МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
УМАНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ САДІВНИЦТВА
Факультет економіки і підприємництва
Кафедра інформаційних технологій

Допущено до захисту
"_____" _____ 2023 р.
Зав. кафедри, к.т.н., доцент
_____ **Ліщук Р.І.**

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ АНАЛІЗУ ОСОБИСТИХ ВИТРАТ ТА СПОЖИВЧОГО ПОПИТУ

Виконав – студент 2 курсу, групи 21м-кн
освітньої програми "Комп’ютерні науки"
спеціальності 122 Комп’ютерні науки
Міюсов Дмитро Юрійович

Науковий керівник – к.т.н., доцент
Ліщук Роман Ігорович

Умань – 2023

Зміст

ВСТУП 3

РОЗДІЛ 1 АНАЛІЗ ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАДАЧІ 5

1.1. Аналіз проблеми..... **Помилка! Закладку не визначено.**

1.2. Підходи до локалізації об’єктів інтересу на зображенні ... **Помилка! Закладку не визначено.**

1.3. Постановка задачі 7

РОЗДІЛ 2 ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ КЛАСИФІКАЦІЙНОГО АНАЛІЗУ ДЕРМОСКОПІЧНИХ ЗОБРАЖЕНЬ 16

2.1. Архітектура екстрактора ознак..... 16

2.2. Архітектура детектора..... 19

2.3. Метрики оцінки ефективності класифікатора ..... **Помилка! Закладку не визначено.**

РОЗДІЛ 3 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА АНАЛІЗ РЕЗУЛЬТАТІВ..... 38

3.1. Формування вхідного математичного опису**Помилка! Закладку не визначено.**

3.2. Короткий опис програмної реалізації системи**Помилка! Закладку не визначено.**

3.3. Аналіз результатів моделювання**Помилка! Закладку не визначено.**

ВИСНОВКИ..... 53

СПИСОК ЛІТЕРАТУРИ..... 54

ДОДАТКИ..... 58

ВСТУП

"Якщо у вас є час вивчити лише одну тему з економічної теорії, то нехай це буде питання попиту та пропозиції", таку думку висловив Флінн Шон Масакі у своїй книзі "Економіка для чайників"[1], не можна з ним не погодитися, адже ці два поняття можливо не єдині, але точно головні рушійні сили економіки. Саме попит і пропозиція формують ринок товарів та послуг безпосередньо впливаючи на асортимент і ціни.

Попит – це насамперед зацікавленість споживачів у товарі, а також фінансова спроможність придбати цей товар. Вивчення і прогнозування споживчого попиту є необхідною умовою для прийняття ефективних маркетингових рішень, з метою отримання максимуму з існуючих обмежених ресурсів [2].

Дана робота спрямована на створення зручного веб-сервісу, який дозволить отримувати актуальні аналітичні дані, що сформовані на основі персональних витрат споживачів.

Зацікавленим сторонам, якими є підприємства в сфері виробництва, торгівлі і послуг, а також домогосподарства і окремі члени цих домогосподарств, буде надаватися візуалізований аналіз витрат, що дозволить відслідковувати зміни в споживчому попиті і виявляти зацікавленість покупців до того чи іншого товару, порівнювати між собою попит на взаємозамінні товари, аналізувати динаміку цін на товар в продовж певного періоду, визначати сезонність товару тощо.

Доступність реальних даних про персональні витрати представляє велику цінність з точки зору економічного і маркетингового планування виробників і підприємців. Розроблюваний веб-сервіс задовольняє вказані потреби, надаючи швидкий і зручний доступ до таких даних, що і є метою даної роботи.

РОЗДІЛ 1 АНАЛІЗ ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАДАЧІ

1.1. Актуальність проблеми

Метою даної роботи є створення веб-сервісу, який буде акумулювати дані користувачів мобільних додатків "персональних витрат" і на основі цих даних надавати аналітику видатків населення.

Спектр застосування таких аналітичних даних дуже широкий, але насамперед зацікавленими сторонами є підприємства зі сфери торгівлі та надання послуг і виробники, агреговані дані про покупки допоможуть цим компаніям у прийнятті бізнес рішень і у плануванні об’ємів виробництва і продажу [3]. Дані про витрати використовуються для прогнозування попиту [3–6], аналізу ефективності роботи закладів харчування [7], [8], для оцінки масовості використання антибіотиків [9], для виявлення спалахів інфекційних захворювань [10], як інструмент оцінки економічного рівня країни [11], тощо.

Обробка і аналіз даних про витрати є складним завданням і потребує розробки системи, яка буде ефективно аналізувати і візуалізовувати дані. Існують алгоритми і моделі, які спеціально адаптовані для обробки інформації про покупки. Прикладом таких моделей є модель Басса/Нортона [8], паралельний алгоритм правил асоціації [12], Gradient Boos Algorithm [5] та інші. Також існують вже готові системи, що реалізують і обробку, і аналітику, і візуалізацію даних [13], [14]. Всі ці методи та системи аналізу даних можуть бути використані в розроблюваній системі, та не можуть в повній мірі вважатися аналогами даної системи з огляду на унікальність вхідних даних.

Актуальність теми аналізу витрат обумовлена впливом споживчого попиту на формування ринку товарів та послуг. Від того, які групи товарів і послуг користуються попитом у споживачів, в значній мірі залежить структура виробництва, торгівлі і сфери послуг. А достовірні дані про витрати населення як найкраще відображають реальний споживчий попит на ті чи інші товари чи послуги, а також дають можливість аналізувати суми витрат, які споживачі готові на це витрачати [14].

Сьогодні, коли потенційний покупець може придбати необхідне в магазині поруч з будинком, або замовити той же товар в пару кліків із Китаю чи Сполучених Штатів, конкуренція стала надзвичайно високою. Щоб мати кращі умови купівлі – продажу, виробництва чи надання послуг, учасники ринку повинні чітко розуміти на скільки їх послуга чи товар будуть цікаві потенційному покупцю. Тому вивчення споживчого попиту, це те з чого потрібно починати відкриваючи новий бізнес і те, чим мають постійно займатися вже досвідчені бізнесмени [15]. Попит формується не сам по собі, а під впливом зовнішніх факторів, до яких можна віднести:

- Ціна на товар
- Кількість покупців
- Місткість ринку
- Доходи споживачів
- Ціна на супутні товари
- Очікування споживачів
- Кліматичні умови
- Надзвичайні ситуації [16].

Яскравим прикладом того, як під впливом зовнішніх факторів змінюється попит, є стрімке зростання пошукового запиту в Google "павербанк купити" після 10 жовтня (рис. 1.1). З чого можна зробити висновок, що велика кількість факторів, що мають вплив на попит, роблять його нестабільним і мінливим, створюючи тим самим необхідність зацікавлених сторін постійно відслідковувати тенденції і зміни на ринку товарів та послуг.

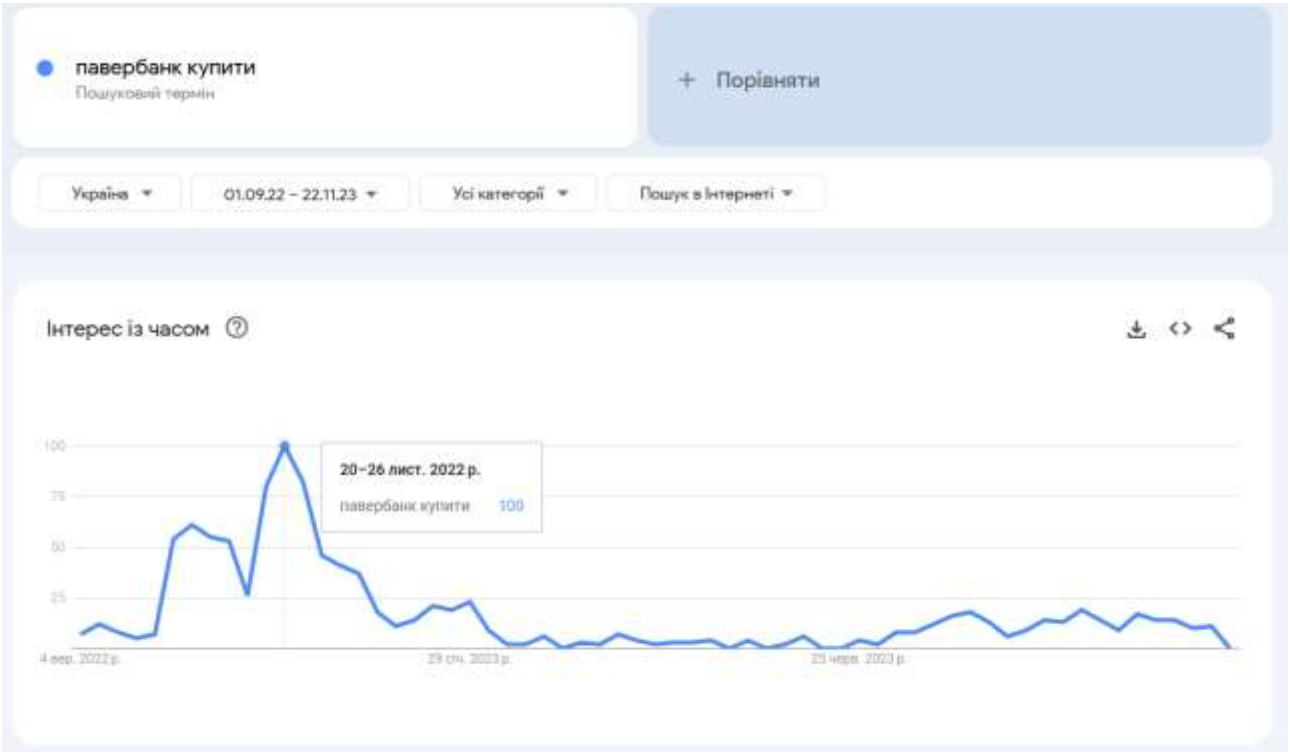


Рисунок 1.1 – Динаміка популярності запиту "павербанк купити"

1.2. Огляд існуючих рішень

E- Receipt

26 вересня 2018р. Прес-служба Державної фіскальної служби України повідомила про запуск інформаційної системи E-Receipt (рисунок 1.2) в рамках реалізації експериментального проекту, розпочатого Мінфіном відповідно до постанови Кабміну щодо реєстрації та експлуатації новітніх моделей програмних та/або програмно-технічних комплексів, призначених для реєстрації розрахункових операцій. Крім того, ДФС реалізовано і електронний сервіс, який дозволяє покупцеві товарів (послуг) здійснювати пошук та перегляд фіскального касового чеку, що формується як традиційними реєстраторами розрахункових операцій, так і електронний фіскальний чек, створений РРО.

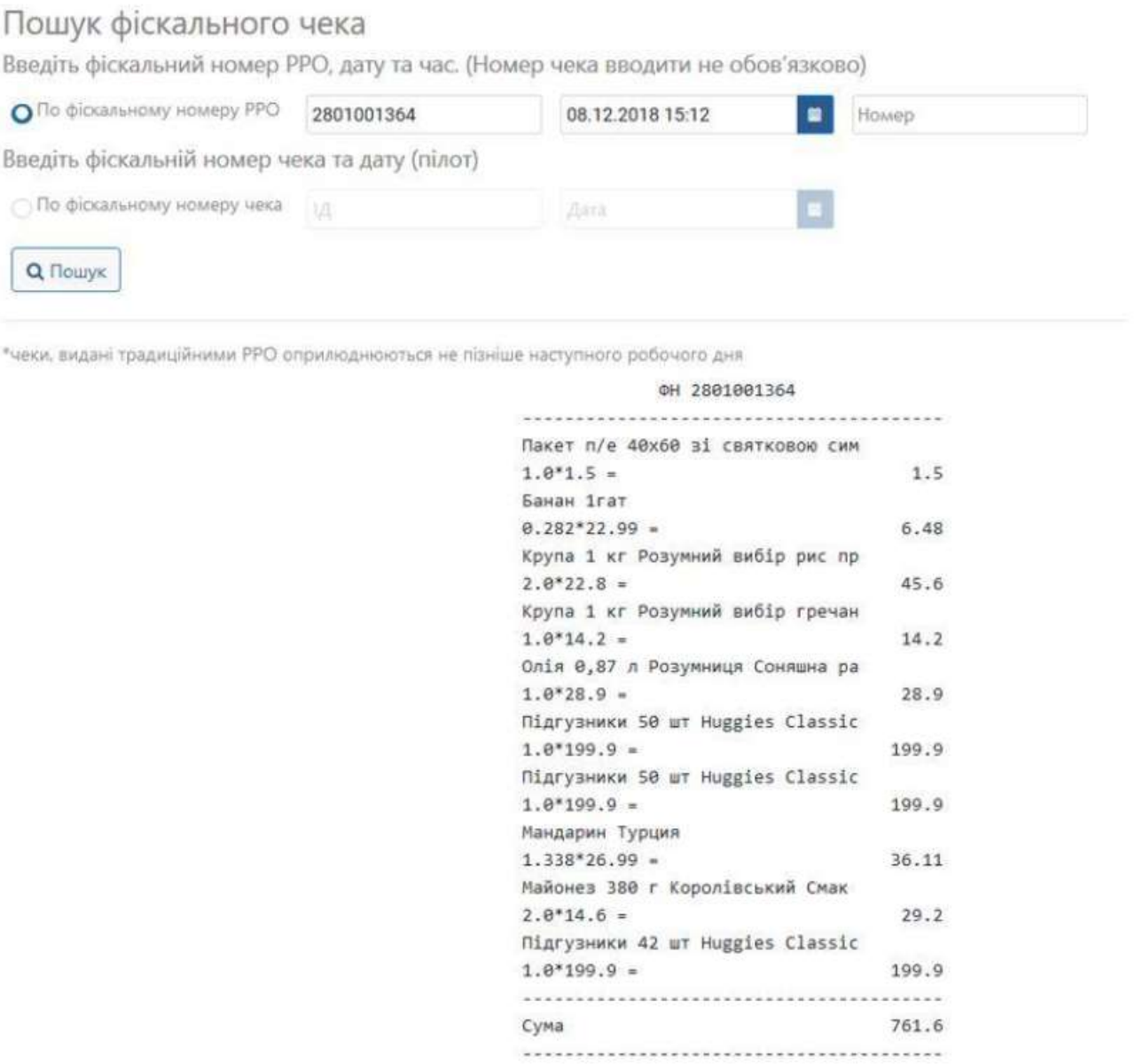


Рисунок 1.2 – Веб-інтерфейс ресурсу E-Receipt

Сервіс знаходиться у відкритій частині Електронного кабінету Державної податкової служби України, доступ до якої здійснюється в режимі реального часу (24/7/365) без ідентифікації особи за посиланням: https://cabinet.tax.gov.ua/cashregs/check, але на період воєнного стану доступ до публічних електронних ресурсів обмежений, тому на даний час не має можливості скористатися даним сервісом. Для пошуку та перегляду фіскального касового чеку необхідно ввести фіскальний номер РРО дату та час

видачі чеку (для традиційних РРО). Пошук електронного чеку здійснюється за його фіскальним номером та датою.

Дана інформаційна система забезпечує зберігання інформації з чеків одразу у текстовому вигляді завдяки використанню програмного забезпечення Державної податкової служби суб'єктами господарювання

Суттєвими перевагами E-Receipt є:

- автоматичне потрапляння інформації з чеку безпосередньо до власної бази даних;

- доступність даних у вигляді тексту а не зображення;

У контексті використання E-Receipt для виконання задач, вирішення яких є темою даної роботи – система має наступні недоліки:

- Для отримання інформації з системи необхідно знати Фіскальний номер реєстратора розрахункових операцій (РРО) та дату операції, що вказаний у чеку, або фіскальний номер чеку, тобто треба мати або оригінал чеку або інший чек від даного продавця;

- Неможливо визначити надійність збереження даних так як відсутня інформація про засоби та терміни зберігання[17].

Google Trends

Сервіс надає доступ до практично не фільтрованої вибірки фактичних пошукових запитів у Google. Він анонімізований (ніхто не ідентифікований особисто), категорізований (вибір теми пошукового запиту), агрегований (згрупований). Google Trends дозволяє шукати певну тему в Google або певний набір пошукових термінів, що дозволяє виявляти інтерес до теми чи пошукового терміна протягом певного часу, де їх найчастіше шукають (рівень міста, області, країни чи глобальний) або, що ще люди шукають разом з цим запитом. [18]

Переваги сервісу:

- Сервіс безкоштовний;

- Репрезентативна вибірка, що сформована на основі мільйонів щоденних запитів користувачів;

- Широка географія аналітики;

- Можливість порівнювати між собою динаміку популярності декількох пошукових запитів.

Недоліки:

- Складність пошуку потрібної інформації, що викликана необхідністю правильно формулювати запит;

- Відсутня можливість ввести потрібний населений пункт для перегляду популярності запиту, лише вибір зі списку, що надає сервіс;

- Популярність запиту визначається в балах, тобто відсутні данні про реальну кількість користувачів, що здійснювали пошук;

- Не можливо визначити свою цільову аудиторію.

Державна служба статистики України Державна служба статистики України (Держстат) є спеціально уповноваженим центральним органом виконавчої влади в галузі статистики, діяльність якого спрямовується та координується Кабінетом Міністрів України та який забезпечує формування і реалізує державну політику у сфері статистики.[19] На офіційному сайті держстату 2 рази на рік публікуються дані про "Витрати і ресурси домогосподарств України (за даними вибіркового обстеження умов життя домогосподарств)". Ці дані знаходяться у відкритому доступі. Держстат надає інформацію про структуру сукупних витрат, про споживання продуктів харчування в домогосподарствах, про структуру сукупних витрат домогосподарств залежно від рівня середньодушових еквівалентних загальних доходів, тощо (рис. 1.3).

Споживання продуктів харчування в домогосподарствах

(у середньому за місяць у розрахунку на одну особу, кг/on average per month per person, kg)

	9 місяців 2020/9 months of 2020					9 місяців 2021/9 months of 2021				
	усі домогос-подар-ства/all households	у т.ч. проживають/including living		домогоспо-дарства з дітьми/ households with children	домогос-подарства без дітей/ households without children	усі домогос-подар-ства/all households	у т.ч. проживають/including living		домогоспо-дарства з дітьми/ households with children	домогос-подарства без дітей/ households without children
		у міській місцевості/in urban areas	у сільській місцевості /in rural areas				у міській місцевості/in urban areas	у сільській місцевості /in rural areas		
М'ясо і м'ясопродукти	5,1	5,3	4,6	4,4	5,7	5,1	5,3	4,6	4,4	5,7
Молоко і молочні продукти	18,8	18,7	19,0	16,5	21,3	18,9	19,1	18,4	16,4	21,6
Яйця, шт	20	19	20	18	22	19	19	19	17	21
Риба і рибопродукти	1,3	1,3	1,3	1,1	1,5	1,3	1,3	1,3	1,1	1,5
Цукор	2,5	2,3	2,9	2,2	2,8	2,3	2,2	2,7	2,1	2,6
Олія та інші рослинні жири	1,4	1,3	1,5	1,2	1,6	1,3	1,2	1,4	1,1	1,5
Картопля	5,9	4,9	7,7	5,2	6,6	5,6	4,7	7,5	5,0	6,3
Овочі і баштанні	9,2	9,0	9,5	7,5	10,8	9,1	8,8	9,2	7,5	10,7
Фрукти, ягоди, горіхи, виноград	3,5	4,1	2,6	3,4	3,8	3,6	4,3	2,7	3,6	3,8
Хліб і хлібні продукти	8,0	7,3	9,5	6,9	9,3	7,6	7,0	8,7	6,4	8,9

Рисунок 1.3 – Приклад даних про витрати опублікованих держстатом

Як видно з рисунку 1.3 дані надаються тільки для основних груп товарів і представлені в загальному вигляді. Це дає обмежені можливості для аналітики витрат.

Переваги:

- достовірність даних;
- різноманітність показників

Недоліки:

- узагальненість даних;
- періодичність публікації

Наведені вище приклади лише частково вирішують окреслену в роботі проблему, та не забезпечують повною мірою виконання задач та не задовольняють вимогам даної роботи, також перелічені системи мають цілі, відмінні від цілей виконуваної роботи

1.3. Вивчення підходів вирішення проблеми

Збір даних про витрати

Данні про витрату з’являються у момент її здійснення. На сьогоднішній день існує велика кількість засобів для збереження інформації про персональні витрати, до таких можна віднести фізичне збереження розрахункових документів, або зовнішні накопичувачі (компакт диски, флешки, зовнішні жорсткі диски тощо) для збереження оцифрованих копій чеків. Надійність використання таких систем невисока – папір, що використовується для друку чека, швидко псується, а зовнішні накопичувачі можуть вийти з ладу, або згубитися. Додатковим мінусом підходу є великий проміжок часу між запитом інформації та її представленням.

Іншою можливістю зберігання є використання спеціальних застосунків для ведення персонального обліку витрат для смартфона чи комп’ютера, в такому випадку дані можна отримати швидко, проте надійність зберігання все ще не висока з тих же причин, що і для зовнішніх накопичувачів: мобільні пристрої можуть бути загублені, накопичувач у комп’ютері може вийти з ладу.

Також існує можливість зберігання даних про витрати до хмарних сховищ. Хмарне сховище – це сервіс зберігання даних побудований на серверних технологіях. Надійність такого сервісу обумовлена використанням дата центрів (центри зберігання та обробки даних) - спеціалізованих технічних майданчиків для розміщення інформації в Інтернеті, підключених до неї в автономну систему (або мережу в її складі) по множині каналів зв'язку.

Збереження даних

Для збереження даних про витрати необхідно з’ясувати їх структуру. Після здійснення витрати дані доступні у вигляді розрахункового документа. На території України зміст та форма розрахункового документа визначена "Положенням про форму та зміст розрахункових документів/електронних розрахункових документів" Міністерства Фінансів затвердженого наказом від 21.06.2016 №13. [20] Перелік атрибутів розрахункового документа наведений у Додатку 1 до вищезазначеного положення:

1. найменування суб’єкта господарювання;

2. назва господарської одиниці;

3. адреса господарської одиниці;

4. Індивідуальний податковий номер платника ПДВ;

5. Податковий номер/серія (за наявності) та номер паспорта;

6. Кількість;

7. Вартість;

8. назва товару;

9. код товарної підкатегорії;

10. цифрове значення штрихового коду товару;

11. літерне позначення ставки ПДВ;

12. ідентифікатор торгівця

13. ідентифікатор платіжного пристрою;

14. сума комісійної винагороди;

15. вид операції;

16. Реквізити електронного платіжного засобу

17. Платіжна система;

18. Код авторизації;

19. Касир;

20. Форма сплати;

21. Сума сплачених коштів;

22. Валюта;

23. Загальна вартість придбаних товарів;

24. ПДВ;

25. Акциз;

26. Сума до сплати;

27. Номер чека;

28. Дата час;

29. Заводський номер реєстратора розрахункових операцій;

30. Фіскальний номер реєстратора розрахункових операцій

На основі наведеної інформації створено таблицю сутностей та їх атрибутів:

Таблиця 1.1 – Перелік сутностей та їх атрибутів

Сутність	Атрибути (номери пунктів включно)
Чек	12-30 + email + категорія + файл
Суб’єкт господарювання	1-5
Товар	6-11
Користувач	Email, пароль, персональні дані, рівень доступу

До переліку атрибутів сутності "Чек" додано ідентифікатор користувача, що є власником та джерелом даних, абстрактну категорію, до якої власник відносить дану витрату та файл, що є цифровою копією чеку та реалізує функціонал збереження документа. З точки зору надійності збереження даних, БД є хмарним сервісом, тож розроблювана система має засоби для роботи із хмарними сервісами.

Простий та швидкий доступ до раніше збережених даних

З метою ідентифікації та співставлення відвідувача системи та власника даних до БД також зберігаються облікові дані, тож таблиця 1.1 містить ще одну сутність – Користувач. Дані мають бути доступними за умови передачі принаймні ідентифікатора користувача. Додатково передбачено можливість оновлення(доповнення) характеристик вже збережених даних та їх вилучення. Тобто до системи надсилаються повідомлення декількох типів:

- додавання інформації;
- отримання інформації;
- оновлення даних;
- вилучення даних

Система таких повідомлень називається API – (англ. Application Programming Interface) – набір чітко визначених методів для взаємодії різних компонентів. В нашому випадку арі забезпечує взаємодію розроблюваного веб-

сервісу та джерел надходження даних про витрати, і має в розпорядженні повідомлення вищезазначених типів.

Уніфікація способу надсилання даних за допомогою API дозволяє відокремити задачі зчитування даних про витрати із носіїв (чеки, записи "від руки" тощо) та власне задачу аналізу і зберігання таких даних. Що також спрощує задачу для розробників, яким не потрібно реалізовувати фунціонал зберігання та аналізу, а лише зчитування.

1.4. Постановка задачі

Після отримання системою даних, слід перебачити можливість їх аналізу, тобто обробку детальної інформації про кожну окрему затрату, таку як категорія, назва, і дата покупки, а також сума і валюта витрати, з метою їх подальшого сортування і перетворення в практичні висновки. Прикладом таких висновків є аналітика попиту на електрочайники за рік по місяцям, яка може бути представлена у вигляді таблиці чи візуалізована графічно. Тому розроблювана система повинна мати набір інструментів та методів обробки даних, що здатні виконати такий аналіз.

Користувачам будуть безкоштовно надаватися результати аналізу їх персональних витрат, а аналітика агрегованих даних всіх користувачів буде доступна на комерційній основі, тож таблиця 1.1 містить атрибут "рівень доступу" для сутності "Користувач" що реалізує диференціацію користувачів із безкоштовним доступом та комерційним. Для реалізації самого механізму диференціації в веб-сервісі розроблений кабінет користувача.

Дана інформаційна системи має працювати із хмарними сервісами (сховище і БД), містити логіку обробки даних, має сайт з кабінетом користувача та API як інтерфейс доступу до даних.

РОЗДІЛ 2 ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1. Взаємодія користувацьких додатків із системою

Програмні платформи, що можуть служити джерелами даних для розроблюваної системи, можуть бути різноманітними. Це і мобільні додатки, і прикладні програми і веб-сайти для обліку персональних витрат. Дані будуть передаватися у форматі JSON.

У випадках, коли необхідно налагодити взаємодію невизначених систем та розроблюваної системи доцільно запровадити універсальний інтерфейс взаємодії – API (Прикладний програмний інтерфейс) – набір визначень підпрограм, протоколів взаємодії та засобів для створення програмного забезпечення. Спрощено – це набір чітко визначених методів для взаємодії різних компонентів. API надає розробнику засоби для швидкої розробки програмного забезпечення. API може бути для веб-базованих систем, операційних систем, баз даних, апаратного забезпечення, програмних бібліотек.

Існує декілька підходів реалізації WEB API, усі вони мають свої особливості, переваги та недоліки, двома найпоширенішими є:

REST – підхід до архітектури мережевих протоколів, які забезпечують доступ до інформаційних ресурсів. Був описаний і популяризований 2000 року Роєм Філдінгом, одним із творців протоколу HTTP. В основі REST закладено принципи функціонування Всесвітньої павутини і, зокрема, можливості HTTP. Філдінг розробив REST паралельно з HTTP 1.1 базуючись на попередньому протоколі HTTP 1.0 [21].

SOAP – протокол обміну структурованими повідомленнями в розподілених обчислювальних системах, базується на форматі XML [22].

Спочатку SOAP призначався, в основному, для реалізації віддаленого виклику процедур (RPC), а назва була абревіатурою: Simple Object Access

Protocol – простий протокол доступу до об'єктів. Зараз протокол використовується для обміну повідомленнями в форматі XML, а не тільки для виклику процедур. SOAP є розширенням мови XML-RPC.

SOAP можна використовувати з будь-яким протоколом прикладного рівня: SMTP, FTP, HTTP та інші. Проте його взаємодія з кожним із цих протоколів має свої особливості, які потрібно відзначити окремо. Найчастіше SOAP використовується разом з HTTP.

SOAP є одним зі стандартів, на яких ґрунтується технологія веб-сервісів

Таблиця 2.1 – Порівняльна характеристика SOAP та REST

SOAP	REST
SOAP – насамперед є протоколом. Має визначену специфікацію. Специфікацією передбачено наявність WSDL файлу, який містить інформацію щодо функціоналу веб-сервісу	REST – Архітектурний стиль, що перетворює веб сервіс на RESTful коли останній відповідає наступним вимогам Клієнт Серверний Без збереження стану Кешований Багатошарова система Уніфікований інтерфейс
SOAP не може використовувати REST, оскільки SOAP це протокол, а REST – архітектурний стиль	REST може використовувати SOAP як базовий протокол
SOAP використовує інтерфейси для надання функціоналу клієнтам. SOAP запроваджує файл WSDL який містить необхідно інформацію щодо функціоналу сервісу	REST використовує Uniform Service locators для доступа до компонентів системи. Наприклад, якщо є об’єкт employee за адресом http://example.com.

SOAP вимагає більшої пропускної здатності, оскільки повідомлення SOAP містять багато інформації всередині: <?xml version="1.0"?> <SOAP-ENV:Envelope xmlns:SOAP-ENV ="http://www.w3.org/2001/12/soap-envelope" SOAP-ENV:encodingStyle =" http://www.w3.org/2001/12/soap-encoding"> <soap:Body> <Demo.example xmlns="http://tempuri.org/"> <EmployeeID>int</EmployeeID> </Demo.example> </soap:Body> </SOAP-ENV:Envelope>	REST не потребує великої пропускної здатності під час відправлення повідомлень на сервер. REST повідомлення містять, як правило, JSON повідомлення. Нижче наведено приклад, з якого видно, що розмір повідомлення у порівнянні з SOAP значно менший: {"city":"Sumy","country":"Ukraine"}
SOAP працює тільки з форматом XML	REST може використовувати різні формати текст, HTML, XML, JSON і т.д., але найбільш вживаним є формат JSON

Виходячи із більшої гнучкості архітектури REST [23], простоти використання, меншого обсягу даних, що передаються, - для даної роботи обрано саме REST API у якості шляху взаємодії користувацьких додатків та розроблюваної системи [17]. Схема взаємодії користувацьких додатків із системою зображена на Рисунку 2.1

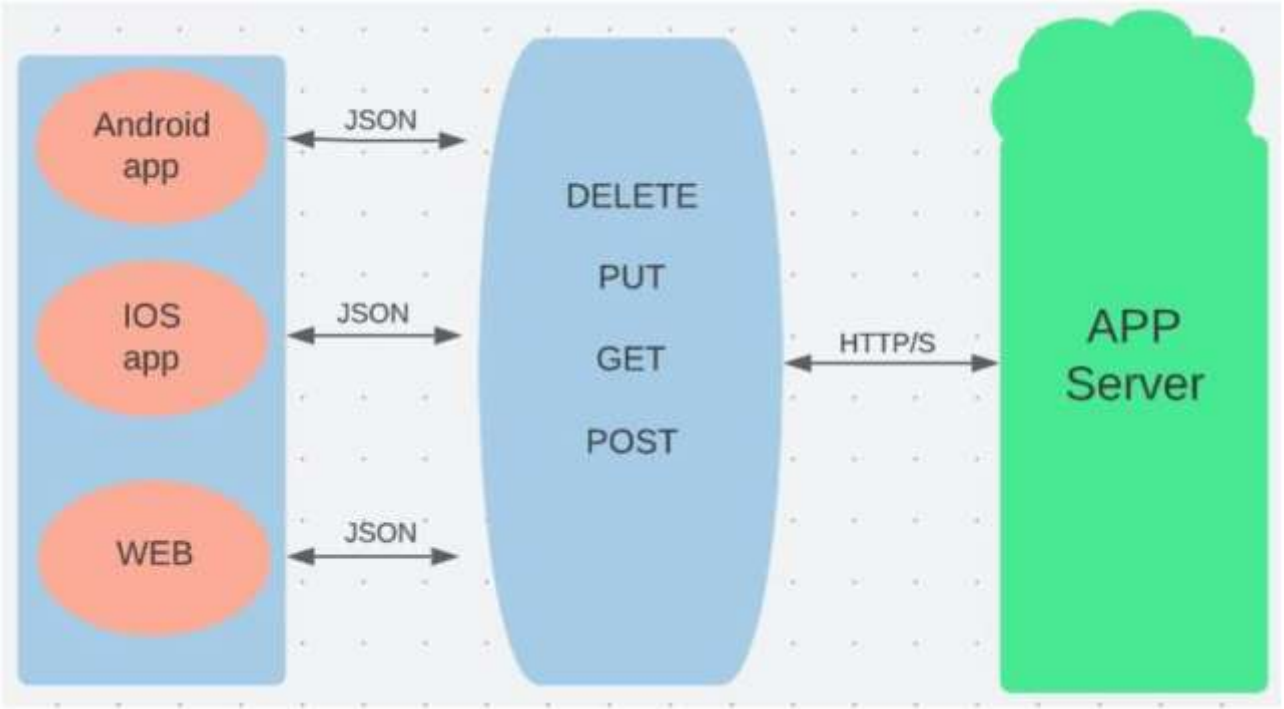


Рисунок 2.1 – Схема взаємодії додатків із системою

2.2. Вибір методу авторизації

Як зазначено вище даної роботи – система повинна здійснювати реєстрацію та авторизацію користувачів. Сучасні технології авторизації для веб-систем можна розділити на дві групи:

1. Session-Based – спосіб автентифікації при якому клієнт відправляє на сервер ідентифікаційні дані (наприклад логін та пароль) за допомогою HTTP запиту, після чого сервер порівнює отримані дані із наявними в себе, і, у разі співпадання даних, присвоює унікальний ідентифікатор сесії (Session id) та відправляє його у складі HTTP відповіді (рисунок 2.2). При подальших запитах від клієнта використовується отриманий Session id, що передається у заголовку Cookies.

2. Token-Based – алгоритм автентифікації, при якому розрізняються принаймні три учасники процесу, власник ресурсу, сервіс та сховище ресурсів.

Сервіс надає доступ до сховища ресурсів власнику. Приклад реалізації схеми зображено на рисунку 2.3

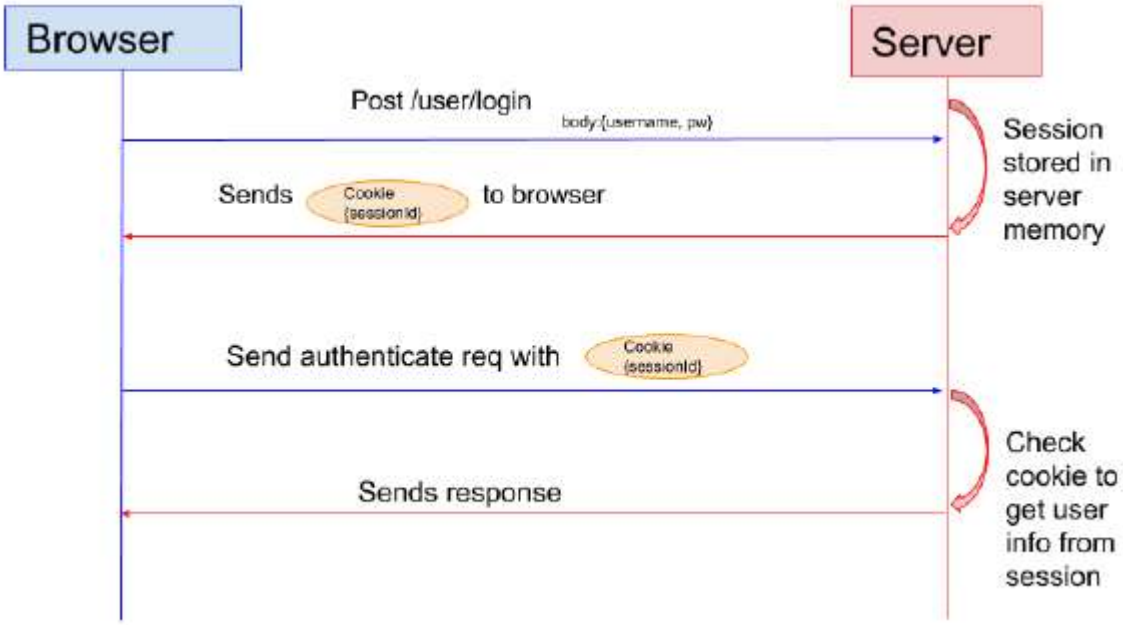


Рисунок 2.2 – Session Based Автентифікація

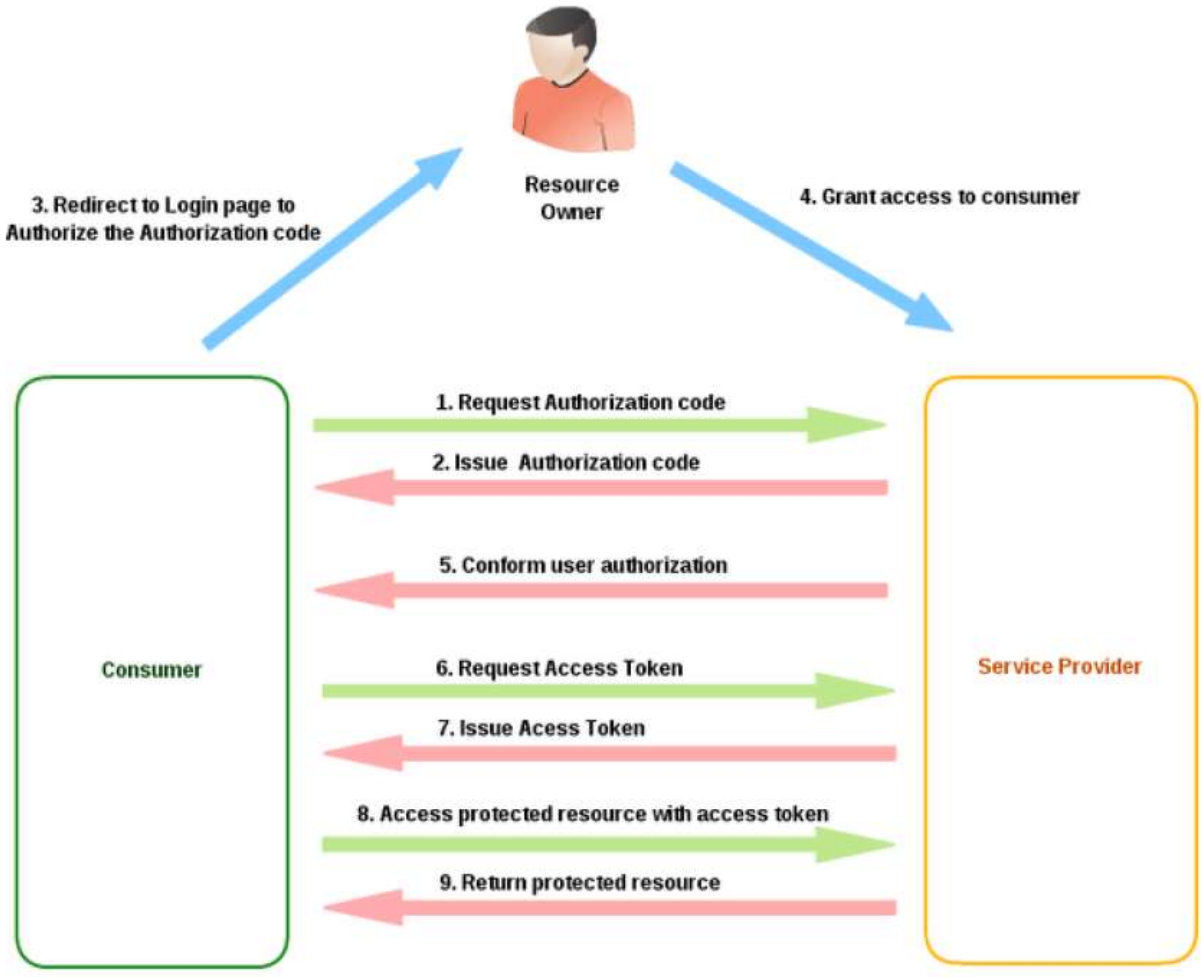


Рисунок 2.3 – Token-Based автентифікація

При використанні Session based автентифікації відомості про сесію зберігаються у файлах cookie. HTTP-cookie – у комп'ютерній термінології поняття, яке використовується для опису інформації у вигляді текстових або бінарних даних, отриманих від веб-сайту на веб-сервері, яка зберігається у клієнта, тобто браузера, а потім відправлена на той самий сайт, якщо його буде повторно відвідано. Зберіганням та обробкою cookie файлів зазвичай займається браузер. Під час використання токенів куки файли не використовуються. Токен передається у http заголовку.

Розроблювана система не передбачає використання API користувачем із браузера, натомість користувач буде взаємодіяти із інтерфейсом програми, а програма буде взаємодіяти із системою через REST api. Тому у якості алгоритму авторизації обрано саме Token-based автентифікацію.

Найбільшого розповсюдження досяг стандарт OAuth:

Oauth – це відкритий стандарт авторизації, який дозволяє користувачам відкривати доступ до своїх приватних даних (фотографії, відео, списки контактів), що зберігаються на одному сайті, іншому сайту, без необхідності вводу імені користувача та паролю.

OAuth [24] дозволяє користувачам роздавати сайтам маркери доступу, до даних що розміщуються на сайтах-сервісах. Кожен маркер доступу надає доступ конкретному сайту (наприклад, сайту редагування відео) до конкретних ресурсів (наприклад, тільки відео від конкретного альбому) та на визначений термін (наприклад, на наступні 2 години). Це дозволяє користувачам надавати доступ третім сайтам до їх інформації, що зберігається на інших сайтах — постачальниках послуг, не передаючи повною мірою самих даних та без застосування імені/паролю. [17].

2.3. Вибір провайдерів хмарних сервісів

Факторами, що впливають на вибір хмарного сховища для зберігання основних даних – цифрових копій розрахункових документів, є наявність та

доступність інструментів програмної взаємодії (API) із хмарним сховищем, вичерпна документованість функцій API, безоплатність (найменша вартість) використання API, надійність сервісу, найменша вартість одиниці об’єму сховища.

Таблиця 2.2 – Орієнтовна вартість сервісів хмарного сховища на кінець 2022 р.

	Вартість 1ГБ/місяць	Використання мережі, 1ГБ (in/out)	1000 операцій читання/запису
Amazon	$0.023	$0/$0.09	$0.00043/$0.0054
Google	$0.023	До $0.14	$0.0004/$0.005
Microsoft	$0.02	$0/$0.01	$0.0006/$0.007

Як видно з Таблиці 2.2 – пропозиції від найбільших провайдерів хмарних сервісів практично однакові з точки зору вартості, тож у якості сховища буде обрано хмарний сервіс від Amazon.

Критерії вибору хмарного сервісу для розгортання БД такі ж як і для хмарного сховища, тож вибір буде між Amazon, Google та Microsoft.

Таблиця 2.3 – Орієнтовна вартість хмарних сервісів БД на кінець 2022р.

	Вартість 1год	Сховище, 1ГБ	Число з’єднань	Мережа, 1ГБ
Amazon	$0.02	$0.137		$0.09
Google	$0.0126	$0.108	250	До $0.12
Microsoft	$0.0202	fixed	50	$0.09

Як видно з таблиці 2.3 – різниця у характеристиках хмарних сервісів для розгортання БД не суттєва, тож для запобігання використання великої кількості бібліотек у майбутньому, будемо використовувати сервіс Amazon і для БД теж. Amazon Relational Database Service (Amazon RDS) дозволяє просто налаштовувати, використовувати і масштабувати реляційні бази даних в хмарі.

Сервіс забезпечує економічне і масштабується використання ресурсів при одночасній автоматизації трудомістких завдань адміністрування, таких як виділення апаратного забезпечення, налаштування бази даних, установка виправлень і резервне копіювання. Це дозволяє зосередити увагу на додатках, щоб забезпечити для них високу продуктивність, високу доступність, безпеку і сумісність.

Amazon RDS доступний у вигляді інстансу бази даних декількох типів: оптимізовані для роботи з пам'яттю, для високої продуктивності або виконання операцій введення-виведення - і пропонує на вибір шість відомих ядер баз даних, в тому числі Amazon Aurora, PostgreSQL, MySQL, MariaDB, Oracle Database і SQL Server. За допомогою сервісу AWS Database Migration Service можна просто перенести або реплікувати існуючі бази даних в Amazon RDS [25].

Amazon Simple Storage Service (Amazon S3) - це сервіс зберігання об'єктів, що пропонує кращі в галузі показники продуктивності, масштабованості, доступності та безпеки даних. Це означає, що клієнтами Amazon можуть бути компанії будь-яких розмірів і з будь-яких областей діяльності. Вони можуть використовувати сервіс для зберігання і захисту будь-яких обсягів даних в різних ситуаціях, наприклад для забезпечення роботи сайтів, мобільних додатків, для резервного копіювання та відновлення, архівації, корпоративних додатків, пристроїв IoT і аналізу великих даних. Amazon S3 пропонує прості у використанні інструменти адміністрування, які дозволяють організувати дані і точно налаштувати обмеження доступу відповідно потребами бізнесу або законодавчими вимогами. Amazon S3 забезпечує надійність +99,999999999% (тут 11 дев'яток) і зберігає дані мільйонів додатків в інтересах компаній з усього світу [26].

2.4. Вибір бази розроблюваної системи

Для реалізації вищевказаних елементів системи існують готові окремі рішення. У розроблюваній системі вони мають бути частиною цілого із налагодженою взаємодією. Таку можливість надають розповсюджені фреймворки, каркаси веб-застосунків, на базі яких можна побудувати систему, що задовольняє різноманітним потребам.

"Каркас веб-застосунків, веб-фреймворк – програмний каркас, призначений для створення веб-застосунків, служб або ресурсів. Він спрощує розробку, частково за рахунок автоматизації, і позбавляє від необхідності написання рутинного коду. Більшість каркасів спрощують доступ до баз даних, також зменшують дублювання коду".

Існує багато веб-фреймворків, більшість з них реалізують шаблон MVC – архітектурний шаблон, який використовується під час проектування та розробки програмного забезпечення. Цей шаблон передбачає поділ системи на три взаємопов'язані частини: модель даних, вигляд (інтерфейс користувача) та модуль керування. Застосовується для відокремлення даних (моделі) від інтерфейсу користувача (вигляду) так, щоб зміни інтерфейсу користувача мінімально впливали на роботу з даними, а зміни в моделі даних могли здійснюватися без змін інтерфейсу користувача."[17]

Найбільш відомими, серед PHP веб-фреймворків, є Symfony і Laravel. Для здійснення зваженого вибору веб-фреймворку порівняємо їх основні характеристики (таб. 2.4).

Таблиця 2.4 – Порівняння фреймворків PHP

Характеристики	Laravel	Symfony
Модульність та масштабування	Використовує програми на основі MVC із низкою попередньо створених залежностей. Це робить його трохи менш гнучким, але більш зручним, якщо ви використовуєте програми MVC.	Використовує багаторазові компоненти, що забезпечує більш надійну модульність. Код організовано краще.
Механізм створення шаблонів	Механізмом створення шаблонів за замовчуванням є Blade, що дозволяє повторне використання коду, чого немає у Twig. Lumen – ще один субфреймворк, який використовує фреймворк, що робить його ідеальним для створення API.	Використовує Twig як механізм створення шаблонів за замовчуванням.

Підтримка баз даних	Використовує об’єктно-реляційне відображення (ORM) для доступу до даних через Eloquent. Laravel підтримує наступні бази даних із коробки: MySQL, PostgreSQL, SQLite та SQLServer.	Використовує об’єктно-реляційне відображення (ORM) для доступу до даних через Doctrine. Symfony підтримує такі бази даних із коробки: Drizzle, MySQL, Oracle, PostgreSQL, SAP Sybase SQL Anywhere, SQLite, SQLServer.
Міграції бази даних	Міграції бази даних виконуються вручну, але не потребують визначення полів.	Перенесення даних відбувається автоматично, вимагаючи лише простих визначень для полів у моделі.
Моделювання даних	Потрібні серйозні знання SQL. Крім того, Eloquent зазвичай прив’язує вашу програму до схеми БД, що робить її менш гнучкою в цьому плані	Не вимагає серйозних знань SQL, хоча ви повинні створити репозиторій для кожного виклику.

Проаналізувавши основні характеристики цих двох фреймворків можна зробити висновок, що обидва чудово підходять для веб-розробки. За допомогою Laravel та Symfony можна створювати проекти будь-якої складності, які вимагають нетрадиційних рішень. Варто зазначити що більшість PHP фреймворків побудовані на базі Symfony. Для забезпечення найбільшої гнучкості розроблюваної системи та створення потенціалу для майбутніх вдосконалень перевагу буде надано PHP веб-фреймворкіу Symfony.

2.5. Механізм отримання користувачем кінцевого результату

Програмний інтерфейс доступу забезпечує сервіс даними від різних користувацьких систем, але це все ще обмін інформацією на рівні система – система, доцільно передбачити можливість спілкування на рівні користувач – система. Оскільки даний проект є веб-сервісом реалізованим на базі php веб фреймворку то доступ для користувача буде запроваджено через веб інтерфейс із використанням можливостей Symfony, у результаті має бути створений кабінет користувача із відповідним набором можливостей.

Система агрегує дані від користувачів, що ведуть облік своїх витрат, також агрегована та консолідована аналітика планується надаватись на комерційній основі користувачам-представникам зацікавлених компаній. Як наслідок маємо 2 різні типи користувачів, для кожного з яких буде реалізовано власний тип кабінету. Дані, що надходитимуть до системи через API, можуть бути наслідком ручного користувацького введення, існує вірогідність типових помилок. Для великих об’ємів даних доцільно передбачити виконання періодичних, калькуляційних операцій. Зважаючи на вищевказане необхідно також розробити кабінет для виконання адміністративних дій із даними. Усі 3 типи кабінетів мають бути реалізованими засобами фреймворку Symfony. Функціонал кабінетів наступний:

Таблиця 2.5 – Функціонал кабінетів

Кабінет користувача	-авторизація -редагування власних даних -доступ до існуючої номенклатури -система відгуків та оцінювання -перегляд статистики на основі власних даних
Кабінет компанії	-авторизація -реакція на відгуки до власної продукції -статистика та аналітика -зворотній зв’язок
Кабінет адміністрування	-авторизація -реакція на зворотній зв’язок -рутинні задачі із даними -модерація відгуків

Для демонстрації можливостей системи та заохочування представників компаній до співпраці необхідно розробити односторінковий веб-сайт також відомий як Лендінг - це завершальна сторінка колонки продаж, також – веб-сторінка, яка відкривається при натисканні на рекламне оголошення чи ланку (лінк). "Цільова сторінка" є логічним продовженням рекламного оголошення або посилання. Часто "лендінги" пов'язані з соціальними медіа, розсилками електронною поштою або маркетинговими кампаніями пошукових двигунів (контекстною рекламою) з метою підвищення ефективності реклами [27]. Схема взаємодії компонентів веб-сайту зображені на рисунку 2.4.

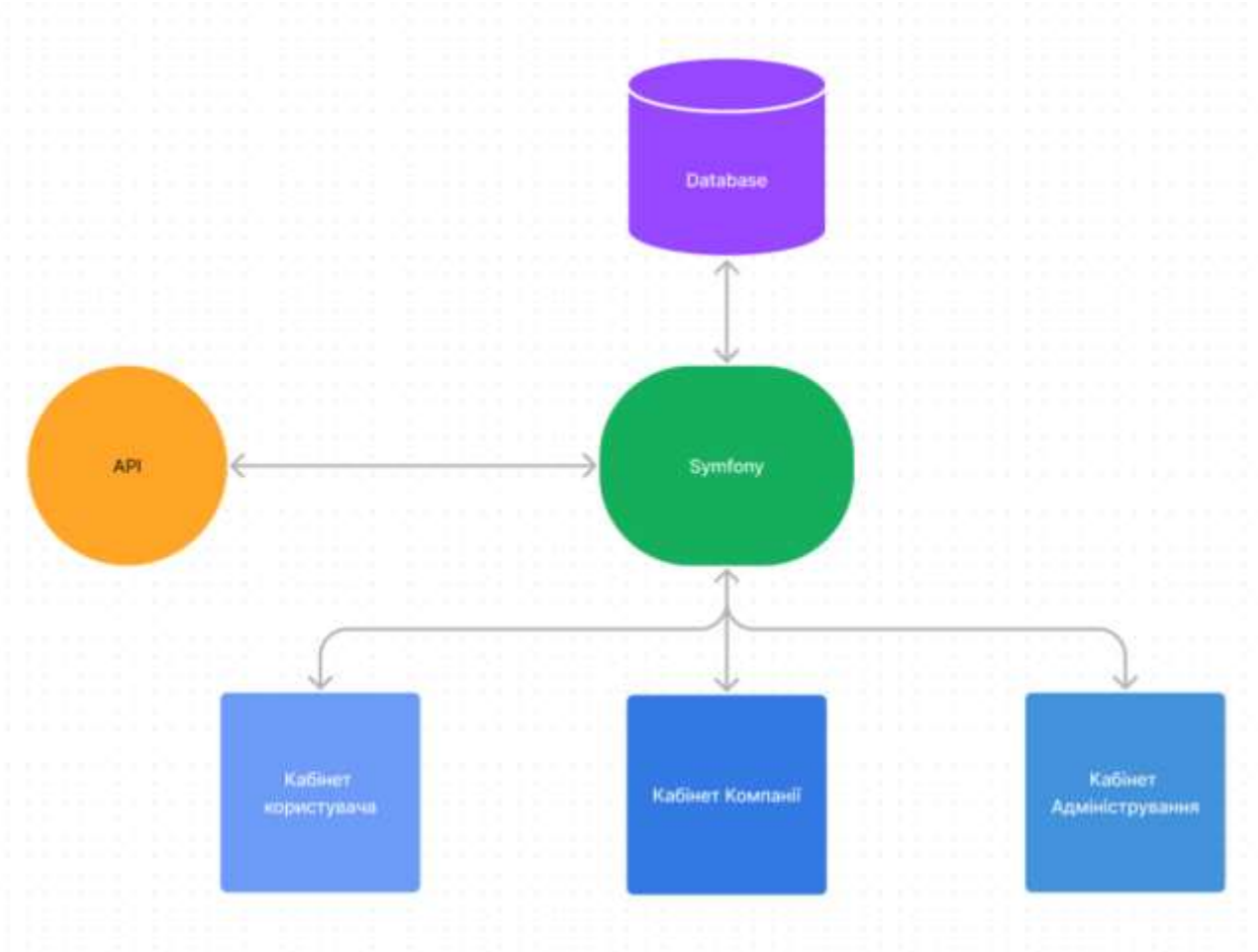


Рисунок 2.4 – Компоненти веб-сайту

2.6. Реалізація аналізу даних

Надання аналітики витрат кінцевому користувачу є важливим етапом розроблюваної системи. веб-сервіс надає два види аналітичних даних:

- розширений аналіз персональних витрат користувача (його власних), який буде надаватися безкоштовно;

- агрегована аналітика витрат (всіх користувачів), буде надаватись на комерційній основі.

Аналітичні дані будуть представлені у вигляді діаграм. Для реалізації цього функціоналу буде використовуватися Symfony UX Chart.js – це пакет Symfony, що інтегрує бібліотеку Chart.js в проекти Symfony.[28]

Chart.js – це безкоштовна бібліотека JavaScript для створення діаграм на основі HTML. Це одна з самих простих бібліотек візуалізації на JavaScript, яка поставляється з наступними вбудованими типами діаграм:

- Точкова діаграма
- Лінійний графік
- Гістограма
- Кругова діаграма
- Пончикова діаграма
- Пухирцева діаграма
- Діаграма області
- Радарна діаграма
- Змішана діаграма[29]

Приклад доступних в Chart.js діаграм зображено на рисунку 2.5

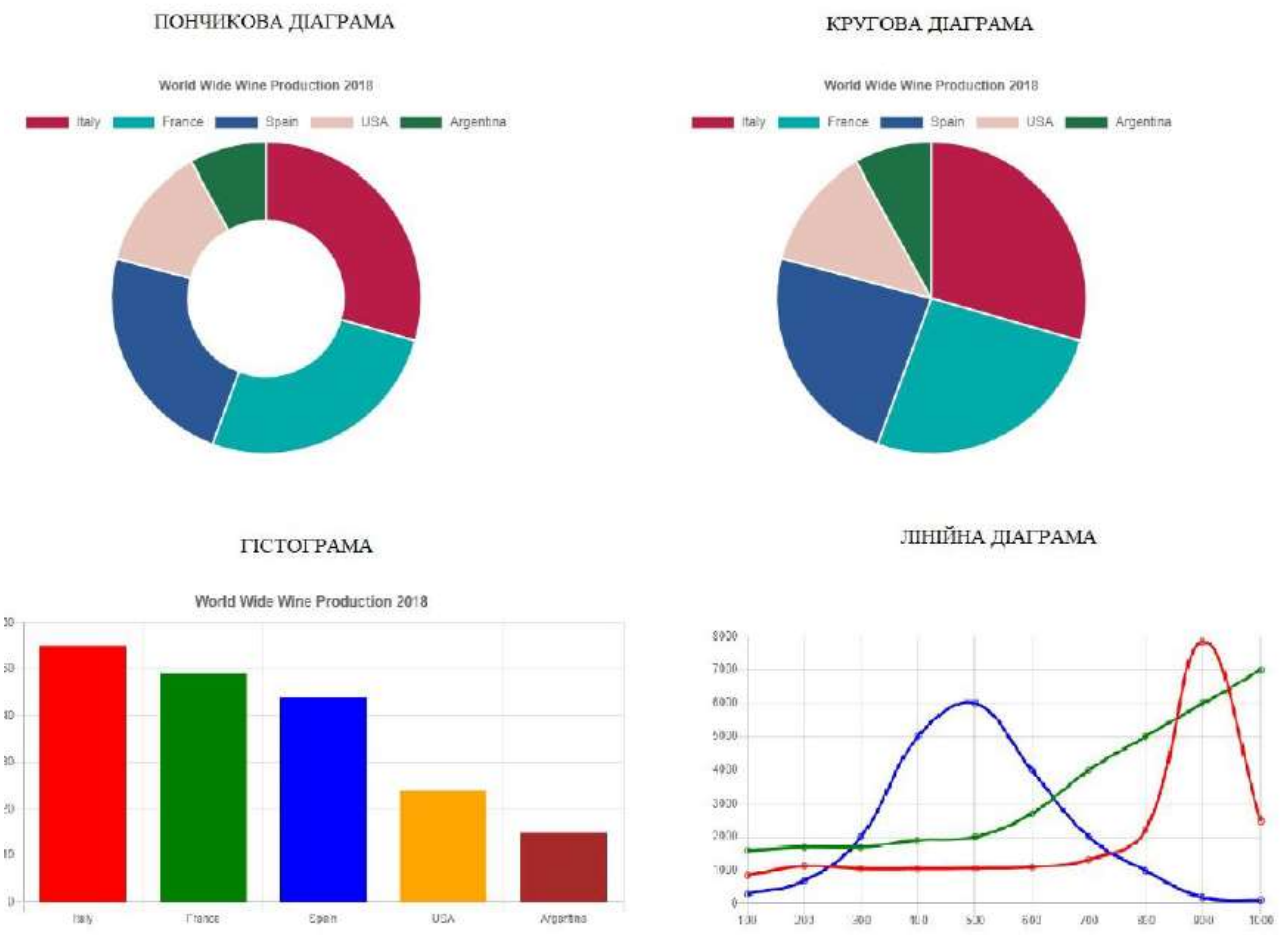


Рисунок 2.5 – Можливості Chart.js

Для отримання аналітики витрат користувач повинен авторизуватися на сайті. Після цього в особистому кабінеті він може зробити запит аналітичних даних. Для формування запиту у особистому кабінеті буде доступний фільтр критеріїв аналізу, на основі яких буде проводитись аналітика витрат (рис. 2.6).

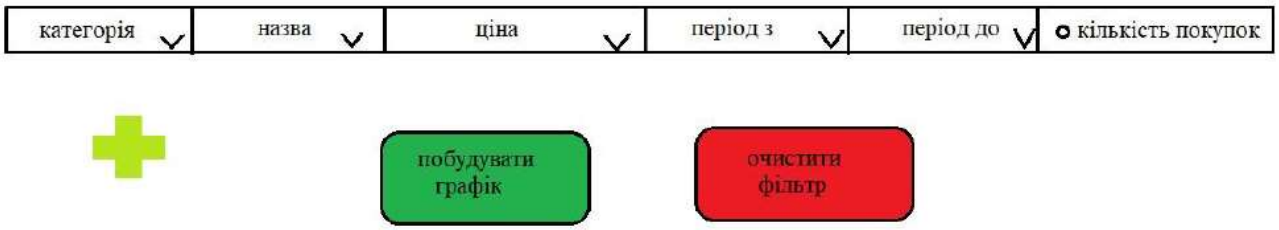


Рисунок 2.6 – Фільтр для формування запиту аналітики

Як видно з рисунку 2.6 користувач може обрати категорію до якої належить товар/послуга зі списку, після того як буде обрано категорію стане

доступним можливість обрати назву зі списку. В полі "ціна" можна обрати з 3 варіантів: середня, максимальна і мінімальна. Також можна обрати за який період проводити аналіз витрат і якщо цікавить кількість здійснених витрат, то потрібно позначити поле "кількість покупок". Також можна добавити товар/послугу для порівняння, для цього потрібно натиснути зелений хрестик. Після натиснення кнопки "побудувати графік" користувач отримає візуалізовану аналітику витрат сформовану на основі обраних ним фільтрів у запиті. Для того, щоб очистити фільтр потрібно натиснути відповідну кнопку. В залежності від того, які критерії будуть обрані, можна отримати різні аналітичні дані, такі як:

- Кількість проданих товарів за період, для виявлення сезонності товару, чи інформації на скільки часто даний товар продається;

- Порівняти кількість покупок товару за мінімальну і максимальну ціну для аналізу того, за яку ціну товар продається краще;

- Порівняти товари між собою, щоб дізнатися який купують частіше, тощо.

2.7. Моделювання інформаційної системи

Функціональні вимоги до ІС

Розроблюваний веб-сервіс повинен відповідати наступним функціональним вимогам:

- Повинна мати графічний інтерфейс (сайт) для взаємодії системи і користувача;

- Надавати можливість авторизації і реєстрації користувача;

- Повинна мати 3 рівня доступу до системи (адміністратор, користувач, організація);

- здійснювати передачу даних з користувацьких додатків до системи;

- забезпечувати синхронізацію (зв’язування) переданих даних з користувачем в системі;

- забезпечувати користувачу можливість змінювати свої реєстраційні дані;

- забезпечувати користувачу можливість редагування свої даних про видатки;

- Можливість залишати відгуки до своїх покупок;

- Забезпечувати зворотній зв'язок з адміністратором сервісу;

- Надавати аналітичні дані (діаграми, графіки) витрат 2 рівнів: персональних і витрат всіх користувачів, останній вид на платній основі;

- Система повинна дозволяти адміністратору вносити зміни до БД;

- Адміністратор повинен мати зворотній зв'язок з користувачами: отримувати повідомлення від користувачів, відповідати на повідомлення, отримувати запити на зміну персональних даних і на зміну рівня доступу до системи, модерувати відгуки.

Нефункціональні вимоги до системи

В результаті аналізу функціональних вимог до розроблюваної ІС були сформовані наступні нефункціональні вимоги:

- Система повинна мати API для взаємодії користувацьких додатків із системою;

- Зручний метод авторизації ;

- Хмарне сховище;

- Базу даних MySQL;

- Повинна бути розроблена на базі веб-фреймворку PHP;

- Повинна мати графічний інтерфейс;

- Для створення діаграм повинен використовуватись JS;

Структура інформаційної системи

Важливим етапом проектування ІС є моделювання. На основі сформованих вимог до розроблюваної системи були визначені компоненти системи і змодельована схема їх взаємодії (рис. 1).

Компонентами розроблюваної системи є:

- API, PHP фреймворк, БД та хмарне сховище.

- API забезпечує передачу структурованих даних від користувача до фреймворку.

- Фреймворк забезпечує взаємодію граничних елементів системи.

- БД забезпечує збереження структурованих даних користувача.

- Хмарне сховище забезпечує збереження файлів, які надходять до системи від користувача.

Наступним кроком є створення контекстної IDEF0 – діаграми. Згідно вимог до розроблюваної ІС її основними функціями є отримання і збереження інформації про персональні витрати від користувачів і надання їм аналітики створеної на основі цих даних. Таким чином, визначимо головний процес контекстної діаграми, як "Веб-сервіс обліку персональних витрат". Далі визначимо вхідні і вихідні дані, а також управління і механізми. Для того, щоб користувач міг отримати аналітику персональних витрат він має бути зареєстрованим в системі і надавати дані про свої витрати, тому вхідними даними будуть персональні дані користувача і інформація про його витрати.

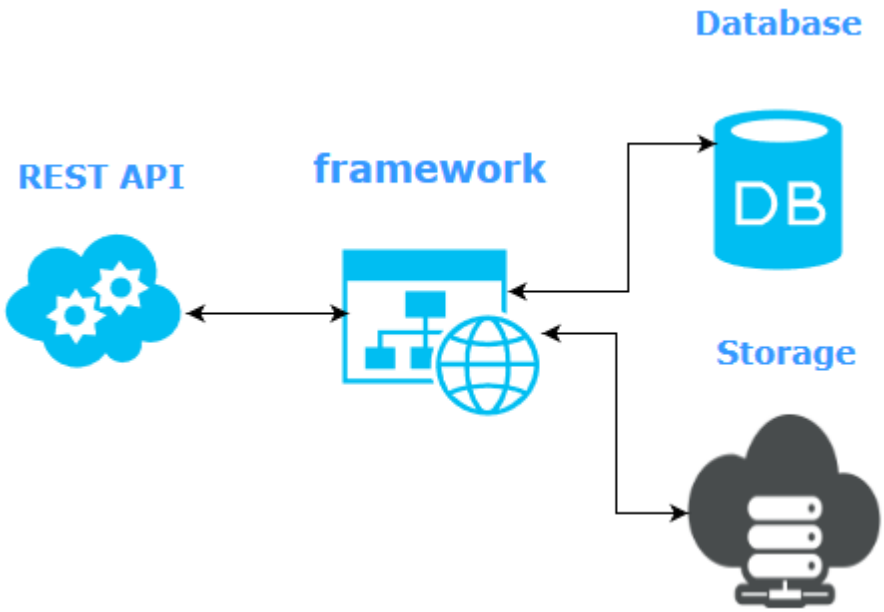


Рисунок 2.7 – Схема взаємодії елементів системи

Деякі користувачі хочуть отримувати платну аналітику витрат, для цього вони повинні мати доступ до такого функціоналу. Цей доступ надається адміністратором системи, а дозвіл на нього відображений у БД. Тому вихідними даними будуть аналітика витрат і змінена БД.

Елементами управління будуть БД, API і рівень доступу. БД містити персональні дані користувача і під час авторизації буде здійснюватися

порівняння даних з БД з даними, які вказав користувач. API регулює структуру даних, які відправляє користувач системі. Перевірка здійснюється за наступними критеріями: формат даних (тільки JSON), наявність ідентифікатора користувача (email), мінімальний рівень заповненості полів даними. Останнім елементом управління буде рівень доступу. Він регулює доступ до даних: користувачу надає доступ до його даних, компанії – до даних про витрати всіх користувачів (лише для побудови аналітики), адміністратору – доступ до управління сайтом і БД.

Механізми системи – це користувачі, адміністратор і монітор системи. Користувачі роблять запити до системи і надсилають інформацію. Адміністратор вносить зміни до БД. Монітор системи – здійснює автоматизовані процеси.

Контекстна діаграма розроблюваної ІС Веб-сервіс обліку персональних витрат представлена на рисунку 2.8.



Рисунок 2.8 – Контекстна діаграма інформаційної системи

Далі проведемо декомпозицію контекстної діаграми, описавши послідовність дій при роботі з веб-сервісом з обліку персональних видатків:

- Авторизація;
- Визначення рівня доступу;
- Запит до системи;
- Обробка запита;
- Внесення змін до БД (при необхідності).

Діаграма декомпозиції показана на рисунку 2.9.

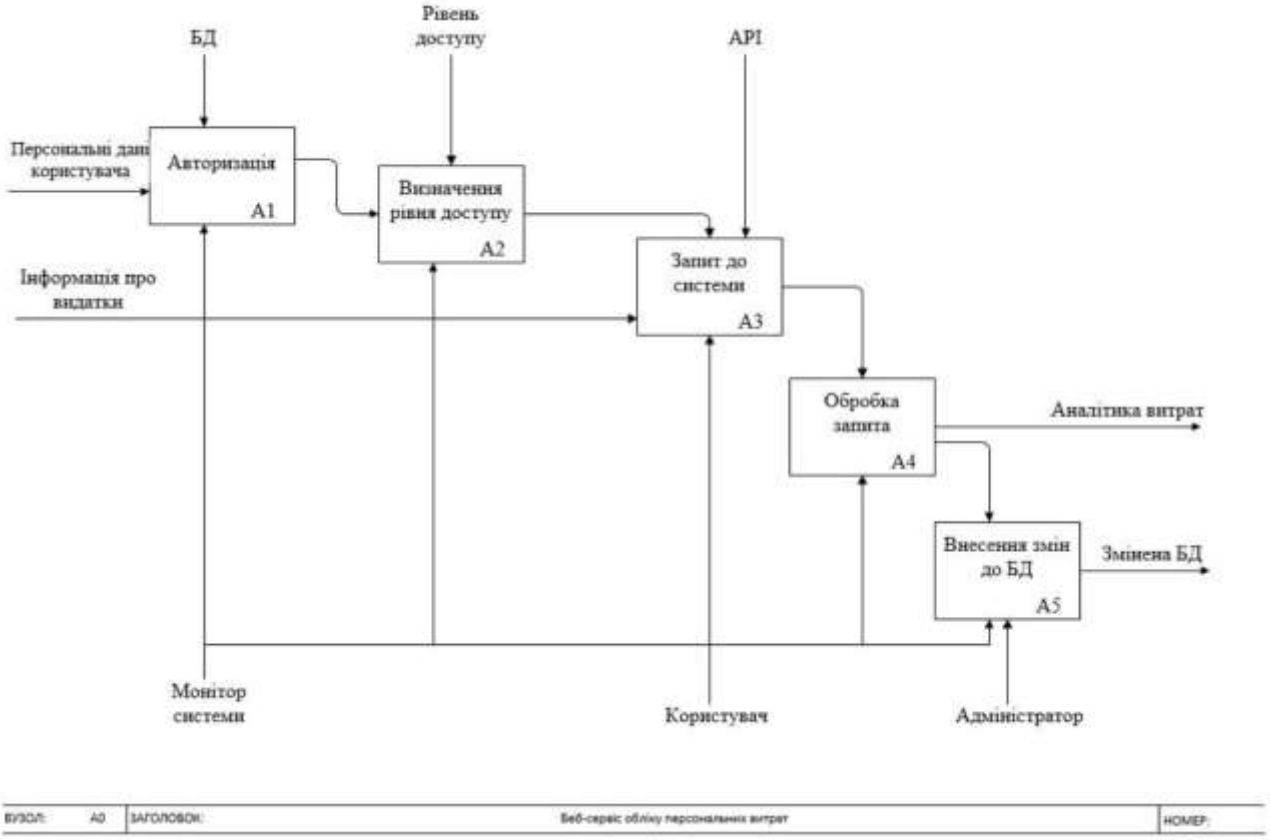


Рисунок 2.9 – Декомпозиція інформаційної системи

Варіанти використання системи

Діаграма варіантів використання інформаційної системи представлена на рисунку 2.10. З системою взаємодіє 4 актори: Адміністратор, незареєстрований користувач, зареєстрований користувач і зареєстрована компанія.

Адміністратор – це користувач, який після авторизації в системі, увійшов в кабінет адміністратора. Адміністратор може редагувати базу даних

користувачів для зміни їх рівня доступу до системи чи зміни персональних даних на їх вимогу. Також адміністратор модерує відгуки залишені користувачами і переглядає та відповідає на повідомлення користувачів. Незареєстрований користувач – це користувач, який не зареєстрований в системі. Він може відвідати сайт і зареєструватися або залишити сайт.

Зареєстрований користувач – це користувач, який після авторизації в системі, увійшов до кабінету користувача. Він має такі можливості: переглядати та редагувати свої персональні витрати, залишати відгуки до своїх покупок, залишати заявку на зміну персональних даних, відправляти повідомлення адміністратору, переглядати візуалізовану статистику персональних витрат.

Зареєстрована компанія – це користувач, який після авторизації в системі, увійшов до кабінету компанії. Він має такі можливості: залишати заявку на зміну персональних даних, відправляти повідомлення адміністратору, переглядати візуалізовану статистику витрат на основі даних всіх користувачі.

Варіанти використання API представлені на рисунку 2.11. З API взаємодіє 3 актори: незареєстрований користувацький додаток, зареєстрований користувацький додаток і БД ІС Веб-сервіс обліку персональних витрат.

Незареєстрований користувацький додаток – це додаток, який ще не зареєстрований в ІС, він має змогу лише пройти реєстрацію.

Зареєстрований користувацький додаток – це додаток, що зареєстрований в системі. Він має змогу авторизуватися і після успішної авторизації відправляти до БД інформаційної системи дані і отримувати свої раніше завантажені дані з БД.

База даних ІС – це сховище, яке реалізує збереження структурованих даних отриманих від користувацького додатку.

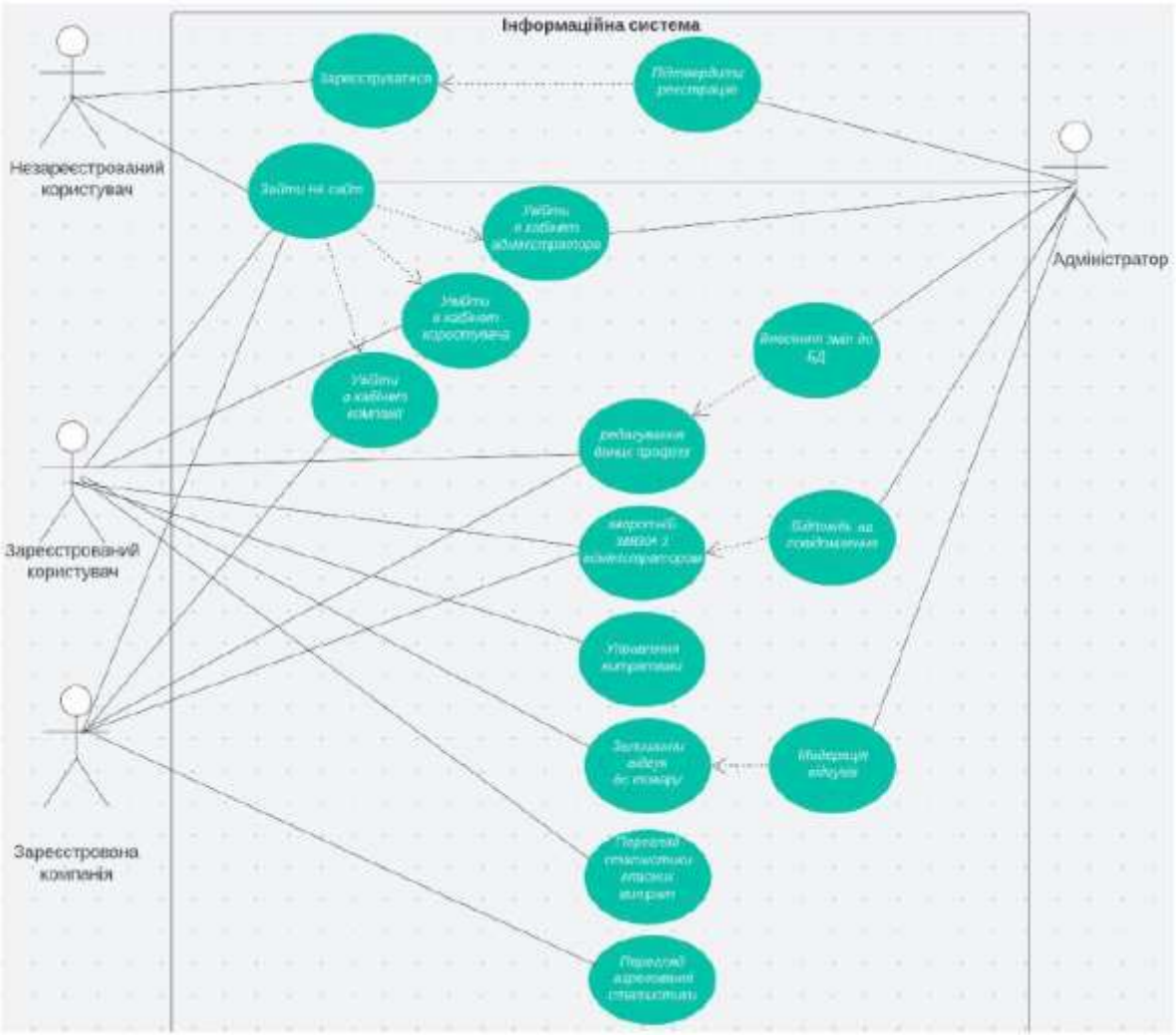


Рисунок 2.10 – Варіанти використання інформаційної системи

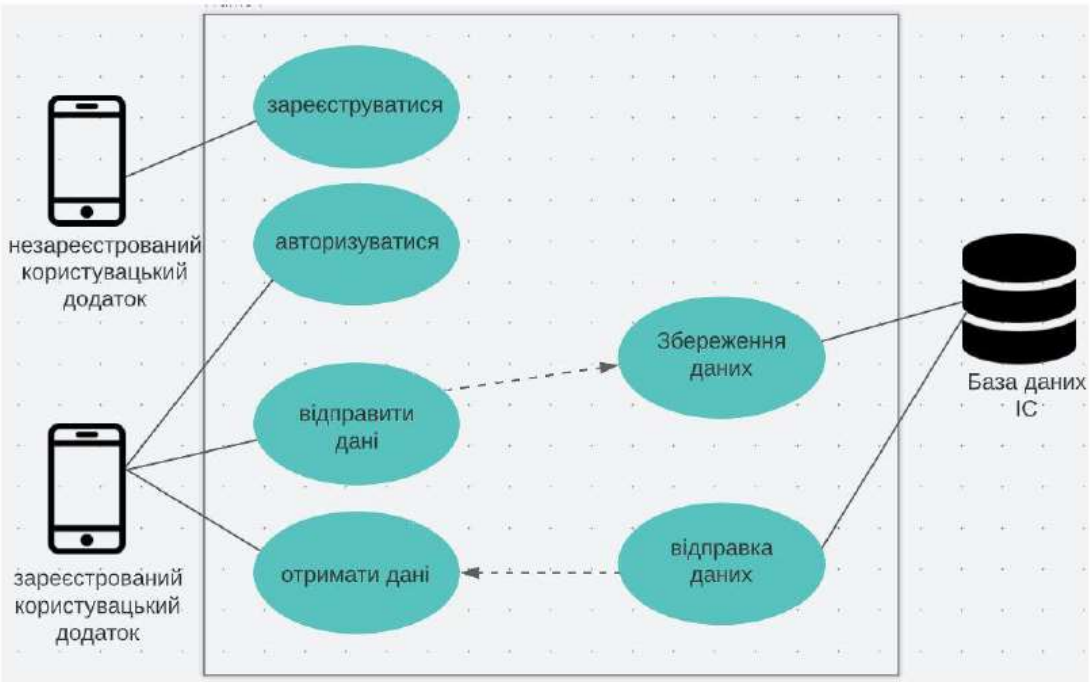


Рисунок 2.11 – Варіанти використання API

РОЗДІЛ 3 РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ АНАЛІЗУ ОСОБИСТИХ ВИТРАТ ТА СПОЖИВЧОГО ПОПИТУ

3.1. Symfony

Symfony це безкоштовний php фреймворк із відкритими сирцями та набір повторно використовуваних php компонентів [30]. Перший реліз опубліковано 18.10.2005, версія на момент написання – 6.2.

Symfony фреймворк складається із окремих та незалежних один від одного компонентів. Також до складу фреймворку входять набори пов’язаних між собою компонентів та файлів конфігурації – пакетів (bundle). Пакети реалізовують додатковий функціонал, зазвичай складніший ніж можуть реалізувати окремі компоненти, проте досить утилітарний щоб бути відокремленим у самостійний пакет.

Перед встановленням фреймворку треба переконатись, що система задовольняє технічні вимоги для роботи Symfony:

- PHP 8.1 або вище із увімкненими розширеннями Ctype, iconv, PCRE, Session, SimpleXML та Tokenizer;

- Менеджер пакунків Composer;

- Інтерфейс командної строки (CLI) Symfony у вигляді бінарного файлу symfony.exe

Для встановлення Symfony CLI необхідно спочатку встановити Scoop – інсталятор командної строки для Windows:

```
PS C:\Users\iamcr> Set-ExecutionPolicy RemoteSigned -Scope CurrentUser
PS C:\Users\iamcr> irm get.scoop.sh | iex
```

Рисунок 3.1 – Встановлення Scoop

Після виконання команд із Рисунку 3.1 необхідно використати scoop для встановлення Symfony CLI:

```
PS C:\Users\iamcr> scoop install symfony-cli
```

Рисунок 3.2 – Інсталяція Symfony CLI

У результаті виконання команди з Рисунку 3.2 буде встановлено Symfony CLI та зареєстровано бінарний файл у змінній оточення PATH, що надає можливість викликати Symfony CLI як команду "symfony" з будь якої локації системи. Далі необхідно створити директорію де буде розміщуватись проект та виконати наступну команду:

```
PS D:\sumdu\symfony> symfony new web_service
```

Рисунок 3.3 – Інсталяція Symfony

Функціонал веб застосунку або веб сервісу залежить від конкретного набору компонентів, з яких складається Symfony, тому буде виконано огляд компонентів специфічних для даної роботи та опущено огляд компонентів, що складають ядро фрейворку.

Doctrine ORM

Веб-сервіс використовує реляційну базу даних MySQL. Для відокремлення бізнес логіки веб сервісу від безпосередньої роботи із реляційною БД буде встановлено DoctrineBundle – пакунок, що забезпечує інтеграцію Doctrine ORM у проект на Symfony.

Doctrine ORM – об'єктно-реляційний проектор (ORM) для PHP, який базується на шарі абстракції доступу до БД (DBAL). Однією з ключових можливостей Doctrine є запис запитів до БД на власному об'єктно-орієнтованому діалекті SQL, званий DQL (Doctrine Query Language) і базується на ідеях HQL (Hibernate Query Language). [31] Встановлення пакунку відбувається шляхом виконання команди:

```
PS D:\sumdu\symfony\web_service> composer require doctrine/doctrine-bundle
```

Рисунок 3.4 – Інсталяція Doctrine ORM

EasyAdmin

Проект передбачає наявність персонального кабінету. Даний функціонал буде реалізовано за допомогою інструментів наданих пакунком EasyAdmin. EasyAdmin це пакунок відповідальний за генерацію фронтенду на підставі коду написаного на php. Основні можливості EasyAdmin:

- Створення інформаційної дошки (Dashboard) – вхідної точки персонального кабінету. Тут відображається головне меню, можливо налаштувати швидкий доступ до виконання частих задач, відображення графіків тощо;

- Організація доступу до даних через CRUD контролери;

- Контролювання зовнішнього вигляду фронтенду, окремих його елементів за допомогою php;

- Обробка подій;

- Обробка дій;

- Використання системи безпеки Symfony.

Встановлення пакунку виконується командою з Рисунку 3.5

```
PS D:\sumdu\symfony\web_service> composer require easycorp/easyadmin-bundle
```

Рисунок 3.5 – Встановлення EasyAdmin

API Platform

Основним джерелом даних системи є API, реалізація відбувається за допомогою API Platform – пакунку, що перепіс в систему, яку можна використовувати як окремо так і складі застосунку на Symfony. API Platform має багато можливостей, ключовими є:

- Автоматичне створення CRUD;
- Автоматична побудова документації;
- Пагінація результату;
- Великий набір фільтрів;
- Сортування;
- Додаткові налаштування авторизації, підтримка JWT токенів.

Додавання пакунку до складу веб сервісу зображено на Рисунку 3.6:

```
PS D:\sumdu\symfony\web_service> symfony composer require api
```

Рисунок 3.6 – Додавання API Platform

3.2. Amazon Web Services

Для використання хмарних сервісів Amazon необхідно мати зареєстрований акаунт. Взаємодіяти із хмарними сервісами можна через веб-інтерфейс та через API на програмному рівні. Мова програмування реалізації даного проекту – PHP, тому використовується AWS SDK for PHP, команда додавання пакету до проекту відображена на Рисунку 3.7:

```
PS D:\sumdu\symfony\web_service> composer require aws/aws-sdk-php
```

Рисунок 3.7 – Додавання AWS SDK for PHP

Через API надходять в тому числі файли, їх зберігання потребує створення хмарного сховища яке у інфраструктурі сервісів Amazon має назву S3. Після використання майстра створення нового екземпляру сховища, "bucket" - у термінології Amazon, отримано наступні атрибути:

- Ім’я сховища (bucket name);

- Регіон (Amazon має термін "Регіон доступності", географічна характеристика що визначає місцезнаходження датацентру, різні регіони мають невеликі відмінності у ціновій політиці використання сервісів).

Ці атрибути перенесені у конфігураційний файл фреймворку та використовуються пакунком AWS SDK для взаємодії із S3.

Інформація, що надходить через API, має бути збереженою до бази даних. Сервіс баз даних у інфраструктурі Amazon – RDS. Скориставшись майстром створення нового інстансу БД, на веб консолі управління AWS, отримано атрибути, необхідні для підключення системи до БД, значення атрибутів збережено до файлу конфігурації фреймворку.

3.3. Реалізація бази даних

Однією з переваг використання ORM систем є відсутність необхідності моделювати базу даних безпосередньо, більш того – немає залежності від типу БД, головне – щоб ORM підтримувала потрібний тип та щоб було відповідне налаштування у файлі конфігурації фреймворку. Натомість достатньо описати клас, де будуть перелічені усі поля таблиці включно із зв’язками з іншими таблицями. Один клас описує одну сутність, тобто таблицю. Коли усі таблиці описані – виконується команда що створює файл міграції та команда яка виконує інструкції із новоствореного файлу (Рисунок 3.8)

```
php bin/console make:migration && php bin/console doctrine:migrations:migrate
```

Рисунок 3.8 – Створення файлу міграції

У результаті отримано реляційну БД (Рисунок 3.9) та набір класів, що використовуються для доступу до даних.



Рисунок 3.9 – ER Діаграма

3.4. Короткий опис програмної реалізації

Нижче наведені діаграми класів створених під час розробки. Під час розробки моделі даних та створення класів що описують таблиці, були також створені класи-репозиторії, вони забезпечують бізнес логіку проекту для роботи з даними. Діаграма класів репозиторіїв наведена у Рисунку 3.10

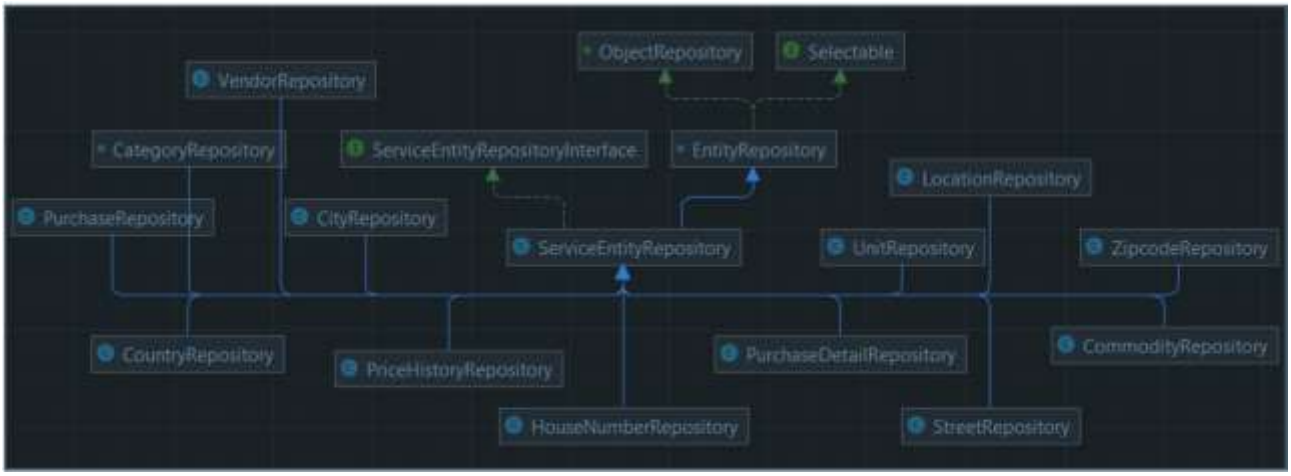


Рисунок 3.10 – Діаграма класів-репозиторіїв

Бізнес логіка відображення кабінету користувача реалізована у CRUD класах пакунку EasyAdmin, їх діаграма відображена на Рисунку 3.11

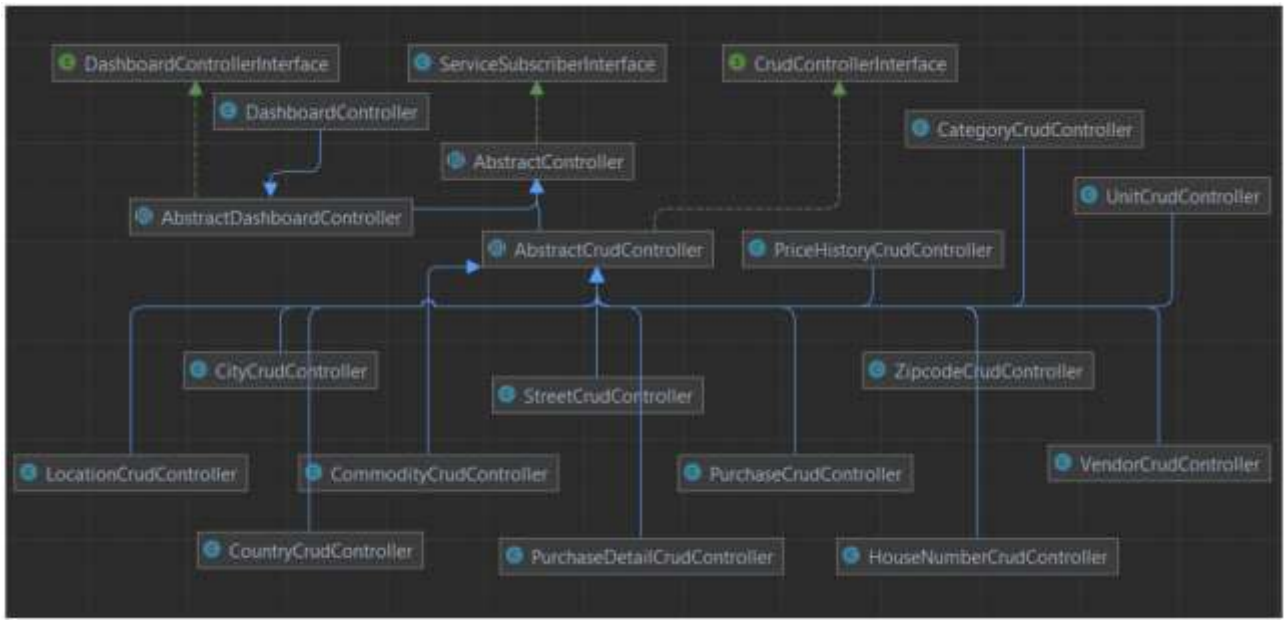


Рисунок 3.11 – діаграма CRUD класів EasyAdmin

API логіка забезпечена використанням PHP анотацій у класах сутностей. Механізм обробки та збереження даних, що надійшли через API реалізовано у класі App\State\PurchaseProcessor.

3.5. Функціональне тестування інформаційної системи

Останнім етапом роботи є тестування створеної системи. Тестування системи – це процес перевірки якості і надійності роботи створеного веб-сервісу, а також валідація відповідності поставленим вимогам і бізнес-цілям.

Функціональне тестування має на меті перевірити:

- Повноту і якість реалізованих функцій зазначених у вимогах до інформаційної системи;
- Реакцію системи на помилки користувача;
- Відсутність відмінностей між реальною поведінкою системи і очікуваною;
- Коректність роботи сервісу під навантаженням;
- Відсутність прихованих дефектів і проблем;
- Доступність сервісу для користувача з різних платформ;
- Повноту дій користувача;
- Складність системи для користувача, можливість працювати без додаткової підготовки

В процесі тестування мають бути перевірені такі підсистеми:

- Підсистема передачі даних через API
- Підсистема сховища структурованих даних
- Підсистема формування аналітичних даних
- Підсистема роботи з персональними видатками
- Підсистема авторизації
- Підсистема взаємодії елементів всієї системи

Тестування проводиться комплексно, тобто всі підсистеми тестуються одночасно на коректність взаємодії, вплив підсистем одна на одну.

Тестування підсистеми передачі даних через API

Тестування направлене на перевірку можливості користувача передавати дані зі сторонніх додатків до системи через API. Також перевіряється правильність і повнота переданих даних і правильне зв’язування даних з користувачем і можливість внесення некоректних даних.

Тестування підсистеми сховища структурованих даних

Тестування сховища структурованих даних направлене на перевірку правильності запису даних і коректну вибірку даних по запиту для роботи ІС.

Тестування підсистеми роботи з персональними витратами

Тест направлений на виявлення помилок при роботі з видатками, таких як неповне відображення або відсутність даних про видатки користувача в його кабінеті, не можливість вносити зміни чи залишати коментарі до товарів, а також не правильна робота фільтрів видатків.

Тестування підсистеми формування аналітичних даних

Метою тестування даної підсистеми є перевірка можливості користувачами формувати аналітичні діаграми і при цьому використовувати лише ті дані, до яких надано доступ, що обмежується поняттям "рівень доступу". Також буде перевірятися робота фільтра.

Тестування підсистеми авторизації

В ході даного тестування буде перевірятися коректність входу до системи під обліковими записами різних рівнів доступу до системи, також буде протестована функцію реєстрації.

Тестування підсистеми взаємодії елементів всієї системи

Даний тест буде проводитися з метою виявлення помилок при взаємодії підсистем.

Методика проведення тестування

Таблиця 3.1 – Тестування

№	Тест	Результат
1	Сценарій роботи з API	
	Протестувати можливість реєстрації в системі з користувацького додатку.	Була проведена успішна реєстрація в системі.
	Протестувати авторизацію і передачу даних з користувацького додатка в систему.	Авторизація пройшла успішно. Дані передались правильно і в повному обсязі.
	Перевірити можливість надсилання коректних не даних.	Дані не були відправлені. Користувач отримав повідомлення про необхідність внесення змін до даних.
	Перевірити правильність взаємодії API і фреймворка.	Під час передачі даних від користувача до системи через API помилок не виникло, тому взаємодія двох підсистем коректна.
2	Сценарій роботи зі сховищем	
	Протестувати коректне заповнення таблиць даними	Під час тестування API були реалізовані сценарії передачі і збереження даних. Сценарії були успішними тому БД зберігає дані правильно

	Авторизуватися в системі під адміністратором і протестувати можливість	Було здійснено вхід до системи під обліковим записом адміністратора і здійснено запит до БД на показ зареєстрованих в
	вибірки персональних даних з БД.	системі користувачів. Після запиту було автоматично сформовано системою список користувачів і виведено в кабінеті адміністратора.
	Авторизуватися під користувачем і перевірити правильність прив’язки даних про видатки з користувачем, а також коректність і повноту відображених даних.	Було здійснено вхід під обліковим записом користувача. Після відкриття вкладки "мої видатки" у кабінеті користувача був показаний список видатків, що належать даному користувачеві. Дані були відображені в повному обсязі, поля заповнені вірно.
3	Сценарій роботи з аналітикою видатків	
	Увійти до системи під обліковим записом користувача і сформувати аналітичну діаграму на основі випадково обраних значень з фільтру.	Вхід в систему було здійснено під користувачем. Після обрання, за допомогою фільтрів, даних було сформовано діаграму на основі коректних даних про персональні витрати.

	Увійти до системи під обліковим записом компанії і сформувати аналітичну діаграму на основі випадково обраних значень з фільтру.	Вхід в систему було здійснено під обліковим записом компанії. Після обрання, за допомогою фільтрів, даних було сформовано діаграму на основі коректних даних про витрати всіх користувачів.
4	Сценарій роботи з видатками	
	Авторизуватися в системі як користувач і перевірити коректність відображення даних користувача.	Вхід до системи здійснено під обліковим записом користувача. Дані, що відображаються в кабінеті належать даному користувачеві і представлені в повному обсязі.
	Перевірити роботу фільтра: наявність всіх пунктів і коректність відображення даних після застосування фільтру.	Фільтр має всі необхідні пункти. Після застосування фільтру були виведені коректні дані.
	Перевірка можливості внесення змін до даних про витрати.	Було відредаговано декілька записів, зміни були збережені і дані відобразились вже зі змінами.
	Перевірка можливості залишити коментар до покупок	Було прокоментовано декілька записів, коментарі збереглись.

	Перевірка можливості редагування персональних даних.	Дані були відредаговані користувачем, після чого був відправлений запит до адміністратора на зміну персональних даних. Після підтвердження адміністратора дані було змінено.
	Перевірка зворотного зв’язку з адміністратором системи.	З облікового запису користувача було відправлене повідомлення адміністратору. Повідомлення було отримане і надіслано відповідь. Відповідь також дійшла до користувача коректно
	Перевірка роботи фільтра для створення діаграм.	У фільтрі відображуються всі необхідні пункти
	Перевірка можливості створення аналітичних діаграм.	Після обрання, за допомогою фільтрів, даних було сформовано діаграму на основі коректних даних
5	Сценарій роботи з авторизацією	
	Перевірити коректність входу до системи під обліковими записами користувача, компанії і адміністратора.	В попередніх сценаріях робота даної процедури була протестована. Тестування були успішними.

	Перевірити можливість реєстрації користувача з сайту.	На сайті було відкрито і заповнено форму реєстрації. Адміністратор отримав запит на створення нового користувача. Після підтвердження запиту, в БД було створено нового користувача, а користувач зміг зайти до системи під своїми обліковими даними.
	Перевірити можливість реєстрації компанії.	На сайті було відкрито і заповнено форму реєстрації з приміткою компанія. Адміністратор отримав запит на створення нового користувача. Після підтвердження користувачем оплати послуги, в
		БД було створено нову компанію, а користувач зміг зайти до системи під своїми обліковими даними.
	Перевірити можливість реєстрації з вказанням некоректної інформації.	Після того як некоректно заповнена форма була відправлена, користувач отримав повідомлення, з проханням виправити дані. Адміністратор не отримав повідомлення про інцидент.

6	Тестування коректної взаємодії всіх підсистем
	Зважаючи на успішно проведені тестування всіх підсистем, які пов’язані між собою, то можемо зробити висновок, що всі підсистеми взаємодіють правильно.

Результат тестування:

- Функції реалізовані в системі відповідають вимогам до ІС в повній мірі;

- Система показала стійкість до помилок користувача;

- Відмінність між реальною поведінкою системи і очікуваною - відсутня;

- Відповідальність за коректну роботи сервісу під навантаженням в даній системі передана сервісу Amazon;

- Під час проведення тестування не було виявлено прихованих дефектів і проблем;

- Сервіс доступний для користувача з різних платформ;

- Веб-сервіс забезпечує всю повноту дій користувача, що заявлена в вимогах;

- Системи інтуїтивно зрозуміла, користувачі мають можливість працювати без додаткової підготовки.

ВИСНОВКИ

У даній роботі було визначено проблему необхідності аналізу споживчого попиту, її актуальності, та запропоновано рішення у вигляді веб-сервісу з збереження та аналітики персональних витрат.

Проведене ознайомлення із існуючими рішеннями в цій області, проаналізовані їх переваги та недоліки. Результатом проведеного аналізу стало підтвердження необхідності реалізації сервісу.

Здійснений огляд засобів вирішення поставленої задачі, сформовано структуру складових частин розроблюваної системи, визначено перелік технологій та інструментів необхідних для реалізації функціоналу сервісу. Розроблено архітектуру веб-сервісу та визначено набір інструментів для реалізації задачі.

У якості ядра системи було обрано веб-фреймворк Symfony, обґрунтовано необхідність розробки API, з підтримкою авторизації через oAuth2 та з підтримкою взаємодії із сервісами Amazon RDS та Amazon S3. Для отримання користувачами кінцевого результату і ознайомленням з прикладами аналітичних даних, реалізовано веб-інтерфейс з кабінетом користувача і кабінетом компанії.

На виході побудовано веб-сервіс здатний приймати дані про покупки через API та акумулювати їх у хмарному сховищі. Запроваджено систему надання аналітики отриманих даних. Сервіс корисний звичайним споживачам та привабливий для компаній зацікавлених у аналітиці споживчого ринку.

СПИСОК ЛІТЕРАТУРИ

1. Флінн Ш.М. Економіка для "чайників". К.: Діалект, 2008. 368 с.

2. Білецька Л.В., Білецький О.В., Савич В.І. Економічна теорія: навч. посіб., 2-ге вид. К.: Центр учбової літератури, 2009. 685 с.

3. Jiawei Han, Jian Pei, and Hanghang Tong, "Data Mining: Concepts and Techniques," Morgan Kaufmann, 2022. https://books.google.be/books?hl=uk&lr=&id=NR1oEAAAQBAJ&oi=fnd&pg=PP1&dq=++%E2%80%9CData++Mining++Concepts+++and+++Techniques&ots=_M8CLLvho4&sig=Cnjc0LJUnUeqpLFUAUDHFD76t-Y&redir_esc=y#v=onepage&q=%E2%80%9CData%20%20Mining%20%20Concepts%20%20%20and%20%20%20Techniques&f=false (accessed Dec. 02, 2022).

4. K. Vengatesan, E. Visuvanathan, A. Kumar, S. Yuvaraj, and P. Shivkumar Tanesh, "AN APPROACH OF SALES PREDICTION SYSTEM OF CUSTOMERS USING DATA ANALYTICS TECHNIQUES," JOURNAL Advances in Mathematics: Scientific Journal, vol. 9, no. 7, pp. 1857–8365, 2020, doi: 10.37418/amsj.9.7.70.

5. S. Cheriyan, S. Ibrahim, S. Mohanan, and S. Treesa, "Intelligent Sales Prediction Using Machine Learning Techniques," Proceedings - 2018 International Conference on Computing, Electronics and Communications Engineering, iCCECE 2018, pp. 53–58, Mar. 2019, doi: 10.1109/ICCECOME.2018.8659115.

6. Q. Zhang, H. Zhan, and J. Yu, "Car Sales Analysis Based on the Application of Big Data," Procedia Comput Sci, vol. 107, pp. 436–441, Jan. 2017, doi: 10.1016/J.PROCS.2017.03.137.

7. E. Fernandes, S. Moro, P. Cortez, F. Batista, and R. Ribeiro, "A data-driven approach to measure restaurant performance by combining online reviews with historical sales data," Int J Hosp Manag, vol. 94, p. 102830, Apr. 2021, doi: 10.1016/J.IJHM.2020.102830.

8. Z. P. Fan, Y. J. Che, and Z. Y. Chen, "Product sales forecasting using online reviews and historical sales data: A method combining the Bass model and sentiment

analysis," J Bus Res, vol. 74, pp. 90–100, May 2017, doi: 10.1016/J.JBUSRES.2017.01.010.

9. T. P. van Boeckel et al., "Global antibiotic consumption 2000 to 2010: an analysis of national pharmaceutical sales data," Lancet Infect Dis, vol. 14, no. 8, pp. 742–750, Aug. 2014, doi: 10.1016/S1473-3099(14)70780-7.

10. M. Pivette, J. E. Mueller, P. Crépey, and A. Bar-Hen, "Drug sales data analysis for outbreak detection of infectious diseases: A systematic literature review," BMC Infect Dis, vol. 14, no. 1, pp. 1–14, Nov. 2014, doi: 10.1186/S12879-014-0604-2/TABLES/3.

11. Y. T. Negash, T.-D. Bui, M. Piekut, and K. Piekut, "Changes in Patterns of Consumer Spending in European Households," Sustainability 2022, Vol. 14, Page 12794, vol. 14, no. 19, p. 12794, Oct. 2022, doi: 10.3390/SU141912794.

12. G. Lei, K. Xiao, F. Cui, X. Luo, and M. Dai, "A Parallel Algorithm of Association Rules Applicable to Sales Data Analysis," Recent Advances in Computer Science and Communications, vol. 14, no. 3, pp. 916–925, Mar. 2020, doi: 10.2174/2666255813666200304144112.

13. K. Zhao, R. Sun, C. Deng, L. Li, Q. Wu, and S. Li, "Visual Analysis System for Market Sales Data of Agricultural Products," IFAC-PapersOnLine, vol. 51, no. 17, pp. 741–746, Jan. 2018, doi: 10.1016/J.IFACOL.2018.08.107.

14. K. Singh and R. Wajgi, "Data analysis and visualization of sales data," IEEE WCTFTR 2016 - Proceedings of 2016 World Conference on Futuristic Trends in Research and Innovation for Social Welfare, Oct. 2016, doi: 10.1109/STARTUP.2016.7583967

15. Остервальдер О., Піньє І. Побудова бізнес-моделей. М.: Альпіна Паблішер, 2020. 288 с.

16. Базелевич В.Д. Економічна теорія: Політекономія: підручник, 6-е вид. К.: Знання-Прес, 2007. 719 с.

17. Мілославський І.О. Веб-сервіс збереження зображень розрахункових документів з RESTful api : випускна робота. Суми, 2019. 45 с.

18. FAQ about Google Trends data. URL: https://support.google.com/trends/answer/4365533?hl=en (дата звернення: 01.11.2022)

19. Державна служба статистики України. URL: https://ukrstat.gov.ua/ (дата звернення: 03.11.2022)

20. Верховна Рада України. Документ z0220-16. URL: https://zakon.rada.gov.ua/laws/show/z0220-16#Text (дата звернення 11.12.2022)

21. Вікіпедія. REST. URL: https://uk.wikipedia.org/wiki/REST (дата звернення: 04.11.2022)

22. Вікіпедія. SOAP. URL: https://en.wikipedia.org/wiki/SOAP (дата звернення: 04.11.2022) 23. Polák M., Holubová I. REST API management and evolution using MDA: ACM International Conference Proceeding Series, vol. 13-17-July-2015, pp. 102–109, Jul. 2015, doi: 10.1145/2790798.2790820.

24. Офіційний сайт протоколу Oauth 2.0 URL: https://oauth.net/2/ (дата звернення: 04.11.2022)

25. Amazon RDS. URL: https://aws.amazon.com/rds/ (дата звернення: 04.11.2022)

26. Amazon S3. URL: https://aws.amazon.com/s3/ (дата звернення: 04.11.2022)

27. Вікіпедія. Сторінка-вітрина. URL: https://uk.wikipedia.org/wiki/%D0%A1%D1%82%D0%BE%D1%80%D1%96%D0%BD%D0%BA%D0%B0-%D0%B2%D1%96%D1%82%D1%80%D0%B8%D0%BD%D0%B0(дата звернення: 07.11.2022)

28. Symfony UX chart.js URL: https://symfony.com/bundles/ux-chartjs/current/index.html (дата звернення: 07.11.2022)

29. W3schools. Chart.js. URL: https://www.w3schools.com/js/js_graphics_chartjs.asp (дата звернення: 07.11.2022)

30. Вікіпедія. Symfony. URL: https://uk.wikipedia.org/wiki/Symfony (дата звернення 11.12.2022)

31. Вікіпедія. Doctrine (PHP). URL: https://uk.wikipedia.org/wiki/Doctrine_(PHP) (дата звернення 11.12.2022)

ДОДАТКИ

ДОДАТОК А.

ЛІСТИНГ КЛАСУ APP\STATE\PURCHASEPROCESSOR

```
<?php
namespace App\State;
use ApiPlatform\Metadata\DeleteOperationInterface;
use ApiPlatform\Metadata\Operation;
use ApiPlatform\State\ProcessorInterface;
use App\Dto\CommodityDto;
use App\Dto\LocationDto;
use App\Entity\Category;
use App\Entity\City;
use App\Entity\Commodity;
use App\Entity\Country;
use App\Entity\HouseNumber;
use App\Entity\Location;
use App\Entity\PriceHistory;
use App\Entity\Purchase;
use App\Entity\PurchaseDetail;
use App\Entity\Street;
use App\Entity\Unit;
use App\Entity\Vendor;
use App\Entity\Zipcode;
use Doctrine\ORM\EntityManagerInterface;
class PurchaseProcessor implements ProcessorInterface
{
public function __construct(
private ProcessorInterface $persistProcessor,
     private ProcessorInterface $removeProcessor,
private EntityManagerInterface $em
)
{
}
private function resolveLocation(LocationDto $data) : ?Location
{
if(!($country = $this->em->getRepository(Country::class)
->findOneBy(['isoCode' => $data->country])))
{
$country = (new Country())->setIsoCode($data->country);
$this->em->persist($country);
}
if(!($city = $this->em->getRepository(City::class)
->findOneBy(['name' => $data->city])))
{
$city = (new City())->setName($data->city)->setCountry($country);
$this->em->persist($city);
}
if(!($zipcode = $this->em->getRepository(Zipcode::class)
->findOneBy(['zipcode' => $data->zipcode])))
{
$zipcode = (new Zipcode())->setZipcode($data->zipcode)-
>setCity($city);
```

```
$this->em->persist($zipcode);
    }
if(!($street = $this->em->getRepository(Street::class)
->findOneBy(['name' => $data->street])))
{
$street = (new Street())->setName($data->street)->setCity($city);
$this->em->persist($street);
}
if(!($house = $this->em->getRepository(HouseNumber::class)
->findOneBy(['houseNumber' => $data->houseNumber])))
{
$house = (new HouseNumber())->setHouseNumber($data->houseNumber)-
>setStreet($street);
$this->em->persist($house);
}
if(!($location = $this->em->getRepository(Location::class)
->findOneBy([
'country' => $country,
'city' => $city,
'zipcode' => $zipcode,
'street' => $street,
'houseNumber' => $house
])))
{
$location = (new Location())
->setCountry($country)
->setCity($city)
->setZipcode($zipcode)
->setStreet($street)
    ->setHouseNumber($house)
;
$this->em->persist($location);
}
return $location;
}
private function resolveUnit(string $name) : Unit
{
if(!($unit = $this->em->getRepository(Unit::class)->findOneBy(['name'
=> $name])))
{
$unit = (new Unit())->setName($name);
$this->em->persist($unit);
$this->em->flush();
}
return $unit;
}
private function resolveCategory(string $name, ?int
$parentCategoryId) : Category
{
if(!($category = $this->em->getRepository(Category::class)-
>findOneBy(['name' => $name])))
{
$category = (new Category())->setName($name);
$this->em->persist($category);
```

```
if(!empty($parentCategoryId))
{
if(!($parentCategory = $this->em->getRepository(Category::class)-
>findOneBy(['id' => $parentCategoryId])))
    {
throw new \Exception("Cant resolve parent category with id
{$parentCategoryId}");
}
else
{
$category->setParentId($parentCategory);
}
}
$this->em->flush();
}
return $category;
}
private function resolveCommodity(array $data, Category $category) :
Commodity
{
if(!($commodity = $this->em->getRepository(Commodity::class)-
>findOneBy(['name' => $data['name']])))
{
$commodity = (new Commodity())
->setName($data['name'])
->setPrice($data['price'])
->setCategory($category)
;
$priceHistory = (new PriceHistory())->setCommodity($commodity)-
>setPrice($data['price']);
$this->em->persist($commodity);
$this->em->persist($priceHistory);
}
else
    {
if($commodity->getPrice() !== $data['price'])
{
$priceHistory = (new PriceHistory())
->setPrice($data['price'])
->setCommodity($commodity);
$this->em->persist($priceHistory);
}
}
return $commodity;
}
private function resolveVendor(string $name, Location $location) :
Vendor
{
if(!($vendor = $this->em->getRepository(Vendor::class)
->findOneBy(['name' => $name, 'location' => $location])))
{
$vendor = (new Vendor())->setName($name)->setLocation($location);
$this->em->persist($vendor);
}
```

```
return $vendor;
}
/**
* @param Purchase $data
* @param Operation $operation
* @param array $uriVariables
* @param array $context
     * @return mixed
* @throws \Exception
*/
public function process($data, Operation $operation, array
$uriVariables = [], array $context = [])
{
if ($operation instanceof DeleteOperationInterface) {
return $this->removeProcessor->process($data, $operation,
$uriVariables, $context);
}
$location = $this->resolveLocation($data->getVendor()->locationData);
$data->getVendor()->setLocation($location);
/**
* @var CommodityDto $item
*/
foreach($data->commodityData as $item)
{
$data->addPurchaseDetail(
(new PurchaseDetail())
->setCommodity($this->resolveCommodity($item, $this-
>resolveCategory($item['category'], $item['parentCategoryId'] ??
null)))
->setPrice($item['price'])
->setUnit($this->resolveUnit($item['unit']))
->setVolume($item['volume'])
);
    }
$result = $this->persistProcessor->process($data, $operation,
$uriVariables, $context);
return $result;
}
    }
```