфігурацію. Таким чином можна швидко налаштувати конфігурацію так, як це потрібно користувачу. Користувач бачить, що визначає модель і про що вона сповіщає, на графіку;
4. *Налаштування розкладу і сповіщень*. Користувач вибирає розклад запуску перевірки за допомогою Cron Expression (нотація для запису періодичності), та вибирає канал, в який потрібно буде надіслати сповіщення.

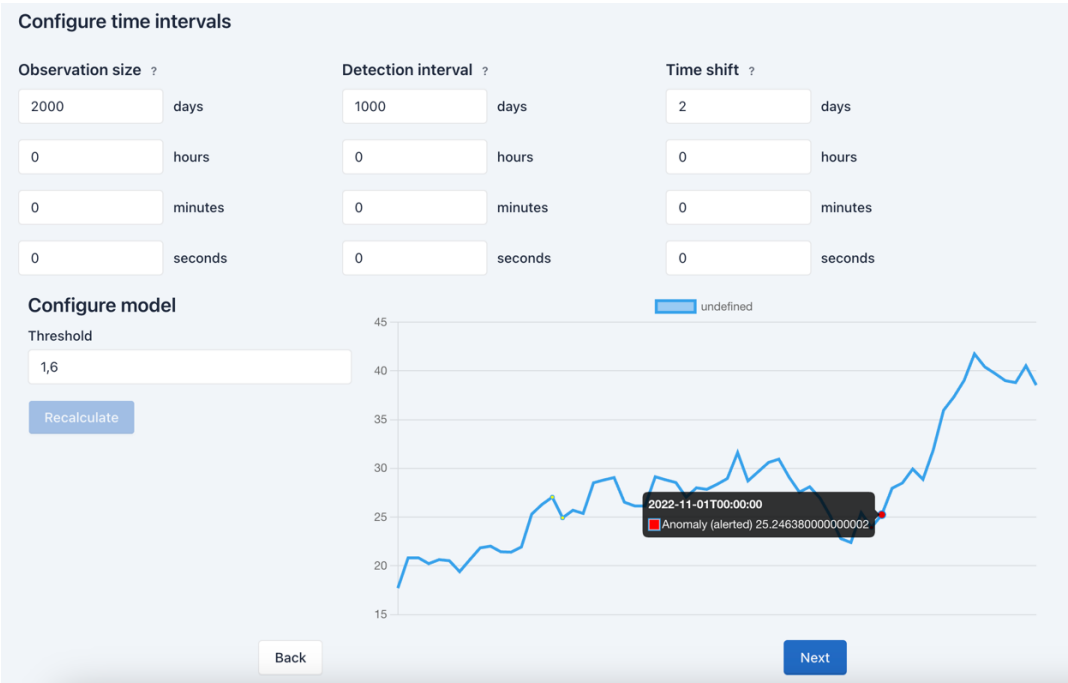

Рис. 3.2.4. Інтерактивне налаштування моделі в менеджері

Після проходження всіх цих кроків створюється конфігурація запуску визначення аномалій, яка зберігається в базі даних і додається до розкладу планувальника. При цьому перезавантажень планувальника для додавання чи видалення конфігурації не потребується.

Планувальник працює постійно і порівнює поточний час з розкладом виконання моделі, за збігу – планує виконання визначень, конфігурує виконання і надсилає в чергу повідомлення з задачею на виявлення. Ці повідомлення отримує виконавець та робить саму перевірку. Таким чином, за допомогою черги задач процес обробки даних, який є дуже вартісним з точки зору ресурсів, відділений від інших функцій системи, і можна окремо керувати його ресурсами.

Виконання перевірки складається з чотирьох етапів: вибір даних з бази, виконання обробки даних моделлю, запис результатів до БД, і надсилання сповіщення (за потреби). Якщо при виконанні перевірки відбулась програмна помилка, користувач буде сповіщений про це в обраному каналі комунікації, також це буде відображено на сторінці запуску. Якщо при надсиланні сповіщення виникла проблема, через яку це неможливо зробити, користувач побачить це тільки на сторінці запуску.

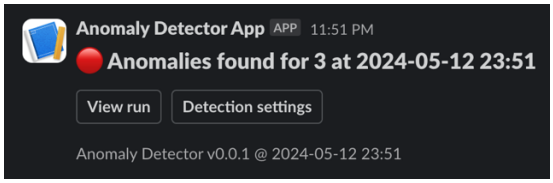

Рис. 3.2.5. Сповіщення про аномалії, надіслане в Slack

Для надсилання сповіщень про помилку при виконанні перевірки або про знайдені аномалії в системі підтримується два механізми: Slack та Gmail.

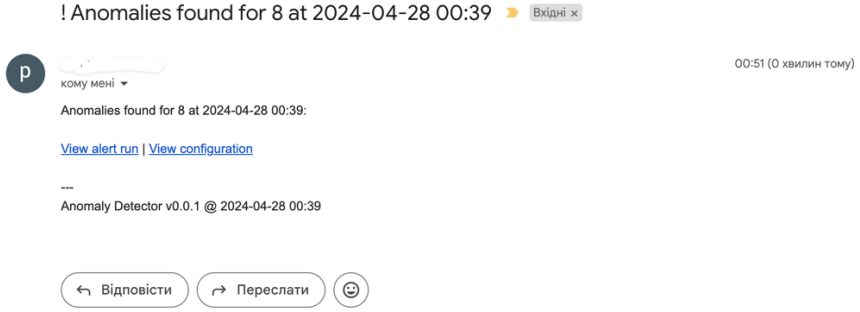

Рис. 3.2.6. Сповіщення про аномалії, надіслане з Gmail

Для запуску перевірок розроблено наступну систему часових інтервалів: інтервал спостереження, інтервал виявлення та інтервал сповіщення, які обраховуються з врахуванням зсуву. Інтервал спостереження – це проміжок часу, який використовується для тренування моделі аномалій. Його початком є різниця моменту запуску аномалій, зсуву в часі та тривалість інтервалу спостереження, його кінцем – різниця моменту запуску аномалій та зсуву в часі. Для знайдених на цьому проміжку аномалій сповіщення не надсилаються. Інтервал виявлення – це проміжок часу, який використовується для обробки моделлю аномалій. Його початком є різниця моменту запуску аномалій, зсуву в часі та тривалість інтервалу виявлення, його кінцем – різниця моменту запуску аномалій та зсуву в часі. Для знайдених в цьому проміжку аномалій сповіщення надсилаються (навіть з урахуванням того, що точки в цьому інтервалі також належать інтервалу спостереження). Зсув в часі – це різниця між часом виконання перевірки та кінцем інтервалів спостереження і виявлення. Демонстрація часових інтервалів зображена на *Рис. 3.2._*. Аномалії, які лежать поза межами інтервалу виявлення, називаються *аномаліями без сповіщень*, а ті, що лежать в межах інтервалу – *аномаліями зі сповіщеннями*.

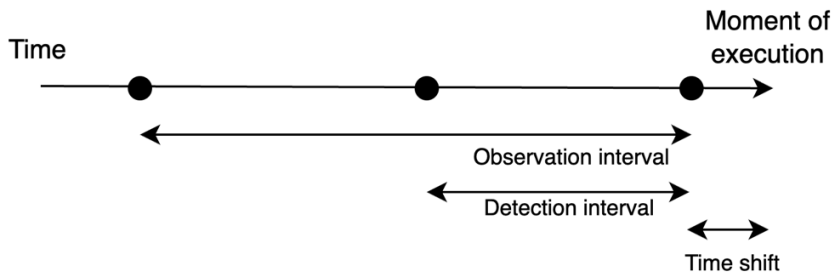

Рис. 3.2.7. Схема часових інтервалів

На сторінці конфігурації можна побачити статистику виконання перевірок для цієї конфігурації: відсоток виконаних перевірок зі сповіщеннями, без сповіщень та без аномалій. Також можна побачити статус виконання 30 останніх перевірок, вибрані налаштування перевірки, вибрані підключення до БД та канал сповіщення, і внизу сторінки доданий список останніх запусків для моделі

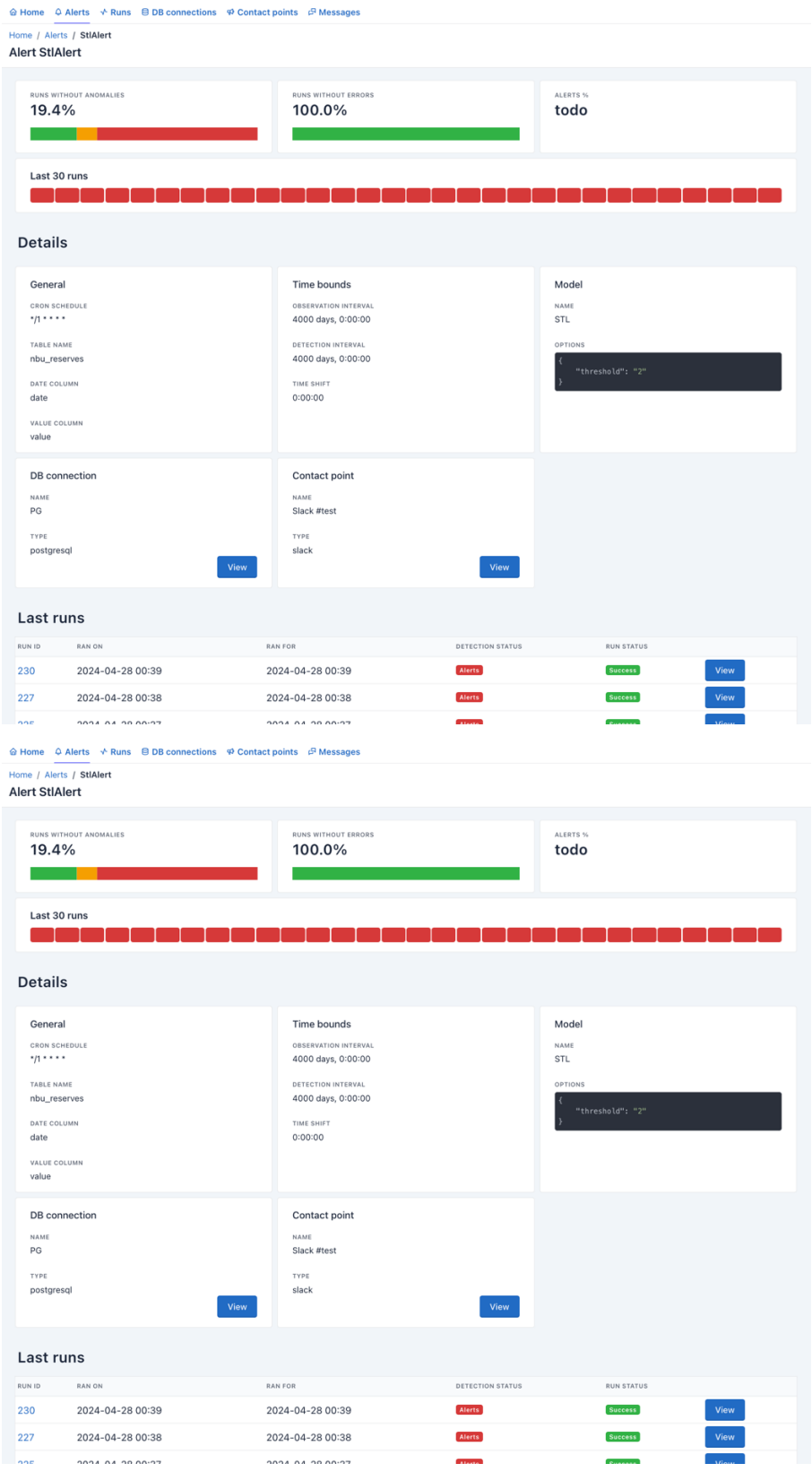

Рис. 3.2.8. Сторінка конфігурації

На сторінці запуску можна побачити статуси запуску, графік, побудований для даних, список аномалій, які викликали сповіщення, список аномалій без

сповіщень, конфігурацію запуску перевірки, розміри часових інтервалів та інші параметри запуску.

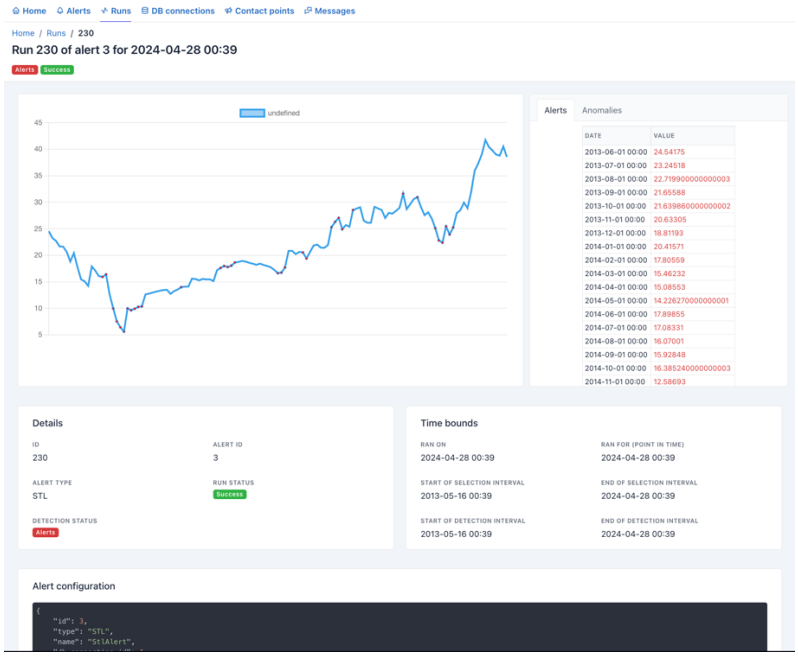

Рис. 3.2.9. Сторінка запуску

Модулі підключень до БД та контакти схожі між собою: обидва модулі містять сторінки для додавання і перегляду таких підключень. Кожне підключення користувачу обов’язково треба перевірити перед додаванням за допомогою кнопки «Перевірити», некоректні підключення неможливо додати в систему.

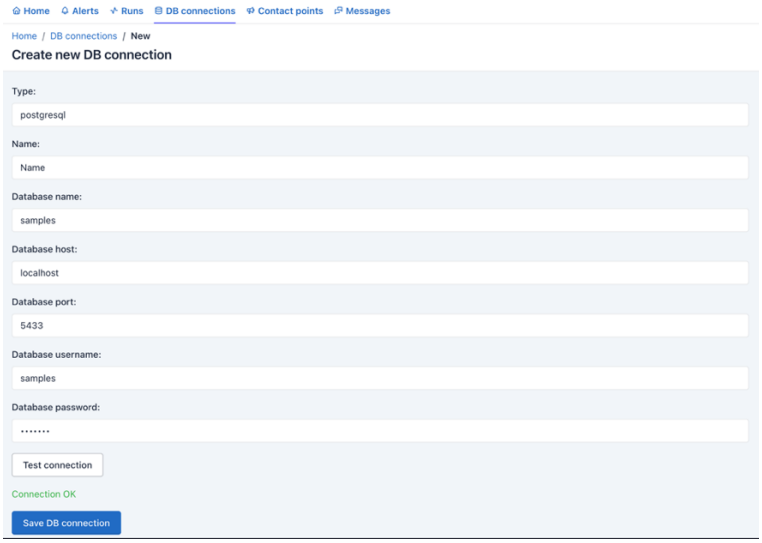

Рис. 3.2.10. Створення підключення до БД

Важливо зазначити, що ключовим моментом реалізації цих модулів є архітектурно і інтерфейсно передбачена можливість додавання нових механізмів (провайдерів). В коді системи створені абстракції, для додавання нових провайдерів потрібно унаслідуватись від цих абстракцій в новому класі, і перевизначити методи абстракції, додавши логіку, що стосується саме цієї системи.

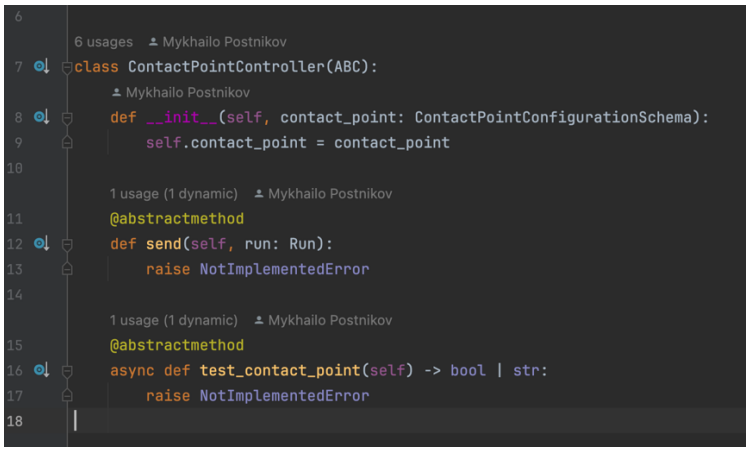

Рис. 3.2.11. Абстракція провайдеру контактів

При цьому інтерфейс користувача автоматично підлаштовується під кожного провайдера: для цього достатньо лише визначити нову модель даних, і системою буде створений інтерфейс, що матиме всі потрібні поля без потреби в додаткових налаштуваннях.

Подібний механізм створений і для моделей визначення аномалій: для додавання нових достатньо унаслідуватись від абстракцій і перевизначити логіку, що відмінна саме для цієї моделі, так само тут зроблене автоматичне генерування налаштувань. Підтримка різних моделей і провайдерів, легкість додавання нових імплементацій цих компонентів, і тим самим висока адаптивність є ключовою особливістю системи, що особливо важлива в індустрії даних, де використовується багато різних технологій, які часто змінюються. Це дозволить використовувати систему в різних умовах.

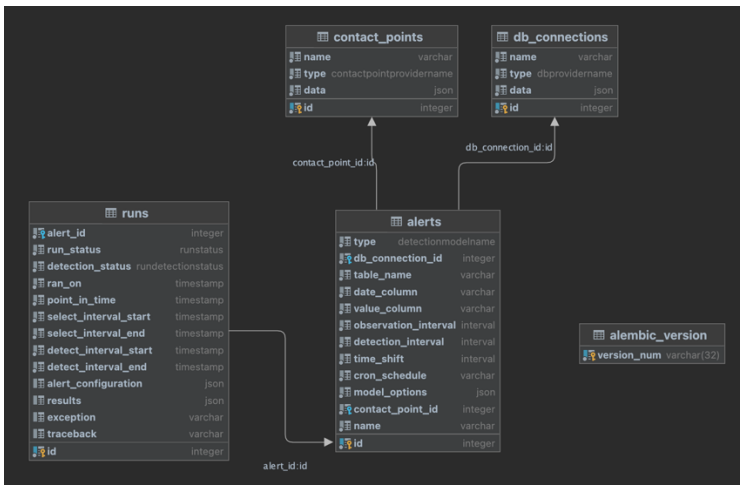

Рис. 3.2.12. Схема бази даних системи

Розроблена система повністю задовольняє поставлені раніше вимоги: легкість у налаштуванні забезпечується інтерфейсом користувача і інтерактивним процесом, гнучкість та легкість кастомізації – обраними технологіями та архітектурними абстракціями, надійність – розділенням компонентів через чергу задач, функціональні вимоги – розробленими складовими модулями. Система виконує поставлену задачу автоматизованого виявлення аномалій в бізнес-даних і відповідає сучасним стандартам в галузі даних.

3.3. Перспективи розвитку системи

Одним із обмежень на етапі формулювання вимог була мінімальна достатня функціональність продукту (MVP), оскільки реалізація додаткового функціоналу потребувала б більше часу і ресурсів. Саме тому система має великий потенціал до розвитку і може бути допрацьована в багатьох моментах.

Передусім, в систему може бути додана підтримка інших баз та сховищ даних (наприклад, AWS Athena, Amazon Redshift), підтримка API для отримання даних, та додана інтеграція з іншими системами комунікації (наприклад, Microsoft Teams). Це забезпечить можливість використання системи в інших умовах та компаніях, оскільки в галузі одночасно використовується велика кількість різних технологій. Велику практичну користь принесе і додавання

інших моделей виявлення аномалій, наприклад, тих, що базуються на машинному навчанні.

Одним з потрібних доопрацювань є додавання вимірів для даних (data dimensions) і аналіз аномалій всередині цих вимірів, як індивідуально, так і в комбінації (наприклад, відслідковуються продажі певного продукту для 10 різних платформ, в цьому випадку ці платформи будуть вимірами даних, і можна буде аналізувати аномалії для кожної з таких платформ).

Також в систему можна буде додати попередню обробку вхідних даних, наприклад, фільтрацію, аґреґацію, або написання власного SQL-запиту для вибору даних з бази. Для надсилання сповіщень можливо додати шаблони, за допомогою яких задаватиметься текст таких сповіщень, що підвищить їхню інформативність і цінність для певного користувача.

Функціональність статистики в системі може бути розширена для перегляду перевірок і їхніх результатів, профілювання даних на основі різних перевірок, або й таким чином, який би дозволив визначати стан даних на основі всіх перевірок.

Система має широкий простір для доопрацювань, але при їх впровадженні має бути дотримана ціль, логіка роботи і виконання вимог до системи.

ВИСНОВКИ

У роботі було досліджене поняття аномалії і дана власна дефініція цьому поняттю. Сформульовано проблематику аналізу даних і актуальне місце виявленню аномалій в ній і окреслене застосування цього підходу в сфері обробки даних загалом. Був проведений огляд і аналіз наукової літератури за темою, що дозволив систематизувати таксономію аномалій і технік їхнього визначення. На основі проаналізованої літератури були наведені основні види і класифікації для аномалій і технік визначення аномалій, що дає уявлення про практичну цінність виявлення аномалій в обробці даних.

Було дане визначення поняттю бізнес-даних, розглянуто застосування цього підходу в цій галузі і перелічено проблеми, з якими сьогодні стикається галузь при аналізі даних. На основі проведеного дослідження підходу виявлення аномалій і поставленої проблеми було зроблене припущення про можливість використання автоматизованої програмної системи для вирішення знайдених проблем аналізу даних. Метою роботи стала побудова автоматизованої системи виявлення аномалій.

Для виконання поставленої мети було проведене дослідження існуючих інструментів, що близькі до тематики роботи. Було знайдено подібні рішення і проведений їхній аналіз з точки зору відповідності їх потребам бізнесу і якості вирішення цих проблем. Аналіз показав, що існуючі системи неактуальні, не підтримуються, і не виконують задачі, потрібні для розв’язання проблеми. Таким чином була підтверджена потреба в системі, що вирішувала б розглянуту у роботі проблему.

На основі зробленого дослідження підходу виявлення аномалій, знайдених при аналізі рішень переваг та недоліків існуючих систем, та власного досвіду автора в галузі були розроблені вимоги, яким має відповідати розроблена система. На виконання цих вимог було обрано архітектуру системи, спроєктовано її архітектуру, та проаналізовано і обрано технології для її програмної імплементації.

Практичним результатом роботи є автоматизована система виявлення аномалій, що була розроблена відповідно до поставлених вимог і з використанням обраних технологій. Система складається з кількох програмних модулів, що разом забезпечують виконання задач аналізу бізнес-даних. Розроблена система складається з програмних модулів, має високу адаптивність, легкість до модифікації, і зручність у використанні. Легкість налаштування забезпечується інтерактивним веб-інтерфейсом, а надійність використання – розмежованою архітектурою з обмеженою відповідальністю компонентів.

Цінністю отриманого результату є його відповідність вимогам бізнесу і сучасним трендам в галузі обробки даних, пристосованість до вирішення проблем, що актуальні сьогодні в галузі, надійність і адаптивність функціонування системи.

Таким чином, в результаті роботи вдалось підтвердити поставлену раніше гіпотезу про можливість використання програмної системи для вирішення проблеми аналітики бізнес-даних. Було дано опис розробленій системі, а також розглянуто можливі варіанти її доопрацювання. Мета дослідження і поставлені завдання були повністю виконані.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Arthur C. Tech giants may be huge, but nothing matches big data. *The Guardian*. URL: https://www.theguardian.com/technology/2013/aug/23/tech-giants-data (дата звернення: 15.05.2024).
2. 50+ Incredible Big Data Statistics for 2024: Facts, Market Size & Industry Growth - Big Data Analytics News. *Big Data Analytics News*. URL: https://bigdataanalyticsnews.com/big-data-statistics/ (дата звернення: 15.05.2024).
3. Laborde R. The Three V's of Big Data: Volume, Velocity, and Variety. *Oracle Life Sciences Blog*. URL: https://blogs.oracle.com/life-sciences/post/the-three-vx27s-of-big-data-volume-velocity-and-variety (дата звернення: 15.05.2024).
4. Cai L., Zhu Y. The Challenges of Data Quality and Data Quality Assessment in the Big Data Era. *Data Science Journal*. 2015. Т. 14. С. 2. URL: https://doi.org/10.5334/dsj-2015-002 (дата звернення: 15.05.2024).
5. Edgeworth F. Y. XLI. On discordant observations. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*. 1887. Т. 23, № 143. С. 364–375. URL: https://doi.org/10.1080/14786448708628471 (дата звернення: 15.05.2024).
6. Grubbs F. E. Procedures for Detecting Outlying Observations in Samples. *Technometrics*. 1969. Т. 11, № 1. С. 1–21. URL: https://doi.org/10.1080/00401706.1969.10490657 (дата звернення: 15.05.2024).
7. Hawkins D. M. Identification of Outliers. Dordrecht : Springer Netherlands, 1980. URL: https://doi.org/10.1007/978-94-015-3994-4 (дата звернення: 15.05.2024).
8. Chandola V., Banerjee A., Kumar V. Anomaly Detection: A Survey. *ACM Computing Surveys*. 2009. Т. 41, № 3. С. 1–58. URL: https://doi.org/10.1145/1541880.1541882 (дата звернення: 15.05.2024).

9. Ben-Gal I. Outlier Detection. *Data Mining and Knowledge Discovery Handbook*. New York. С. 131–146. URL: https://doi.org/10.1007/0-387-25465-x_7 (дата звернення: 15.05.2024).
10. Zhang J. Advancements of Outlier Detection: A Survey. *ICST Transactions on Scalable Information Systems*. 2013. Т. 13, № 1. С. e2. URL: https://doi.org/10.4108/trans.sis.2013.01-03.e2 (дата звернення: 15.05.2024).
11. Hodge V., Austin J. A Survey of Outlier Detection Methodologies. *Artificial Intelligence Review*. 2004. Т. 22, № 2. С. 85–126. URL: https://doi.org/10.1023/b:aire.0000045502.10941.a9 (дата звернення: 15.05.2024).
12. Data-Driven Business Models: A Blueprint for Innovation / J. Brownlow та ін. 2015. URL: https://doi.org/10.13140/RG.2.1.2233.2320 (дата звернення: 15.05.2024).
13. Ravishankar D. T. N. Application of Time Series Analysis for Better Decision Making in Business. *TechnoareteTransactions on Intelligent Data Mining and Knowledge Discovery*. 2022. Т. 2, № 4. URL: https://doi.org/10.36647/ttidmkd/02.04.a005 (дата звернення: 15.05.2024).
14. GitHub - chaos-genius/chaos_genius: ML powered analytics engine for outlier detection and root cause analysis. *GitHub*. URL: https://github.com/chaos-genius/chaos_genius (дата звернення: 15.05.2024).
15. Introduction | Chaos Genius Documentation. URL: https://docs.chaosgenius.io/docs/introduction (дата звернення: 15.05.2024).
16. GitHub - cuebook/CueObserve: Timeseries Anomaly detection and Root Cause Analysis on data in SQL data warehouses and databases. *GitHub*. URL: https://github.com/cuebook/CueObserve (дата звернення: 15.05.2024).
17. Overview | CueObserve. *Overview | CueObserve*. URL: https://cueobserve.cuebook.ai (дата звернення: 15.05.2024).
18. GitHub - elementary-data/elementary: The dbt-native data observability solution for data & analytics engineers. Monitor your data pipelines in minutes.

Available as self-hosted or cloud service with premium features. *GitHub*. URL: https://github.com/elementary-data/elementary (дата звернення: 15.05.2024).

19. Welcome to Elementary - Elementary. *Welcome to Elementary - Elementary*. URL: https://docs.elementary-data.com/introduction (дата звернення: 15.05.2024).

20. Kennedy N. Why You Should Probably be Using Elementary with your dbt Project. *Noah Kennedy*. URL: https://noahlk.medium.com/elementary-on-dbt-an-overview-5dd038d0d3b8 (дата звернення: 15.05.2024).

21. Stack Overflow. Stack Overflow Developer Survey 2023. *Stack Overflow*. URL: https://survey.stackoverflow.co/2023/ (дата звернення: 15.05.2024).

22. 3.12.3 Documentation. *Python Documentation*. URL: https://docs.python.org/3/ (дата звернення: 15.05.2024).

23. Features - FastAPI. *FastAPI*. URL: https://fastapi.tiangolo.com/features/ (дата звернення: 15.05.2024).