МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Тернопільський національний економічний університет
Факультет комп'ютерних інформаційних технологій
Кафедра комп'ютерних наук

ПОЛЯРУШ Олег Віталійович

Математичне та програмне забезпечення для задачі тайм-менеджменту/ Mathematical tools and software for the task of time-management

спеціальність: 8.05010301 - Програмне забезпечення систем
магістерська програма - Програмне забезпечення систем

Магістерська робота

Виконав студент групи ПЗСм-21
О. В. Поляруш

_____

Науковий керівник:
к.т.н., доцент СТРУБИЦЬКА І.П.

_____

Магістерську роботу допущено
до захисту:

"___" _____ 20___ р.

Завідувач кафедри
_____ **А. В. Пукас**

ТЕРНОПІЛЬ - 2016

РЕЗЮМЕ

Дипломна робота на тему: «Математичне та програмне забезпечення для задач тайм менеджмент» складається з трьох розділів та виконана на 104 сторінках.

Об'єктом дослідження є процеси генерації рекомендацій подій для системи тайм менеджмент.

Предметом дослідження є методи та програмні засоби побудови рекомендації для системи вирішення задач тайм менеджмент.

Методи дослідження ґрунтуються на застосуванні системного аналізу, функціонального моделювання, семантичних мереж, теорії множин, теорії графів, теорії алгоритмів та об'єктно-орієнтованого проектування.

Наукова новизна одержаних результатів. Удосконалено інформаційну технологію генерації рекомендацій на основі використання алгоритмів ранжування та самонавчання, для покращення видачі результатів.

Практичне значення одержаних результатів. Запропоновані методи і програмні засоби забезпечили вирішення завдань підвищення ефективності формування плану, що сприяє збільшенню ефективності та якості виконання покладених завдань. Створений програмний продукт дозволяє будувати плани на основі рекомендацій базуючись на засобах виявлення зав'язків між елементами.

Ключові слова: рекомендації, ранжування, об'єктно-рольове моделювання, самонавчання, тайм менеджмент.

SUMMARY

Thesis: "Mathematical tools software for the task of the time-management, consists of three sections and is made on 104 pages.

The object of research is the process of generating recommendation in time management systems.

The subject of research are methods and software for implementing recommendationsystem for the time management systems".

Methods based on using of systems analysis, functional modeling, semantic networks, set theory, graph theory, theory of algorithms and object-oriented design.

Scientific novelty of results. Improved information technology of recommendation generation based on ranking algorithms, which contributed to more efficient decision-making.

The practical significance of results. The methods and software provided meet the challenges of improving the efficiency of forming a plan that increases the efficiency and quality of discharge. The software product allows to plan on the basis of recommendations based on the detection of ties between elements.

Keywords: recommendations, rankings, object-role modeling, learning, time management.

ЗМІСТ

ВСТУП

Актуальність теми.

Швидкий розвиток інформаційних технологій призводить до перенасиченості інформації та попадання під вплив непотрібної інформації. Це призводить до втрати концентрації та розсіяності сучасної людини, адже перенасиченість інформаційного простору різними елементами, які забирають наш час відводячи увагу від поставлених цілей, що призводить до прокрастинації.

Одним із способів вирішення цієї проблеми є чітко сформовані плани виконання задач та контроль їх виконання, що дасть змогу користувачу як найкраще заповнити свій час необхідним та відштовхуватись від поставлених рамок. Також виникає проблема з формуванням плану та фільтруванням даних для побудови планів, побудови системи на базі вподобань користувача по збору відповідної інформації за критеріями для користувача, що дасть змогу як найкраще підібрати необхідну інформацію. У форматі вибору того чим можна зпавнити календар не витрачаючи час на пошук інформації згідно інтересів. Виникає потреба у віддаленому сховищі, яке могло б надавати можливість доступу у найзручніший момент.

Мета і задачі дослідження.

Метою роботи є розробка програмної системи для вирішення задач тайм менеджменту із використання можливості генерації рекомендацій, які б забезпечили підвищення формування планів.

Для досягнення поставленої мети у роботі потрібно розв’язати наступні задачі:

Об'єкт дослідження: процеси генерації планів та використання системи як фільтру інформації згідно потреб користувача.

Предметом дослідження: підоходи до побудови системи із формування рекомндацій із викристаннями алгоритмів смовнавчання.

Методи дослідження: для досягнення поставленої мети використано методи інтелектуального аналізу даних, також викристано прграмні засоби Numpy та Pandas.

Наукова новизна одержаних результатів.

В роботі удосконалено механізм генерації рекомендацій на базі методу опорних векторів із використанням механізму самонавчання, який дає можливість корегувати та покращувати процедуру фільтрації із використанням тестової вибірки, або поправками користувача.

Практичне значення одержаних результатів

Полягає у можливості автоматичного аналізу інформації для побудови планів.

Апробація результатів.

Основні положення магістерського дослідження апробовані на VI Всеукраїнській школі-семінарі молодих вчених і студентів «Сучасні комп'ютерній інформаційні технології», що проходила 20-21 травня 2016 року у м. Тернополі на базі Тернопільського національного економічного університету.

Публікації.

Поляруш О.В. Особливості використання Behavioral targeting для задачі тай менеджменту. Поляруш О. В., Струбицька І. П.. // Матеріали VI Всеукраїнської школи-семінару молодих вчених і студентів «Сучасні комп'ютерні інформаційні технології», АСІТ'2016 – Тернопіль: ТНЕУ, 2016. − с. 148-149.

РОЗДІЛ 1
ОСОБЛИВОСТІ РЕАЛІЗАЦІЇ СИСТЕМ ДЛЯ ВИРІШЕННЯ ЗАДАЧІ TIME MANAGEMENT

1.1 Коротка характеристика об’єкту управління

На сьогоднішній день люди часто стикаються з проблемою надлишкової інформації, що приводить до інформаційного перевантаження. Рекламні компанії та ресурс, які забираються у нас час, змушують розсіюватись між задачами. Наслідком цього є втрата часу на переключення між ними. Система планування часу дасть змогу як найкраще оптимізувати свій графік, розподілити задачі та час на їх виконання, а також полегшить пошук самої інформації маючи визначені певні критерії.

Система планування на базі інтересів користувача – це програмне забезпечення передбачене для використання великого сектору користувачів. Цей програмний продукт надає можливість користувачеві налаштовувати автоматизоване планування графіка дня на базі його вподобань. Також система розпізнає та сортує по різним категоріям, вказані користувачем, нові вхідні дані та показує їх як рекомендації. Попередня історія виборів аналізується та будується матриця вподобань для подальшої генерації можливих варіантів. Генерація матриці вподобань відбувається наступним чином – при першому запуску системи буде надана можливість користувачеві вибрати найпопулярніші категорії для формування початкових даних для матриці, також представлена можливість ввести власні слова, або словосполучення, котрі будуть слугувати шаблоном для пошуку та вибірки даних для формування рекомендацій. Система буде вести спостереження за подальшим виборами користувача, з метою автоматизованого покращення матриці вподобань користувача. Початкові дані для формування вибору користувача, вносяться адміністраторами системи. Система взаємодії із модулем формування даних,

який виконує задачу збору інформації із різного роду джерел. Для формування рекомендацій необхідно побудувати рекомендаційну систему.

Рекомендаційна система — підклас системи фільтрації інформації, яка будує рейтинговий перелік об'єктів (фільми, музика, книги, новини, веб-сайти), яким користувач може надати перевагу. Для цього використовується інформація з профілю користувача. Існують дві основні стратегії створення рекомендаційних систем: фільтрація вмісту і колаборативна фільтрація.

При фільтрації вмісту створюються профілі користувачів і об'єктів:

- профілі користувачів можуть містити демографічну інформацію або відповіді на певний набір питань.
- профілі об'єктів можуть містити назви категорій, назви подій, час виконання тощо. Або якусь іншу інформацію в залежності від типу об'єкта.

При колаборативній фільтрації використовується інформація про поведінку користувачів у минулому. Наприклад, інформація про придбання або оцінки. В цьому випадку не має значення з якими типами об'єктів ведеться робота, але при цьому можна брати до уваги неявні характеристики, які складно було б врахувати при створенні профілю. Головна проблема цього типу рекомендаційних систем — «холодний старт»: відсутність даних про користувачів чи об'єкти, які нещодавно з'явились у системі.

У процесі роботи рекомендаційні системи збирають дані про користувачів, використовуючи поєднання явних і неявних методів.

- Приклади явного збору даних:
  - користувач оцінює запропонований об'єкт за диференційованою шкалою;
  - користувач ранжує групу об'єктів від найкращого до найгіршого;
  - користувач вибирає кращий з двох запропонованих об'єктів;
  - користувачу пропонують створити список його улюблених об'єктів.
- Приклади неявного збору даних:
  - спостереження за тим, що користувач оглядає в інтернет-магазинах або базах даних іншого типу;
  - ведення записів про поведінку користувача онлайн;

- відстеження вмісту комп'ютера користувача;

1.2 Аналіз існуючих методів для вирішення поставленої задачі.

Найпопулярніші техніки побдуови системи генерації рекомендацій, базуються на методах взаємодії із матрицями клафікацій, які дають змогу формувати певні закони розподілу множини. Процес побудови критерію бузєтсья на методеці розіпзнавання образів, із використанням ядрового трюку.

Загальна задача розпізнавання образів полягає у знаходженні та вивченні основних типів відношень (наприклад, кластерів, ранжування, головних компонент, кореляцій, класифікацій) у наборах даних. Для багатьох алгоритмів, які розв'язують ці задачі, дані в сирому представленні мають бути явним чином перетворено на представлення у вигляді векторів ознак через визначене користувачем відображення ознак: на противагу цьому ядрові методи вимагають лише вказаного користувачем ядра, тобто, функції подібності над парами точок даних у сирому представленні.

Для вирішення задач розпізнавання образів використовуються ядрові методи [1]. Ядрові методи завдячують своєю назвою застосуванню ядрових функцій, які дозволяють їм діяти в неявному просторі ознак високої вимірності навіть без обчислення координат даних у цьому просторі, натомість просто обчислюючи внутрішній добуток зображень всіх пар даних у цьому просторі ознак. Ця операція часто має меншу обчислювальну складність, ніж явне обчислення координат. Цей підхід називають ядровим трюком.

Ядровий трюк уникає явного відображення, потрібного для того, щоб лінійні алгоритми навчання вдосконалювались нелінійної функції, або межі рішень. Для всіх x та x'у вхідному просторі X певні функції $k(x,x')$ може бути виражено як внутрішній добутків іншому просторі V. Функцію $k: X \times X \to R$ часто називають ядром або ядровою функцією. Слово «ядро»

використовується в математиці для позначення зважувальної функції зваженої суми або інтегралу.

Деякі задачі в машинному навчанні мають додаткову структуру, ніж просто довільна зважувальна функція *k.* Обчислення буде простіши, якщо ядро можна записати в вигляді «відображення ознак» $\varphi: X \to V$, яке задовільняє

$$k(x, x') = \langle \varphi(x), \varphi(x') \rangle v. \tag{1.1}$$

Ключовим обмеженням є те, що $\langle .,. \rangle$ мусить бути власним внутрішнім добутком. З іншого боку, явне представлення φ не є необхідним, поки V є простором з внутрішнім добутком. Ця альтернатива випливає з теореми Мерсера: неявно визначена функція φ існує тоді, коли простір X може бути споряджено придатною мірою, яка забезпечувала би, щоби функція k задовольняла умову Мерсера.

Теорема Мерсера споріднена з узагальненням того наслідку з лінійної алгебри, що пов'язує внутрішній добуток із будь-якою додано-означеною матрицею. Фактично, умову Мерсера може бути зведено до цього простішого прояву. Якщо ми оберемо як нашу міру лічильну міру $\mu(T) = |T|$ для всіх $T \subset X$, яка лічить число точок всередині множини T, то інтеграл у теоремі Мерсера зводиться до підсумовування (1.2)

$$\sum_{i=1}^{n} \sum_{j=1}^{n} k(x_i, x_j) c_{ci} c_j \geq 0. \tag{1.2}$$

Якщо це підсумовування виконується для всіх скінченних послідовностей точок $(x_1 \ldots x_n)$ і всіх варіантів вибору n дійснозначних коефіцієнтів $(c_1 \ldots c_n)$ то функція k задовольняє умову Мерсера. Деякі алгоритми, які залежать від довільних взаємозв'язків у рідному просторі X, фактично могли би мати лінійну інтерпретацію за іншої постановки: φ області. Лінійна інтерпретація дає нам прояснення алгоритму. Проте, немає потреби часто під час обчислень

обчислювати φ безпосередньо, як у випадку методу опорних векторів. Деякі дослідники посилаються на цю раціоналізацію часу як на головну перевагу. Дослідники також використовують її для обґрунтування сенсу та властивостей наявних алгоритмів. Теоретично, матриця Грама $K \in \mathbb{R}^{n \times n}$ по відношенню до $\{x_1 \ldots x_n\}$, яку іноді також називають «ядровою матрицею», мусить бути додатнонапівозначеною. Емпірично, для евристик машинного навчання варіанти вибору функції *k*, які не задовольняють умову Мерсера, все ще можуть працювати прийнятно, якщо *k* щонайменше наближує інтуїтивне уявлення про подібність. Незалежно від того, чи є *k* мерсеровим ядром, *k* все одно може називатися «ядром». Якщо ядрова функція *k* є також і функцією коваріації як при застосуванні в ґаусових процесах, то матриця Грама *K,* може також називатися коваріаційною матрицею. Припустімо, що K є квадратною матрицею. Тоді $K^T K$ є додатнонапівозначеною матрицею.

До алгоритмів, здатних працювати з ядрами, належать метод головних компонент, канонічно-кореляційний аналіз, гребенева регресія, спектральне кластерування, лінійні адаптивні фільтри.

Канонічно-кореляційний аналіз (CCA) являє собою спосіб надання сенсу матриць крос-коваріацій. Якщо ми маємо два вектори X = (X1, ..., Xn) і Y = (y1, ..., Ym) випадкових величин, та існують кореляції між змінними, канонічний кореляційний аналіз буде знаходити лінійні комбінації Xi і Yj, які мають максимальну кореляцію між собою. Практично всі значення параметричних випробувань, які найбільше зустрічаються, можна розглядати як окремі випадки аналізу канонічної кореляції, що є спільною процедурою для дослідження взаємозв'язку між двома наборами змінних. Роботу канонічно-кореляційний аналізу можна охарактеризувати так: визначено два стовпця вектора $X = (x_1, \ldots, x_n)$ і $Y = (y_1, \ldots, y_n)$ випадкових величин з кінцевими другими моментами, можна визначити крос-коваріацію(1.3)

$$E_{XY} = cov(X, Y), \tag{1.3}$$

щоб для $n \times m$ матриця чиї (i,j) запис є коваріація $cov(x_i, y_i)$.

На практиці ми оцінюємо коваріаційнну матрицю, грунтуючись на оцифрованих даних з X і Y (тобто з пари матриць даних).

Канонічний кореляційний аналіз шукає вектори a та b , як випадкові величини $\alpha`X$ і $\beta'Y$, щоб максимізувати кореляцію (1.4)

$$p = corr(\alpha`X, \beta'Y) \tag{1.4}$$

Випадкові величини $U=\alpha`X$ і $V = \beta'Y$ є першою парою канонічних змінних. Тоді шукаються вектори, що максимізують ту ж кореляції з тим обмеженням де вони повинні бути корельовані з першою парою канонічних змінних; це дає другу пару канонічних змінних. Ця процедура може бути продовжена до $min\{m,n\}$ раз.

Розрахунок відбувається наступним принципом:

Нехай $\Sigma_{XX} = cov(X,X)$ і $\Sigma_{YY} = cov(Y,Y)$тоді параметр для максимізації буде (1.5)

$$p = \frac{\alpha`\Sigma_{XY} b}{\sqrt{\alpha`\Sigma_{XX}\alpha}\sqrt{\mathrm{b}`\Sigma_{YY} b}} \tag{1.5}$$

Першим кроком буде визначення базису змін $c = \Sigma_{XX}^{1/2}\alpha$, $d = \Sigma_{YY}^{1/2}\beta$ в результаті ми отримаємо (1.6)

$$p = \frac{c`\Sigma_{XX}^{1/2}\Sigma_{XY}\Sigma_{XX}^{-1/2} d}{\sqrt{\mathrm{c}`\mathrm{c}}\sqrt{\mathrm{d}`\mathrm{d}}} \tag{1.6}$$

Відносно нерівності Коші-Шварца отримаємо (1.7)

$$\left(c`\Sigma_{XX}^{-\frac{1}{2}}\Sigma_{XY}\Sigma_{YY}^{-\frac{1}{2}}\right)d \leq \left(c`\Sigma_{XX}^{-\frac{1}{2}}\Sigma_{XY}\Sigma_{YY}^{-\frac{1}{2}}\Sigma_{YY}^{-\frac{1}{2}}\Sigma_{YX}\Sigma_{YY}^{-\frac{1}{2}}c\right)(d`d)^{\frac{1}{2}},$$

$$p = \frac{(c`\Sigma_{XX}^{-\frac{1}{2}}\Sigma_{XY}\Sigma_{YY}^{-1}\Sigma_{YX}\Sigma_{XX}^{-\frac{1}{2}}c)^{1/2}}{\sqrt{c`c}} \qquad (1.7)$$

Це вірно, якщо вектор d і $\Sigma_{XX}^{-1/2}\Sigma_{YX}\Sigma_{XX}^{-1/2}$с є колінеарн. Крім того, максимум кореляції досягається, якщо c є власний вектор з максимальним власним значенням для матриці $\Sigma_{XX}^{-1/2}\Sigma_{YX}\Sigma_{YY}^{-1}\Sigma_{YX}\Sigma_{XX}^{-1/2}$. Наступні пари знайдені за допомогою власних значень відбувають завдяки зменшення величин. Ортогональність забезпечується симетрією кореляційних матриць.

Недоліком використання даного метода є нестабільність до пустот та шумів у вхідних даних, що потребує перевірку та на вході. Також може виникнути перенавчання на етапі опрацювання даних для тренування.

Поширеним методом для роботи з ядрами є метод головних компонент. Він є одним з найбільш розповсюджених методів факторного аналізу. МГК широко представлений у літературних джерелах, звернувшись до яких можна одержати відомості про метод головних компонент з різним ступенем деталізації й математичної строгості. Серед інших подібних методів, що дозволяють узагальнювати значення елементарних ознак, МГК виділяється простою логічною конструкцією, й у той же час на його прикладі стають зрозумілими загальна ідея й цілі численних методів факторного аналізу. Метод головних компонент дає можливість по m — числу вихідних ознак виділити m головних компонент, або узагальнених ознак [14]. Простір головних компонент ортогональний. Математична модель методу головних компонент базується на логічному припущенні, що значення множини взаємозалежних ознак породжують деякий загальний результат.

Розв'язування завдання методом головних компонент зводиться до поетапного перетворення матриці вихідних даних X.

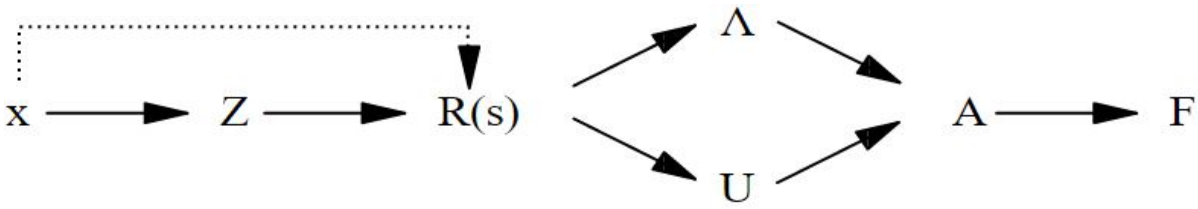

Рис.1.2. Схема математичних перетворень

На малюнку позначення: *X* — матриця вихідних даних розмірністю *n*×*m*, де *n* число об'єктів спостереження, *m* число елементарних аналітичних ознак,*Z* — матриця центрованих і нормованих значень ознак,*R* — матриця парних кореляцій. Λ — діагональна матриця власних (характеристичних) чисел. *A* — матриця факторного відображення, її елементи a_{rj} — вагові коефіцієнти. Спочатку *A* має розмірність *m*m* — за кількістю елементарних ознак *Xj*, потім в аналізі залишається r найвагоміших компонент, *r* ≤ m. Обчислюють матрицю *A* по відомих даним матриці власних чисел Λ і нормованих власних векторах *V* за формулою (1.8)

$$A = V\Lambda 1/2, \tag{1.8}$$

F — матриця значень головних компонент розмірністю *r*×*n*.

Наступним підходом опрацювання ядер є алгоритм роботи на базі адаптивних фільтрів.

Адаптивний фільтр є електронний фільтр, який самостійно регулює свою передавальну функцію відповідно до алгоритму оптимізації з використанням сигналу похибки.

Оскільки параметри адаптивного фільтра змінюються в процесі його роботи, такий фільтр можна віднести до нелінійних пристроїв. Однак, при

кожному фіксованому значенні параметрів адаптивний фільтр — це лінійний пристрій, оскільки між його вхідними і вихідними сигналами зазвичай існує лінійна залежність, обумовлена поточним набором вагових коефіцієнтів (ВК), подібно лінійним фільтрам з фіксованими.

Для того щоб отримати усі змінювані в процесі роботи параметри, необхідно сформулювати критерій роботи адаптивного фільтра. Таким критерієм часто є мінімум деякої цільової функції, зазвичай, функції похибки між необхідним і вихідним сигналами адаптивного фільтра.

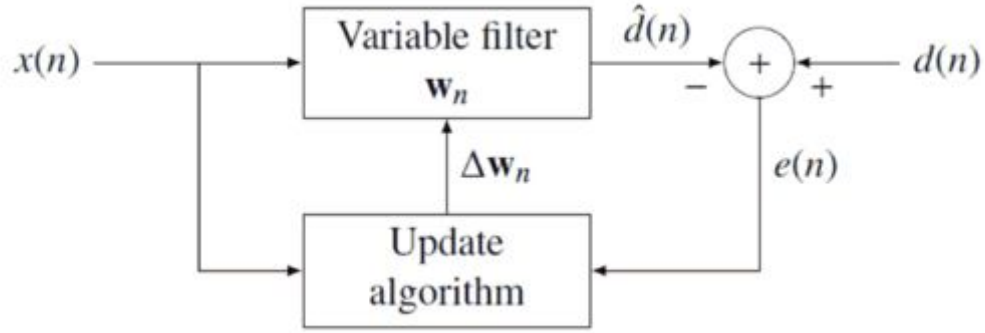

Рис.1.3. Блок схема роботи адаптивного фільтра

Даний підхід тяжко реалізувати на складних наборах даних, при класифікації складних параметрів наближені значення можуть бути не правильно розпізнанні на етапі навчання.

В деяких системах широко використовується регуляризація Тихонова. Цей метод названий на честь Андрія Тихонова [15], є найбільш часто використовуваним методом регуляризації некоректних задач. У статистиці метод відомий як пружною чистою регресією (з окремими випадками ласо регресії і рідж регресії) в машинному навчанні. Також метод поширений як вага розпаду, із декількома незалежними відкриттями. Існують інші назви даного методу - метод Тихонова-Міллера, метод Філіпс-Тумі, вимушений лінійний метод інверсії, а також метод лінійної регуляризації. Це пов'язано з алгоритмом Левенберга-Марквардт для нелінійних задач найменших квадратів.

Припустимо, що при відомій матриці A і вектор b ми хочемо найти вектор x , який задовільнив наступну рівність (1.9)

$$Ax = b \tag{1.9}$$

Стандартний підхід о вирішення є звичайним методом найменших квадратів лінійної регресії. Однак, якщо немає x задовольняє рівняння або більше одного x- то це рішення не є унікальним - проблема, як то кажуть, не буде коректне. У таких випадках звичайна оцінка найменших квадратів приводить до сверхдетермінірованним (більш підігнані), або більш часто недовизначених (під обладнаної) систем рівнянь. Більшість реальних явищ мають ефект фільтрів нижніх частот в прямому напрямку, де A визначає x до b. Тому при вирішенні оберненої задачі, зворотне відображення діє як фільтр високих частот, який має небажану тенденцію посилення шуму (власних значень / сингулярні значення за величиною в зворотному відображенні, де вони були маленькі в прямому відображенні). Крім того, звичайний метод найменших квадратів неявно анулює кожен елемент реконструйованої версії x що знаходиться в нуль-просторі A, а не дозволяє для моделі використання в якості попереднього рівня для x. Метод найменших квадратів прагне мінімізувати суму квадратів залишків, які можуть бути компактно записані як

$$||Ax - b||^2, \tag{1.10}$$

де ||.|| є Евклідова норма.

Для того, щоб віддати перевагу конкретному рішенню з бажаними властивостями, термін регуляризації може бути включений в цю мінімізації

$$||Ax - b||^2 + ||\Gamma x||^2,$$

для будь якого невідомого вибрана матриця Тихонова *Г*. У багатьох випадках ця матриця вибирається кратною одиничної матриці (*Г=αI*) віддаючи перевагу рішенням з меншими нормами; це відомо як регуляризацію L_2. В інших випадках оператори нижніх частот (наприклад, різницевий оператор або оператор зваженого Фур'є) можуть бути використані для забезпечення гладкості, якщо основний вектор, як вважають, в основному безперервний. Це регуляризація покращує обумовленість задачі, що дозволяє пряме чисельне рішення. Явна рішення, позначимо через $\hat{x}$ (1.11)

$$\hat{x} = (A^T A + \Gamma^T \Gamma)^{-1} A^T b. \qquad (1.11)$$

Ефект регуляризації можна варіювати за допомогою шкали матриці *Г*. Для *Г=0* це зводиться до вирішення непоправних найменших квадратів за умови, що $(A^T A)^{-1}$ існує. L_2 регуляризація використовується в багатьох контекстах осторонь від лінійних регресій, таких як класифікація з логістичної регресії або векторних машин підтримки і матричної прогонки.

У багатовимірної статистики і кластеризації даних, спектральні методи кластеризації використовують власні значення подібності матриці даних для виконання зниження розмірності, перш ніж кластеризація менших розмірностей. Матриця подібності подається якості вхідного сигналу і складається з кількісної оцінки відносної подібності кожної пари точок в наборі даних. У програмуванні часто можна зустріти у сегментації зображення, метод спектральної кластеризації відомий як сегментація на основі категоризації об'єктів. Загальна робота алгоритму виглядає так: на вході передається пронумерований набір точок даних, матриця подібності може бути визначена, як симетричною матрицею *A*, де $A_{ij} \geq 0$, що являє собою міру подібності між точками даних з індексами *i* та *j*. Загальний підхід до спектральної кластеризації полягає в використанні стандартного методу кластеризації на векторах відповідності Лапласіан матриці *A*. Є багато різних методів, щоб визначити Лапласіани, які мають різні математичні інтерпретації, тому

кластеризація також матиме різні інтерпретації. Власні вектори, які мають відношення це ті, що відповідають найменшим значеням Лапласіан для найменшого власного значення, за винятком, яке матиме значення 0. Для обчислювальної ефективності ці власні вектори часто обчислюються як власні вектори при відповідних найбільших кількох власних значеннях функцій від Лапласіана.

Одна техніка спектральної кластеризації є алгоритм нормалізовані порізи або алгоритм Ши-Малик, зазвичай використовується для сегментації зображень. Він розділяє точки на дві множини (B_1, B_2) на основі власного вектора *V*, другому відповідає найменше власне значення симетрично нормованої Лапласіани визначається як (1.12)

$$L^{norm} := I - D^{-\frac{1}{2}} A D^{-\frac{1}{2}}, \tag{1.12}$$

,де D діагональ матриці:

$$D_{ii} = \sum_i A_{ij}.$$

Математично еквівалентний алгоритм приймає власний вектор, що відповідає найбільшому власному значенні випадкового блукання нормованої матриці лапсасіаном $P = D^{-1}A$ A. Алгоритм Meila-Ши був розглянутий в контексті дифузійних карт і пов'язаний з обчислювальною квантовою механікою [10].

Інша можливість полягає в тому, щоб використовувати лапласовску матрицю, як $L = D - A$, а не симетричною нормованою лапласовскою матрицею.

Розподіл може бути зроблено різними способами, наприклад, шляхом обчислення медіан m компонент другого найменшого власного вектора *v*, і розміщення всіх точок, компонент *v* більше ніж m в B_1 , а решта в B_2. Алгоритм

може бути використаний для ієрархічної кластеризації шляхом багаторазового розбиття підмножин таким чином. Якщо матриця подібності A не була явно побудована, ефективність спектральної кластеризації може бути поліпшена, якщо рішення відповідної задачі на власні значення виконується в не матричним способом (без явного маніпулювання або навіть обчислення подібності матриця), як і в алгоритмі Ланцоша.

Для великих розмірів графіків, друге власне значення (нормованої) графа лапласіаном матриці часто погано обумовленої, що призводить до повільної збіжності ітераційних вирішувачів значень. Визначення умов є ключовою технологією прискорення збіжності, наприклад, в не матричних методах LOBPCG.

Безкоштовне програмне забезпечення для реалізації спектральної кластеризації доступне у великих проектах з відкритим вихідним кодом, як Scikit, MLlib для псевдо-власних векторів кластеризації з використанням методу ітерацій потужності.

1.3 Постановка задачі дослідження

Необхідно розробити мобільну систему для формування графіку проведення часу, котра зможе формувати рекомендації та робити аналіз даних з метою фільтрації надлишкової інформації, що дасть змогу краще формувати графік та розподіл часу. Система повинна бути розроблена під мобільні платформи, що в свою чергу дасть завжди легкий доступ до плану користувачеві. Також слід враховувати обчислювальну складність, та безпеку даних, адже при певних неполадках із телефоном, може призвести до втрати даних, тому слід враховувати можливість збереження даних віддалено.

Основною задачею буде розробка алгоритму формування множини даних яка буде відповідати потребам користувача. Також задачею буде формування вхідних множин, та оптимального опрацювання вхідної інформації. Це

потребує окремо існуючого елемента системи котрий буде займатись опрацюванням вхідної інформації та заповненням сховищ даних.

Для полегшення здач формування планів, або розподілу часу котрі часто використовуються в задачі управлінні часу. Необхідно розробити систему персоналізованого пошуку, в якому алгоритм вибірково допускає, яку інформацію користувач хотів би бачити, базуючись на інформації про користувача і, в результаті, користувачі відділяються від інформації, яка не відповідає їхнім точкам зору, фактично показуючи інформацію або ресурси котрі користувач вказує у своєму профілі. Яскравими прикладами є результати персоналізованого пошуку Google та персоналізована стрічка новин Facebook.

В машинному навчанні поширеною задачею є класифікація даних. Розгляньмо деякі задані точки даних, кожна з яких належить до одного з двох класів, а метою є вирішувати, в якому класі буде нова точка даних, точку даних розглядають як p-вимірний вектор (список p чисел), і хочуть дізнатися, чи можливо розділити такі точки (p - 1) - вимірною гіперплощиною. Це називається лінійним класифікатором. Існує багато гіперплощин, які могли би розділяти одні й ті ж дані. Одним із варіантів розумного вибору найкращої гіперплощини є такий, який пропонує найбільший проміжок або розділення між двома класами. Тож ми обираємо гіперплощину таким чином, щоби відстань від неї до найближчих точок даних з кожного боку була максимальною. Така гіперплощина, якщо вона існує, відома як максимально розділова гіперплощина, а лінійний класифікатор, що вона його визначає, — як максимально розділовий класифікатор, або рівнозначно, як перцептрон оптимальної стабільності.

1.4 Специфікація вимог до програмного продукту

Зробивши аналіз предметної області, приступаємо до опису варіантів використання. Кожен варіант використання – це окрема функція програмного продукту.

Проаналізувавши всі вимоги до системи, виділено наступні варіанти використання. Вони зображені в вигляді діаграми варіантів використання (рис. 1.4-1.5). З допомогою діаграми варіантів використання, можна з легкістю побачити основні функції системи.

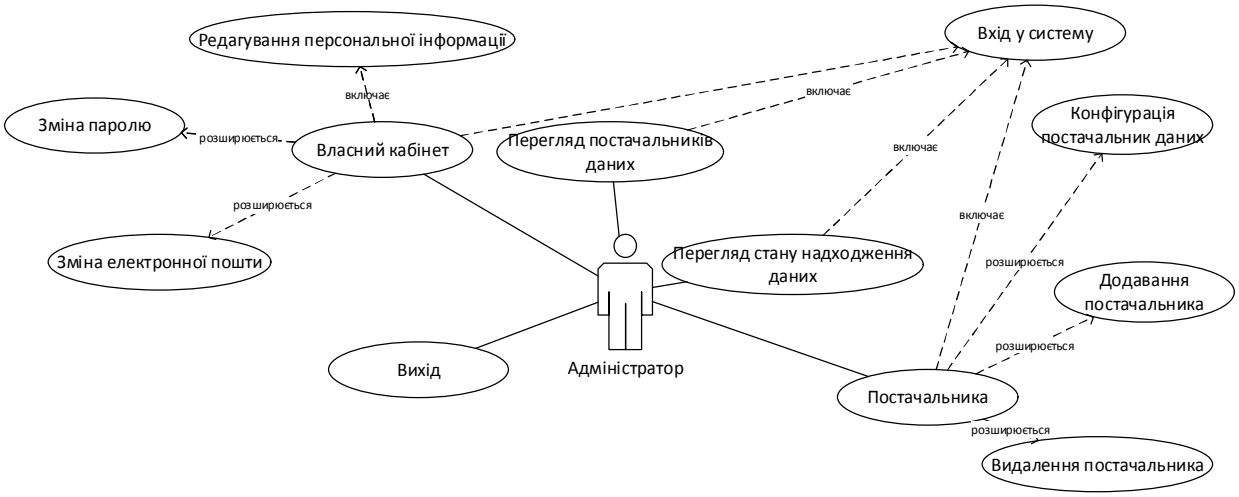

Рис.1.4. Діаграма варіантів використання Адміністратора

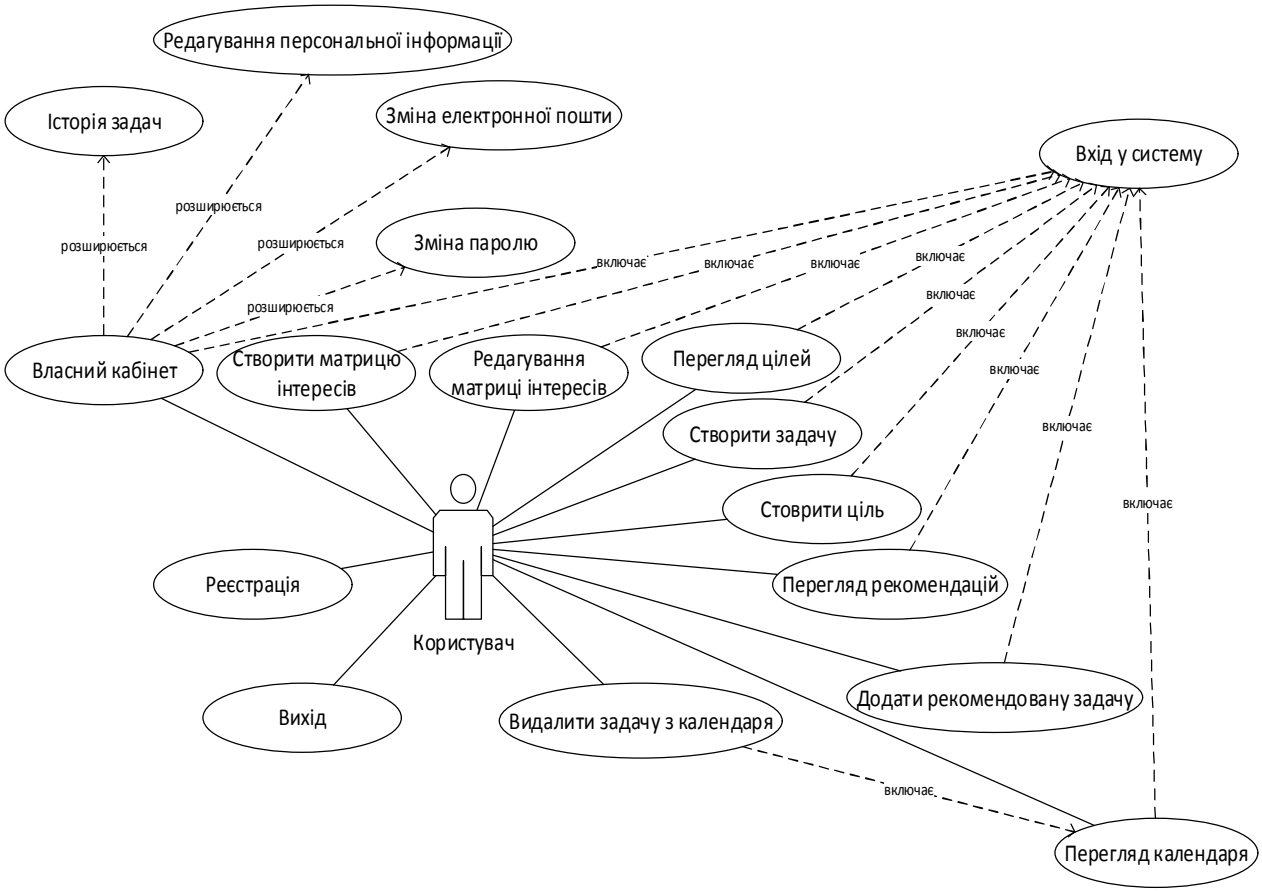

Рис.1.5. Діаграма варіантів використання Користувача

Наступним етапом у специфікації вимог до програмного продукту є розробка діаграми послідовності. З допомогою цієї діаграма можна виділити об’єкти в системі та відобразити дії між отриманими об’єктами сценарію програмного продукту, у вигляді послідовності повідомлень, якими обмінюються об’єкти. Діаграма послідовності зображена на рисунку 1.6.

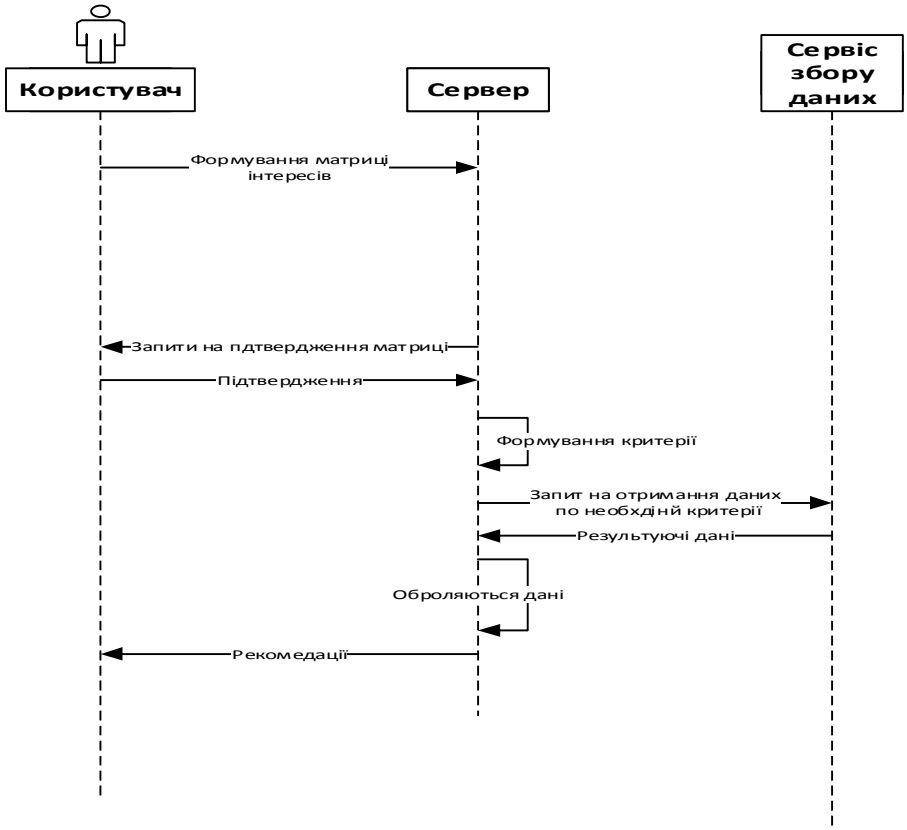

Рис.1.6. Діаграма послідовності

Проаналізувавши варіанти використання, було сформовано специфікацію функціональних вимог, яка наведена у таблиці 1.1.

Таблиця 1.1

Специфікація функціональних вимог

Ідент. вимоги	Назва вимоги (варіанту використання)	Атрибути вимоги	
		Пріоритет	Складність
1.	Реєстрація	Обов’язкова	Середня

Продовження таблиці 1.1

2.	Вхід	Обов’язкова	Середня
3.	Власний кабінет	Обов’язкова	Висока
4.	Зміна паролю	Обов’язкова	Середня
5.	Зміна пошти	Обов’язкова	Середня

6.	Перегляд календаря	Обов’язкова	Висока
7.	Створити задачу	Обов’язкова	Висока
8.	Створити ціль	Обов’язкова	Висока
9.	Історія задач	Обов’язкова	Висока
10.	Додати рекомендовану задачу	Обов’язкова	Висока
11.	Вихід	Обов’язкова	Середня
12	Перегляд рекомендацій	Необов’язкова	Низька
13	Перегляд списку постачальників	Обов’язкова	Висока
14	Додавання постачальника	Обов’язкова	Висока
15	Видалення постачальника	Обов’язкова	Висока
16	Редагування персональної інформації	Обов’язкова	Середня
17	Конфігурація постачальника	Обов’язкова	Висока
18	Створення матриці інтересів	Обов’язкова	Висока
19	Редагування матриці інтересів	Обов’язкова	Висока
20	Перегляд стану надходження даних	Обов’язкова	Висока

Специфікація нефункціональних вимог наведена у таблиці 1.2.

Таблиця 1.2

Специфікація нефункціональних вимог

Ідент. вимоги	Назва вимоги	Характеристики

Продовження таблиці 1.2

1.	Зручність інтерфейсу	Потрібний гнучкий і масштабований інтерфейс
2.	Інтуїтивне розміщення елементів дизайну	Елементи дизайну повинні бути розміщені на привичних для людей місцях
3.	Кольорова гамма	Згідно стандартів

4.	Швидкість відклику	2 секунди
5.	Гнучкість	Можливість додавання нових модулів без перезавантаження старих
6.	Локалізація	Наявність встановлення мови в додатку відносно потреби користувача
7.	Адаптованість	Коректне відображення нових версіях ОС.

Висновки до першого розділу

1. Здійснено аналіз об’єкту управління.
2. Проаналізовано поняття «інтелектуальний аналіз даних» та охарактеризовано основні проблеми, які виникають в процесі його реалізації.
3. Визначено та проведено аналіз основних проблем, які виникають в процесі самонавчання та формування множин на базі інтересів.
4. Описано та зроблено аналіз алгоритмів, для формування вибірок на базі певних критеріїв.
5. Визначені завдання дослідження для підвищення ефективності побудови часового плану.

РОЗДІЛ 2
МАТЕМАТИЧНЕ ТА АЛГОРИТМІЧНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ ЗАДАЧІ TIME MANAGEMENT

2.1. Механізм отримання даних із вхідного потоку інформації

На рисунку 2.1показана архітектура для простої системи вилучення інформації. Вона починається з обробки документа за допомогою декількох процедур:

- спочатку, вихідний текст документа розбивається на пропозиції, використовуючи розділювач речень, і кожне речення поділяється на слова, використовуючи спеціальні ключі .
- потім кожне речення позначається частина-оф-мовлення теги, які будуть виявитися дуже корисним на наступному етапі, з ім'ям виявлення об'єкта. На цьому етапі ми шукаємо згадки про потенційно цікавих об'єктів в кожному реченні.
- в кінці, ми використовуємо механізм виявлення прихованих зав’язків між різними суб'єктами в тексті.

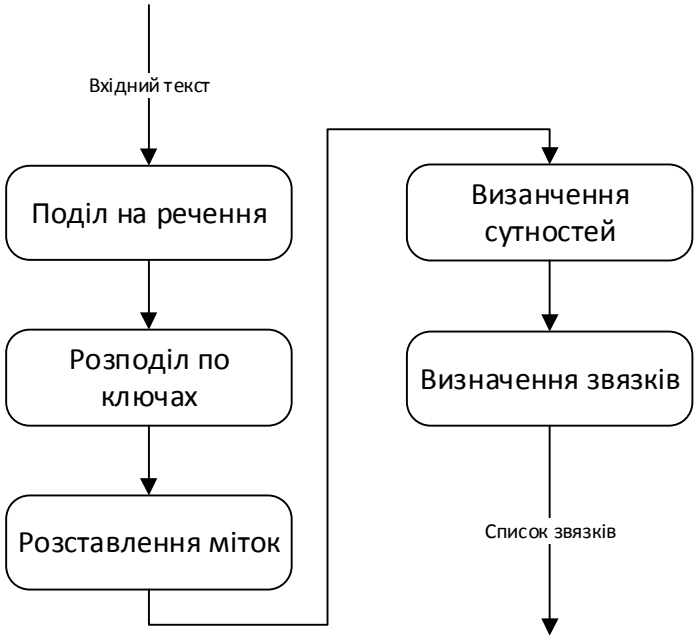

Рис. 2.1. Зображення механізму видобутку даних

Наступним кроком, з виявлення іменованих сутностей є сегментація і маркування об'єктів, які могли б взяти участь в процесі виявлення відносин між об’єктами. Як правило, це будуть певні іменники. У деяких задачах корисно також розглянути питання про невизначені іменники або сполучення іменників, що не обов'язково відносяться до об'єктів таким же чином, як і певні іменник та власні назви.

У видобутку зв’язків, ми шукаємо конкретні шаблони між парами об'єктів, які відбуваються поруч один з одним в тексті, а також використовуємо ці шаблони для створення кортежів записів відносин між суб'єктами.

Основна техніка для виявлення об'єкта є розділення та фільтрації за допомогою спеціальної сегментації сегментів і послідовність назви мульти-маркерів [20], як показано на рисунку 2.2. Менші квадрати показують лексемізацію на рівні слова, в той час як великі квадрати показують більш високого рівня сегменти. Кожний з цих великих квадратів називається сегментом. Як механізм розподілу по ключах, опускає прогалини, відривів зазвичай вибирає підмножина лексем. Крім того, шматочки, які породжені сегментацією не перекривають один одного в початковому тексті.

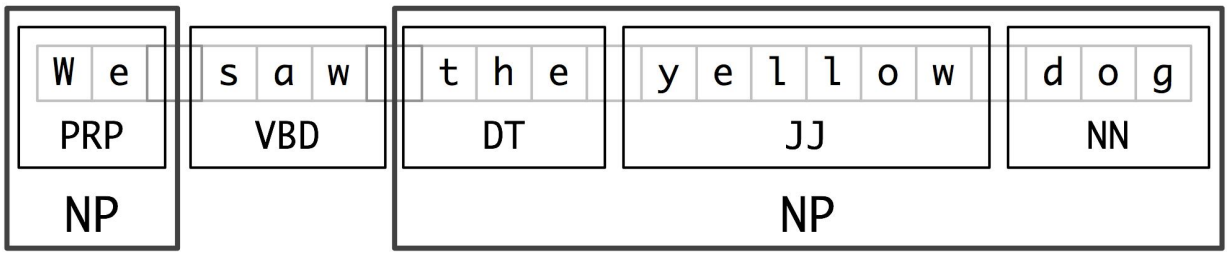

Рис. 2.2. Розподіл на сегменти та іменування.

Наступним етапом є відрив словосполучень або NP-відрив, де ми шукаємо шматки, відповідних окремих фраз іменників. Наприклад, ось деякі:«WallStreetJournal» текст з NP-шматками, відмічені за допомогою дужок:

[The/DT market/NN] for/IN [system-management/NN software/NN] for/IN [Digital/NNP] [

's/POS hardware/NN] is/VBZ fragmented/JJ enough/RB that/IN [a/DT giant/NN] such/JJ as/IN [Computer/NNP Associates/NNPS] should/MD do/VB well/RB there/RB ./.

Як ми можемо бачити, NP-частини часто більш дрібні шматки, ніж повні фрази іменника.Наприклад, «the market for system-management software for Digital's hardware»є однією фразою іменника (містить два вкладені фрази іменника), але він захоплений в NP-шматки більш простий шматок «the market». Одним з мотивів цієї різниці є те, що NP-брили визначені таким чином, щоб не містити інші NP-скибки. Отже, будь-які прийменникові фрази або додаткові, які модифікують номінальним, не будуть включені до відповідного NP-шматку, так як вони майже напевно містити інші фрази іменника.

Одним з найбільш корисних джерел інформації для NP-фрагментація є частина-оф-мовлення. Це один з основних елементів для виконання формування розмітки в нашій системі вилучення інформації.. Для того, щоб створити NP-розділювач, ми спочатку визначимо граматику шматка, що складається з правил, які вказують, як пропозиції повинні бути фрагментовані. В цьому випадку ми визначимо просту граматику з одним правилом регулярного виразу [2]. Це правило говорить про те, що шматок NP повинен бути сформований кожен раз, коли розділювач знаходить додатковий визначник (DT), а потім будь-яке число прикметників (JJ), а потім іменник (NN). Використовуючи цю граматику, ми створюємо шматок аналізатору , і перевірити його на нашому прикладі пропозиції . Результатом є дерево, яке ми можемо побачити на рисунку 2.3 .

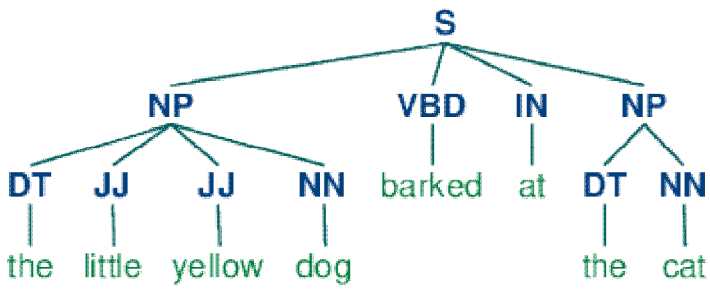

Рис. 2.3 Отримане дерево

Після того, як сутності були ідентифіковані в тексті, тоді ми можемо витягти відносини, які існують між ними. Як зазначалося вище, ми, як правило, будемо шукати відносин між зазначеними типами сутностей. Один із способів наближення цієї задачі є пошук усіх виглядів

$$(X, \alpha, Y)$$

- де X і Y названі об'єкти потрібних типів і α є рядок слів, що визначає відношення між X і Y.

В результаті ми можемо використовувати регулярні вирази, щоб витягнути тільки ті екземпляри, які висловлюють альфа відношення, що ми шукаємо.

2.2 SVM ранжування

У машинному навчанні, SVM ранжування є варіант машинного алгоритму опорних векторів, який використовується для вирішення певних завдань ранжирування (за допомогою навчання для ранжирування)[8]. Початкова мета алгоритму полягає в підвищенні продуктивності пошукової інтернет-системи. Проте, було встановлено, що упорядкування СВМ також може бути використаний для вирішення інших завдань, таких як ранг SIFT.

Формальніше, опорно-вектора машина будує гіперплощину, або набір гіперплощин у просторі високої або нескінченної вимірності, які можна використовувати для класифікації, регресії та інших задач. Добре розділення досягається гіперплощиною, яка має найбільшу відстань до найближчих точок тренувальних даних будь-якого з класів, нижча похибка узагальнення класифікатора досягається, в загальному випадку, збільшенням розділення.

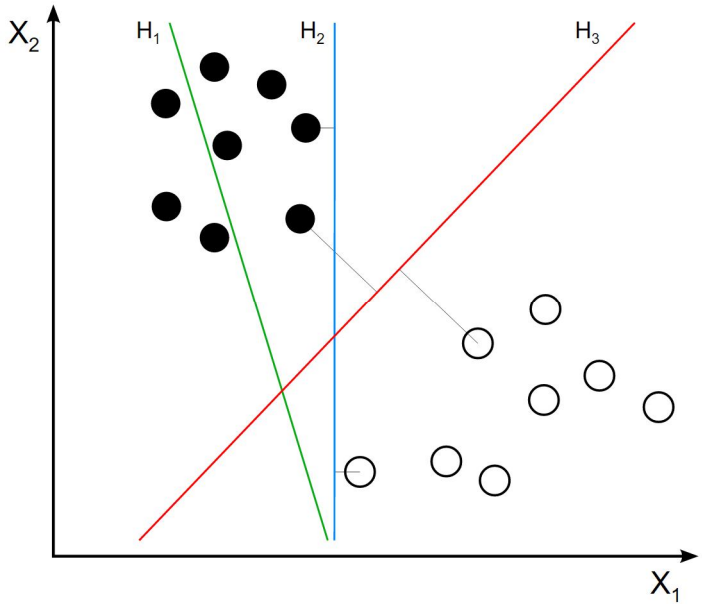

Рис. 2.4 Розподіл на класи

В той час, як первинну задачу можна сформулювати у скінченно-вимірному просторі, часто трапляється так, що множини, які треба розрізнювати, не є лінійно роздільними в ньому. З цієї причини було запропоновано відображати первинний скінченно-вимірний простір до простору набагато вищої вимірності, роблячи розділення простішим у тому просторі. Для збереження помірного обчислювального навантаження, відображення, які використовуються методом опорних векторів, розробляють такими, щоби забезпечувати можливість простого обчислення скалярних добутків у термінах змінних первинного простору, визначаючи їх у термінах ядрових функцій, яку обирають відповідно до задачі. Гіперплощини в просторі вищої вимірності визначаються, як геометричне місце точок, котрої скалярні добутки з вектором у цьому просторі є сталими. Вектори, котрі визначають гіперплощини, зазвичай обиралися як лінійні комбінації з параметрами α_i відображень векторів ознак x_i, які трапляються в базі даних. За такого вибору гіперплощини, точки x простору ознак, які відображаються на гіперплощину, визначаються відношенням (2.1)

$$\sum_i \alpha_i k(x_i, x) = \text{constant} \qquad (2.1)$$

Зауважте, що якщо $k(x, y)$ стає малою з віддаленням y від x, то кожен член цієї суми вимірює ступінь близькості пробної точки x до відповідних основних точок даних x_i. Таким чином, наведена вище сума ядер може використовуватися для вимірювання відносної близькості кожної пробної точки до точок даних, які походять з однієї або іншої з множин, які треба розрізнювати. Зверніть увагу на той факт, що множина точок x, відображена на будь-яку гіперплощину, може бути в результаті доволі вигнутою, уможливлюючи набагато складніше розрізнювання між множинами, які взагалі не є опуклими в первинному просторі.

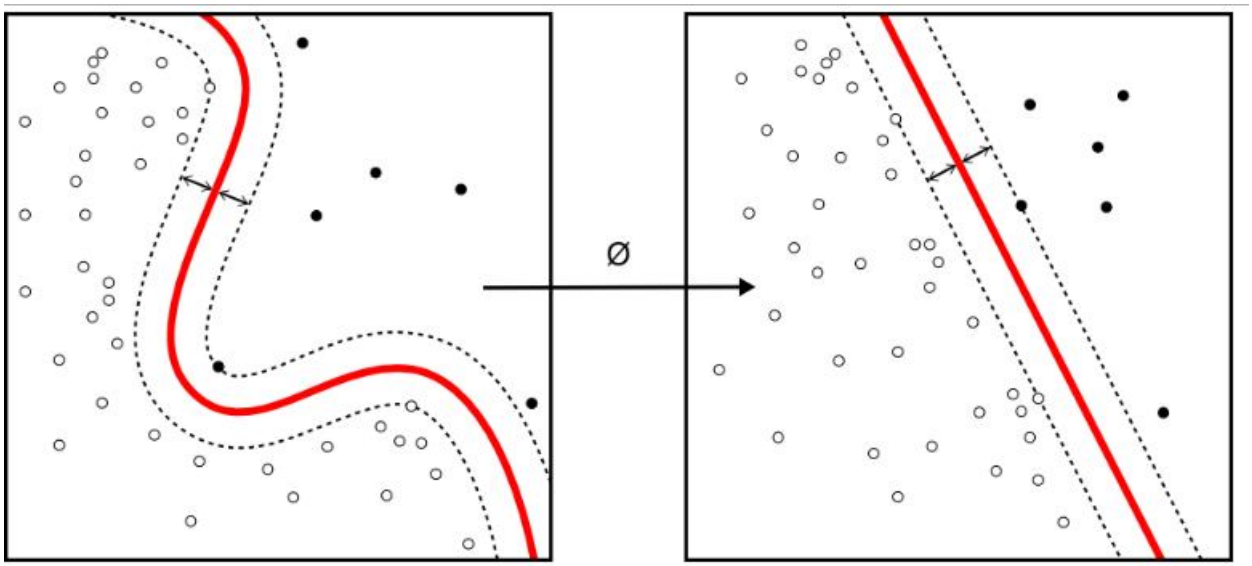

Рис.2.5 Ядрова машина

Алгоритм ранжирування SVM є самонавчаючою функцією пошуку, яка використовує методи парного ранжирування до адаптивного сортування результатів, заснованих на тому, яке «ставлення» вони мають для конкретного запиту. Функція ранжування SVM використовує функцію присвоєння для опису відповідності між пошуковим запитом і особливостей кожного з можливих результатів. Ця функція присвоєння проектує кожну пару даних (наприклад, пошуковий запит і переглянута веб-сторінка) на просторі

особливостей . Ці особливості в поєднанні з відповідними даними по кліку (які можуть виступати в якості проксі-сервера для того, як відповідна сторінка для конкретного запиту), а потім можуть бути використані в якості навчальних даних для ранжирування алгоритму SVM.

Як правило, рейтинг SVM включає в себе три етапи в період навчання:

1. Він відображає подібність між запитами і клацнули сторінок на певному просторі ознак.

2. Він обчислює відстань між будь-якими двома з векторів, отриманих на кроці 1.

3. Вона формує завдання оптимізації, яка схожа на стандартну класифікацію SVM і вирішує цю проблему за допомогою регулярного SVM решателя.

Загальний опис алгоритму ранжування :

$$C \ -\text{це множин із } c_i \text{ елменітв}$$

$$r - \text{метод ранжування}$$

Якщо виконати r на множину C отримаємо асиметричну бінарну матрицю. Якщо ранг c_i більший за c_j(2.2):

$$r\ c_i < rc_j \tag{2.2}$$

то відповідна позиція в матриці встановлюється значенням «1» у іншому випадку «0».

Кендалла Тау також виділяє кореляційний коефіцієнт, котрий широко використовується для порівняння двох методів ранжування на одній множині.

$$r_1, r_2 - \text{метод ранжування}$$

- ці методи ранжирування використовуються на множині Cі коефіцієнт Кендалла Тау можна порахувати із допомогою цієї формули (2.3)

$$\tau(r_1, r_2) = \frac{P-Q}{P+Q} = 1 - \frac{2Q}{P+Q}, \tag{2.3}$$

- де P– це число узгоджених пар, а Q – це число не узгоджених. Пара $d_i \neq d_j$ є узгоджена, якщо r_a і r_b рівні у їхніх двох порядках d_i і d_j у іншому випадку вони не погоджуються.

Також для алгоритму SVM необхідно визначити якість вхідної інформації, а саме:

1. Точність
2. Відгук
3. Середня точність

Нехай для певного запиту до бази даних можна оголосити такі значення:

$$P_{relavent} - \text{ множина відповідної інформації у базіданих}$$

$$P_{retrived} - \text{це множмна виданої фнормації}$$

Ці визначення можуть бути представлені у такій формі:

$$Precision = \frac{|P_{relavent} \cap P_{retrived}|}{|P_{retrived}|} \quad (2.4)$$

$$Recall = \frac{|P_{relavent} \cap P_{retrived}|}{|P_{relavent}|} \quad (2.5)$$

$$AveragePrecision = \int_0^1 Prec(Recall) dRecall \ , \quad (2.6)$$

- де $Prec(Recall)$ є *Precision* у *Recall.*

Нехай r^* і $r_{f(q)}$ будуть очікуваний і запропонований метод ранжування, нижня межа середньої точності методу $r_{f(q)}$ може бути представлена із допомогою формули (2.7)

$$AvgPrec(r_{f(q)}) \geqq \frac{1}{R}\left[Q + \binom{R+1}{2}\right]^{-1} (\sum_{i=1}^{R} \sqrt{i})^2, \quad (2.7)$$

- де Q – це число різних елементів у верхній частині трикутна матриці r^* і $r_{f(q)}$;

R – це число коректної інформації;

SVM класифікатор можна описати наступним чином:

Якщо $(\overrightarrow{x_i}, y_i)$ – це елементи навчальної множини де $\overrightarrow{x_i}$ вектор ознак і y_i класифікує категорію $\overrightarrow{x_i}$. Стандартний SVM класифікатор для певної множини може бути визначений як рішення проблеми оптимізації [5]. Рішення оптимізаційної проблеми представлення у формулі (2.8).

$$\vec{w}^* = \sum_i \alpha_i y_i \, x_i \tag{2.8}$$

Припустімо тепер, що ми хотіли би вчитися нелінійного правила класифікації, яке відповідає лінійному правилу класифікації для перетворених точок даних $\varphi(\overrightarrow{x_i})$ Крім того, в нас є ядрова функція k, яка задовольняє

$$k\left(\overrightarrow{x_i}, \overrightarrow{x_j}\right) = \varphi(\overrightarrow{x_i})\varphi(\overrightarrow{x_j})$$

Ми знаємо, що вектор класифікації $\vec{w}$ у перетвореному просторі задовольняє

$$\vec{w}^* = \sum_i^n c_i y_i \, \varphi(\overrightarrow{x_i}) \quad ,$$

, де c_i отримуються розв'язанням задачі оптимізації

$$f(c_1 \dots c_n) = \sum_{i=1}^{n} c_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} c_i y_i (\ \varphi(\overrightarrow{x_i})\varphi(\overrightarrow{x_j}))c_j y_j$$

мінімізувати

$$\sum_{i=1}^{n} c_i y_i = 0$$

за умов

$$\sum_{i=1}^{n} c_i y_i = 0$$

та

$$0 < c_i < (2n\lambda)^{-1}$$

для всіх *i*.

Коефіцієнти c_i може бути знайдено розв'язанням задачі квадратичного програмування ми можемо знайти такий індекс *i*, що, так що $0 < c_i < (2n\lambda)^{-1}$, так що $\varphi\left(\vec{\mathrm{x}}_{-1}\right)$ лежить на межі розділення у перетвореному просторі, й тоді розв'язати

$$b = \vec{w}\varphi(\vec{x_\iota}) - y_i = \left[\sum_{k=1}^{n} c_k y_k k\left(\vec{x_k}, \vec{x_J}\right)\right] - y_i$$

Нарешті, нові точки може бути класифіковано обчисленням

$$\vec{z} \to sgn(\vec{w}\,\varphi(\vec{z}) + b) = sgn\left(\left[\sum_{i=1}^{n} c_i y_i k(\vec{x_\iota}, \vec{z})\right] + b\right)$$

2.3.Проектування алгоритму ідентифікації іменованих сутностей

Іменована сутність - це певні фрази на подобі іменників, які відносяться до конкретних типів осіб, таких, як організації, особи, дати і так далі. На рисунку 2.15 перераховані деякі з найбільш часто використовуваних типів NE. Вони повинні бути зрозумілі, крім "Facility":це артефакти антропогенні в областях архітектури та цивільного будівництва; і "GPE": геополітичних утворень, таких як місто, штат / провінція, і країни.

NE Type	Examples
ORGANIZATION	*Georgia-Pacific Corp., WHO*
PERSON	*Eddy Bonte, President Obama*
LOCATION	*Murray River, Mount Everest*
DATE	*June, 2008-06-29*
TIME	*two fifty a m, 1:30 p.m.*
MONEY	*175 million Canadian Dollars, GBP 10.40*
PERCENT	*twenty pct, 18.75 %*
FACILITY	*Washington Monument, Stonehenge*
GPE	*South East Asia, Midlothian*

Рис. 2.15. Приклад найпоширеніших іменних сутностей

Мета системи розпізнавання іменованих сутностей (NER) [11], щоб ідентифікувати всі текстові згадки про іменованих сутностей. Це може бути розбита на дві підзадачі: визначення меж NE та визначення його типу. У той час як NER по суті є часто початком до виявлення відносин у добуванні інформації, він також може сприяти вирішенню інших завдань. Наприклад, в автовідповідача (QA), ми намагаємося поліпшити точність пошуку інформації шляхом відновлення не цілої сторінки, а тільки тієї частини, яка містить відповідь на питання цікаві для користувача. Більшість систем QA приймають форматовані документи, а потім намагалися ізолювати мінімальний фрагмент тексту в документі, що містить відповідь. Тепер припустимо, що це питання було «Хто був першим президентом США?», і один з документів, який був витягнутий містив наступний уривок:

«The Washington Monument is the most prominent structure in Washington, D.C. and one of the city's early attractions. It was built in honor of George Washington, who led the country to independence and then became its first President.»

Аналіз питання заставляє нас очікувати, що відповідь повинна мати вигляд «X» був першим президентом США, де «X» являє собою не тільки імена, але також відноситься до іменованої суті типу «PERSON», це повинно дозволити нам ігнорувати перше речення в проході. У той час як він містить два входження «Washington», іменна сутність суб'єкт повинен сказати нам, що жоден з них не має правильний тип.

Одним з варіантів було б шукати кожне слово у відповідному списку імен. Наприклад, в разі пошуку розташувань, можна скористатись необхідним довідником, або географічним словником, як «AlexandriaGazetteer» або «GettyGazetteer». Проте, коли робити це наосліп виникає проблема, як показано на рисунку 2.16.

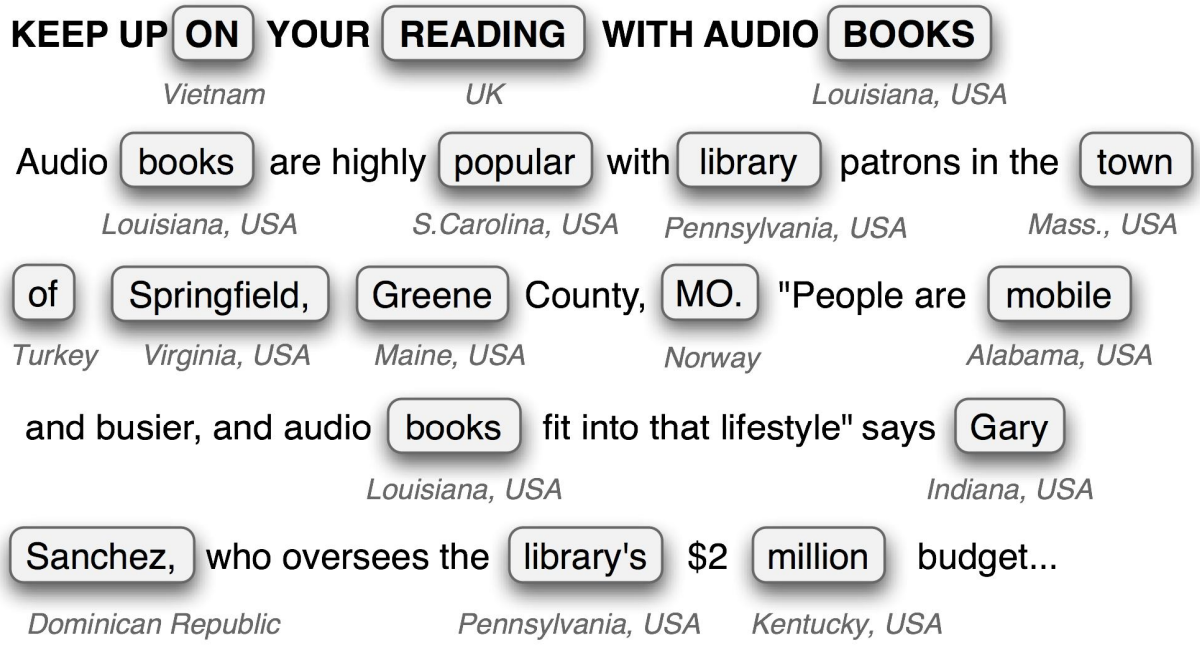

Рис. 2.16. Визначення розташування із допомогою словників

Зауважимо, що словник має хороше освітлення місць у багатьох країнах, і евірно знаходить місця, як Санчес в Домініканській Республіці та у В'єтнамі. Звичайно, ми могли б пропустити такі місця з гео-словника, але тоді ми не зможемо ідентифікувати їх, коли вони з'являються в документі. Це стає ще складніше в разі імен для людей або організацій. Будь-який список таких імен, має погане покриття. Нові організації створюються кожен день, так що якщо ми намагаємося мати справу з сучасною стрічкою новин або записів у блозі, то навряд чи ми зможемо визначити багато об'єктів, що використовують гео-словник пошуку.

Іншим складність пошуку обумовлена тим, що багато термінів названо неоднозначно. Таким чином, в «May»і «North», ймовірно, будуть частини названих сутностей за «DATE»і «LOCATION», відповідно, але обидва могли бути частиною «PERSON»; навпаки «Christian Dior» виглядає як «PERSON», але більш імовірно, будуть мати тип «ORGANIZATION». Термін, як «Yankee» буде звичайний модифікатор в деяких контекстах, але будуть позначені як іменована сутність типу «ORGANIZATION» в фразі «Yankee Infielders.

Подальші проблеми виникають у зв'язку з іменами кількох слів, як «Stanford University», і по іменах, які містять інші назви, такі як«Cecil H. Green Library» і «Escondido Village Conference Service Center». В розпізнаванні іменованих сутностей, ми повинні мати змогу визначити початок і кінець багато-ключової послідовності.

Розпізнавання іменованої сутності є завданням, яке добре підходить до типу підходу класифікатора на основі, яку ми бачили на іменних частинах. Зокрема, ми можемо побудувати макет, що мітить кожне слово в реченні, використовуючи формат IOB [18], де їх шматки позначені відповідного типу.

2.4 Проектування SVM алгоритму

Використовуючи формули, котрі були розглянуті у розділі 1.6, можна спроектувати SVM алгоритм. Для того нам необхідно функцію втрат, нехай

$\tau_{P(f)}$ буде Кендалла Тау між очікуваним методом ранжування r^* і запропонованим методом $r_{f(q)}$, це може доказати, що максимізація $\tau_{P(f)}$ допомагає мінімізувати нижню границю середньої точності $r_{f(q)}$. Протилежне значення $\tau_{P(f)}$ може бути вибрано як функцію втрати для мінімізування нижньої границі середньої точності для $r_{f(q)}$

$$L_{expected=-\tau_{P(f)}=-\int \tau(r_{f(q)},r^*)dPr(q,r^*)} \quad (2.9)$$

де $Pr\ (q,r^*)$ статистичний розподілr^*до певної критерії q.

З того моменту як функцію втрат не можна застосувати, використовується емпірична функція втрат для корегування даних в дії.

$$L_{empirical} = -\tau_S(f) = -\frac{1}{n}\sum_{i=1}^{n}\tau\left(r_{f(q_i)}, r_i^*\right) \quad (2.10)$$

Отримання даних для корегування відбувається наступним чином: n запитів до бази даних і кожний запит відповідає методу ранжування. Множина даних для навчання складається з n елементів. Кожний елемент складається з запиту та відповідного методу ранжування. Функція присвоювання $\Phi(q,d)$ є необхідною для того щоб присвоїти кожному запиту і елементу з бази даних до простору особливостей. Де кожна точка позначена відповідною оцінкою метода ранжування. Оптимальний вектор $\vec{w}^*$ опрацьований системою є (2.11)

$$\vec{w}^* = \sum {\alpha^*}_{k,l}\Phi(q_k, c_i) \quad (2.11)$$

Даний алгоритм використовується для оцінки подій по необхідному запиту. Алгоритм може тренуватись використовуючи кліки певних подій. Можна виділити три основних елементи:

1) запити;

2) оцінка результату;

3) кількість переходів користувачів по отриманому результату;

Використовуючи тільки 2 і 3 елементи не можна сформувати повноцінний набір даних для самонавчання. Отже алгоритм можна відобразити з таким записом:

$$(\vec{w}, \vec{\varepsilon}) = \frac{1}{2}\vec{w}\,\vec{w} + C_{onstant}\sum \varepsilon_{i,j,k}$$

$$\forall \varepsilon_{i,j,k} \geqq 0$$

$$\forall (c_i, c_j) \in \grave{r}_k$$

$$\vec{w}\left(\Phi(q_1, c_i) - \Phi\left(q_1, c_j\right)\right) \geqq 1 - \varepsilon_{i,j,1}$$

...

$$\vec{w}\left(\Phi(q_n, c_i) - \Phi\left(q_n, c_j\right)\right) \geqq 1 - \varepsilon_{i,j,1}, \qquad (2.12)$$

де $k \in \{1,2,\ldots n\}, i,j \in \{1,2,\ldots\}$.

Вирішення задачі лінійної регресії із допомогою SVM ранжування, полягає рішенні задачі мінімізації функції втрат, припускаючи, що рішення задається лінійною комбінацією деяких породжуючих функцій з яких можна побудувати векторну функцію

$$f(x) = \begin{cases} f_1(x) \\ \ldots \\ f_k(x) \end{cases}$$

Тоді функціонал прийме вигляд (2.13)

$$Q_\epsilon(a, X) = \sum_{i=1}^{l} |(\omega, f\,(x_i)) - \omega_0 - y_i|_\epsilon + \tau(\omega, \omega)^2 \to min_{\omega,\omega_0} \qquad (2.13)$$

припускаючи, що

$$f_0(x) = \sum_{i=1}^{k} \omega_i f_i(x).$$

Для цього вводиться визначення $C = \frac{1}{2\tau}$ і додаткові зміни ε_i^+ та ε_i^-.

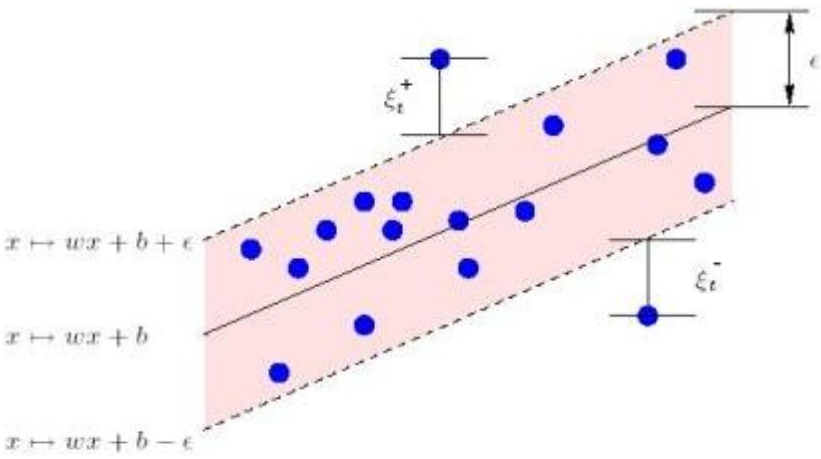

Рис. 2.17 Геометричне значення ε_i^+ та ε_i^-.

Наступним етапом є рішення здачі квадратичного програмування (2.14)

$$\begin{cases} \frac{1}{2}(\omega,\omega) + C\sum_{i=1}^{l}(\varepsilon_i^+ + \varepsilon_i^-) \to min_{\omega,\omega_0,\varepsilon_i^+,\varepsilon_i^-} \\ \big(\omega, f(x_i)\big) - \omega_0 \le y_i + \epsilon + \varepsilon_i^+, \qquad i = 1,..,l; \\ \big(\omega, f(x_i)\big) - \omega_0 \ge y_i - \epsilon - \varepsilon_i^-, \qquad i = 1,..,l \\ \varepsilon_i^+ \ge 0,\ i = 1,..,l; \\ \varepsilon_i^- \ge 0,\ i = 1,..,l \end{cases} \tag{2.14}$$

цю задачу можна перетворити до вигляду $\frac{1}{2}\mu^T H\mu + g\mu \to min_\mu$ при умові $A\mu \le b$ також $lb \le$ с, де $\mu -$ вектор $-$ сотвбець із стовбців $\omega, \varepsilon_i^+, \varepsilon_i^-$. Можна перебудувати матрицю враховуючи стовбці b і lb:

$$\begin{cases} \frac{1}{2}(\omega,\omega) + C\sum_{i=1}^{l}(\varepsilon_i^+ + \varepsilon_i^-) \to min_{\omega,\omega_0,\varepsilon_i^+,\varepsilon_i^-} \\ \big(\omega, f(x_i)\big) - \omega_0 - \varepsilon_i^+ \le y_i + \epsilon, \qquad i = 1,..,l; \\ \big(\omega, f(x_i)\big) - \omega_0 - \varepsilon_i^+ \le -y_i + \epsilon, \qquad i = 1,..,l \\ \varepsilon_i^+ \ge 0, \qquad i = 1,..,l; \\ \varepsilon_i^- \ge 0, \qquad i = 1,..,l \end{cases}$$

Я результат отримаємо :

$$A = \begin{Vmatrix} f^T(\varepsilon_1) & -1 & 0 & \dots & 0 \\ f^T(\varepsilon_2) & 0 & -1 & \dots & 0 \\ & & \ddots & & \\ f^T(\varepsilon_l) & 0 & 0 & \dots & 0 \\ -f^T(\varepsilon_1) & 0 & 0 & \dots & 0 \\ & & \ddots & & \\ -f^T(\varepsilon_l) & 0 & 0 & \dots & -1 \end{Vmatrix},$$

$$b = \begin{Vmatrix} y_1 + \epsilon \\ y_2 + \epsilon \\ \vdots \\ y_l + \epsilon \\ -y_1 + \epsilon \\ \vdots \\ -y_l + \epsilon \end{Vmatrix},$$

$$lb = \begin{Vmatrix} -\infty \\ -\infty \\ \vdots \\ -\infty \\ 0 \\ \vdots \\ 0 \end{Vmatrix},$$

кількість мінусів в безкінечності *lb* відповідає кількості утворених функцій а нулів кількість рівна 2*l.* Таким образом звели задачу до квадратичного програмування, в нашій задачі *C, ε* і проджючі фкнції котрі задані. Для

перевірки точності роботи алгоритму використаєм лінійну комбінацію даних наших функцій, та додаємо випадковий шум. В ході експерименту досліджуються різні, як дискретні, так і неперервні шуми. В якості базової функції вибрано $y = 3 + x - 1.5\sin(x)$. А в якості функцій утворювачів $x, e^x, \sin(x), \cos(x), x^{0.5}, x^{1.5}, x^0$.

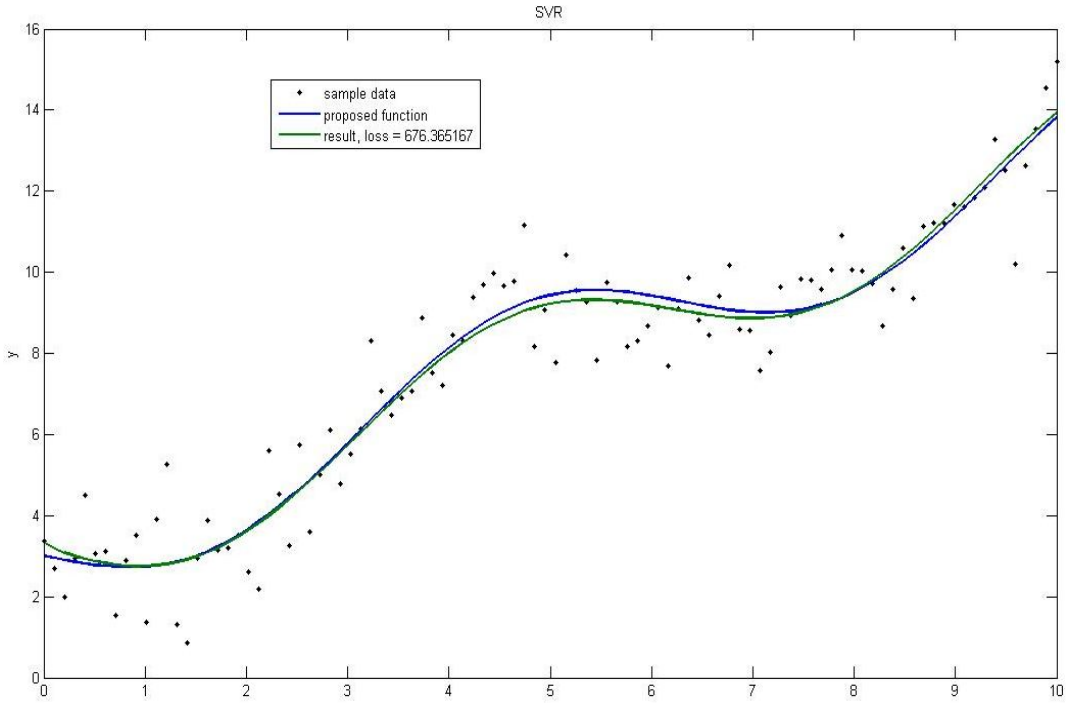

Рис. 2.18 Нормальний розподіл (дисперсія 1)

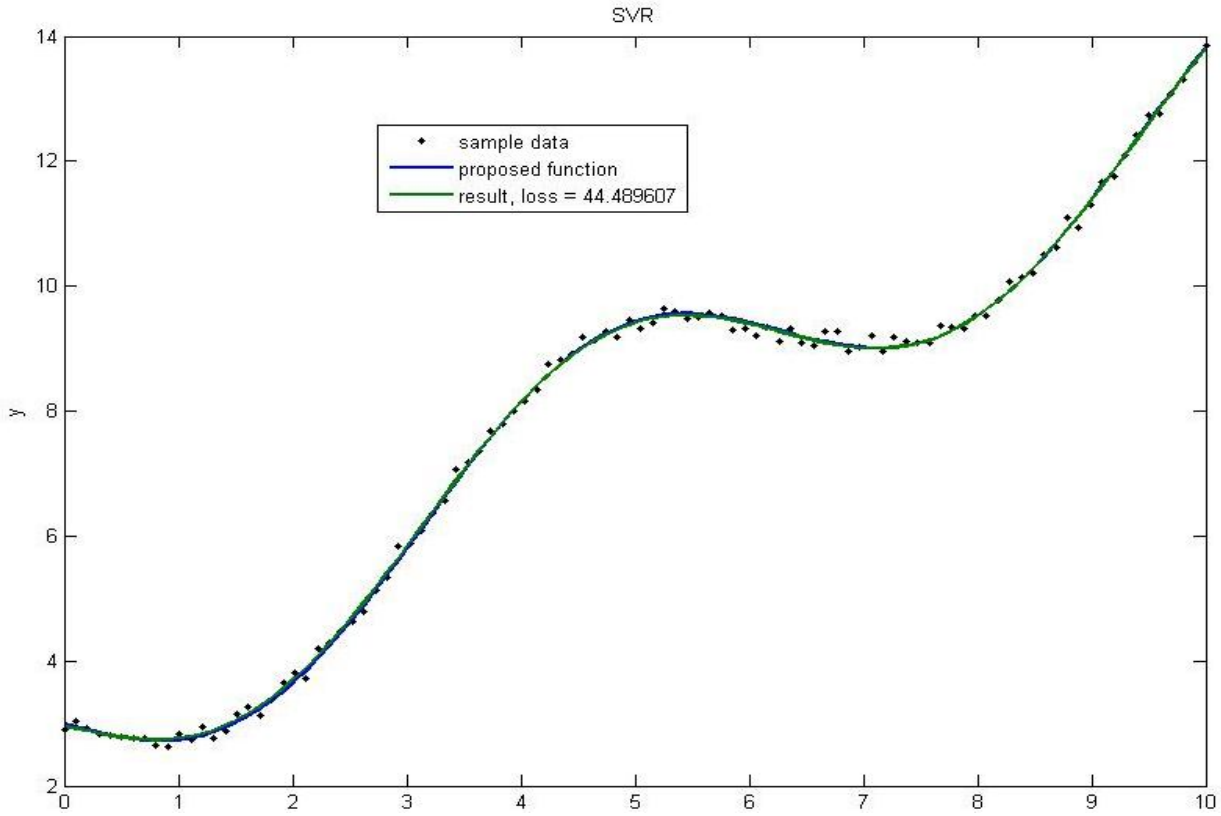

Рис. 2.19 Нормальний розподіл (дисперсія 0.1)

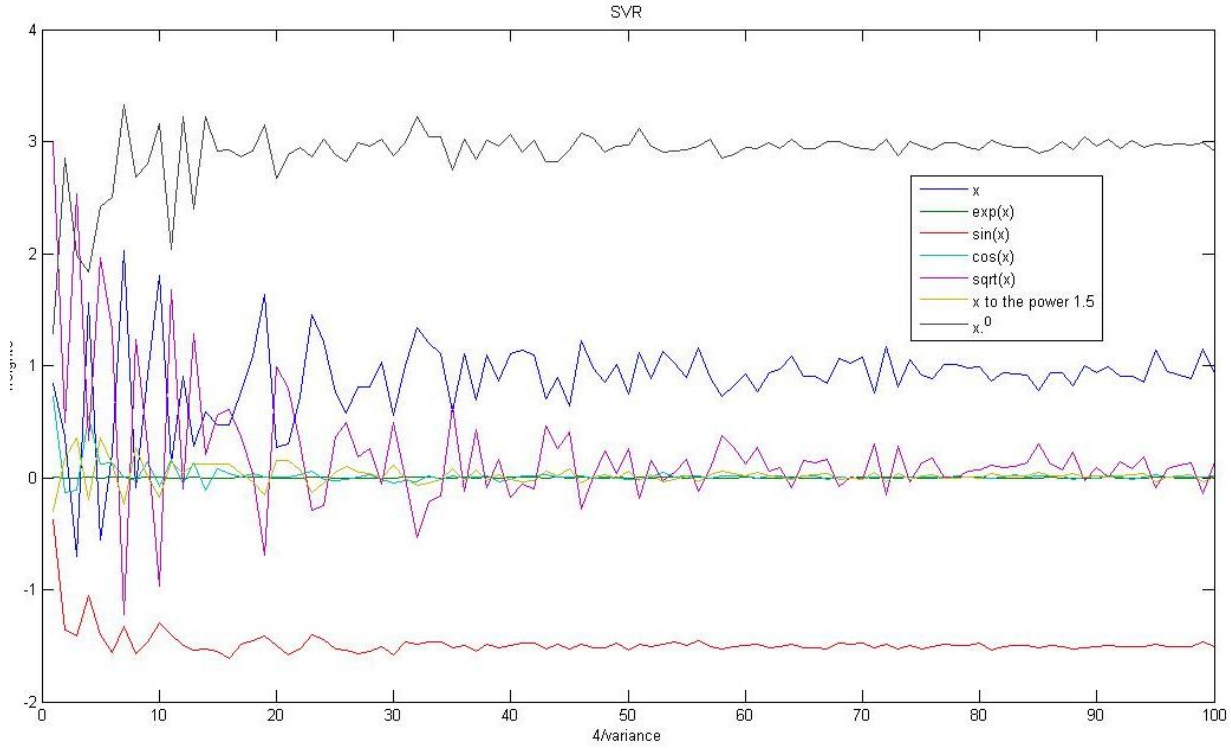

Рис. 2.20 Залежність ваг відповідних функцій від зворотньої дисперсії

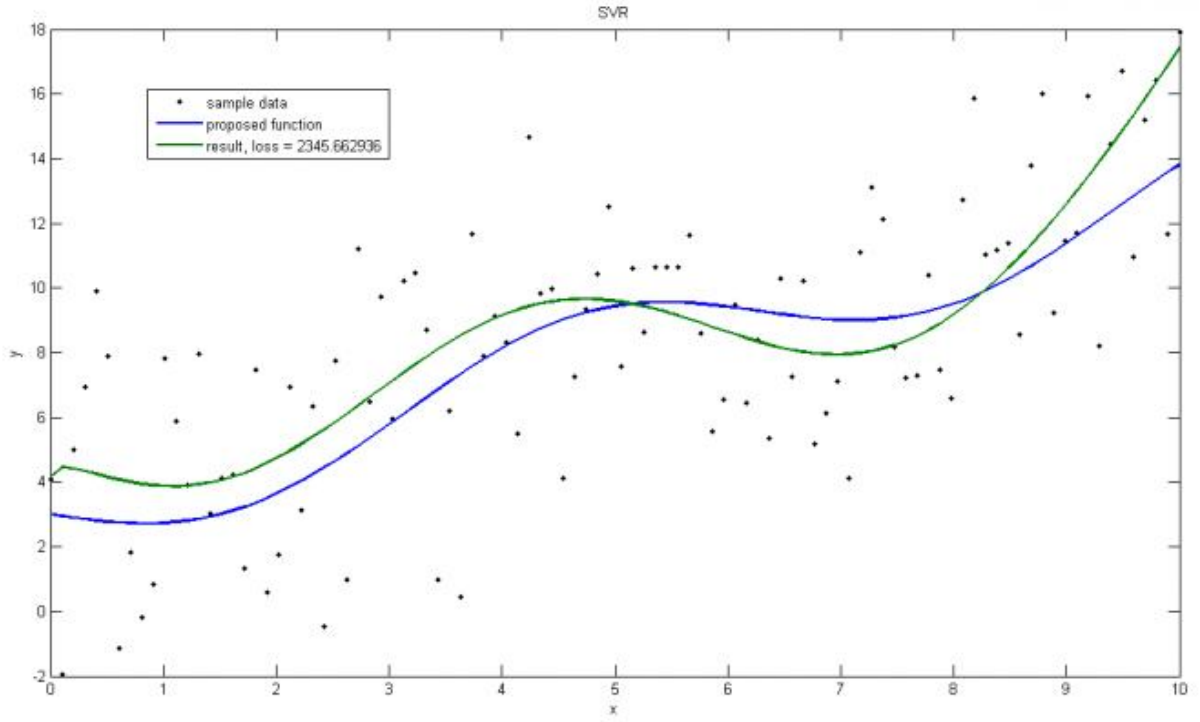

Рис. 2.21 Пуассоновский розподіл з великою дисперсією

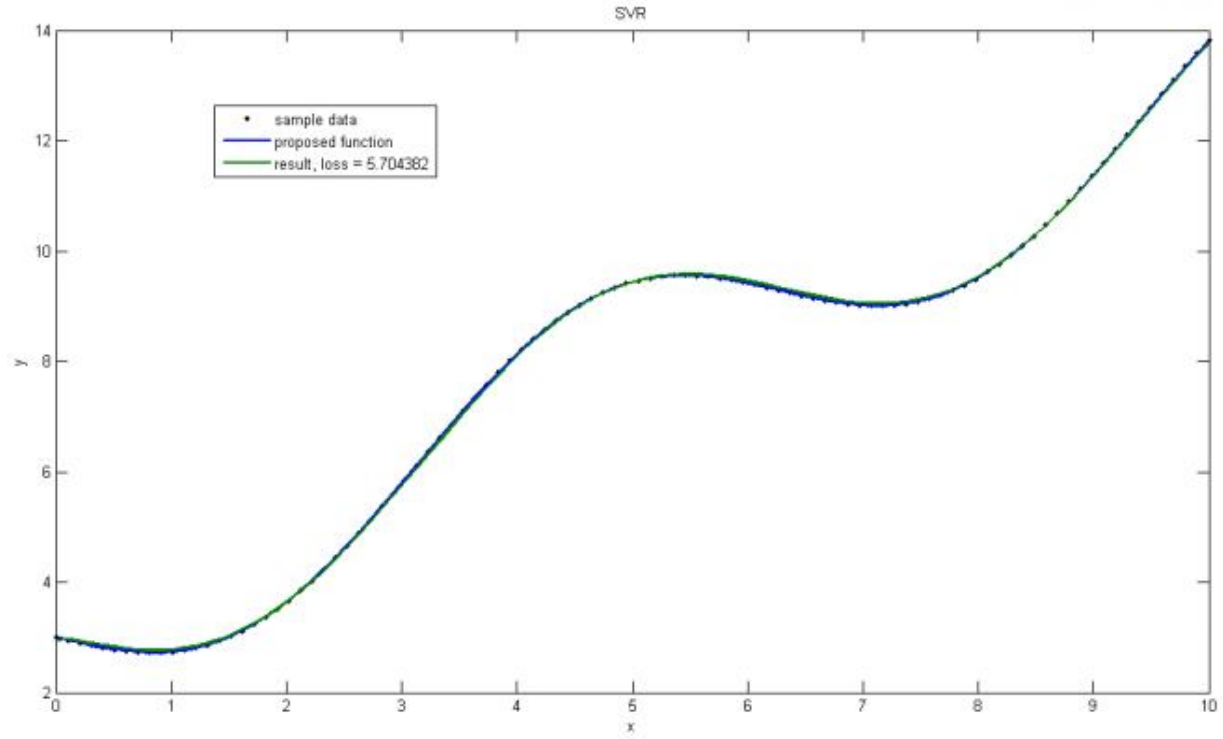

Рис. 2.22 Пуассоновский розподіл з малою дисперсією, отримуємо майже точне рішення.

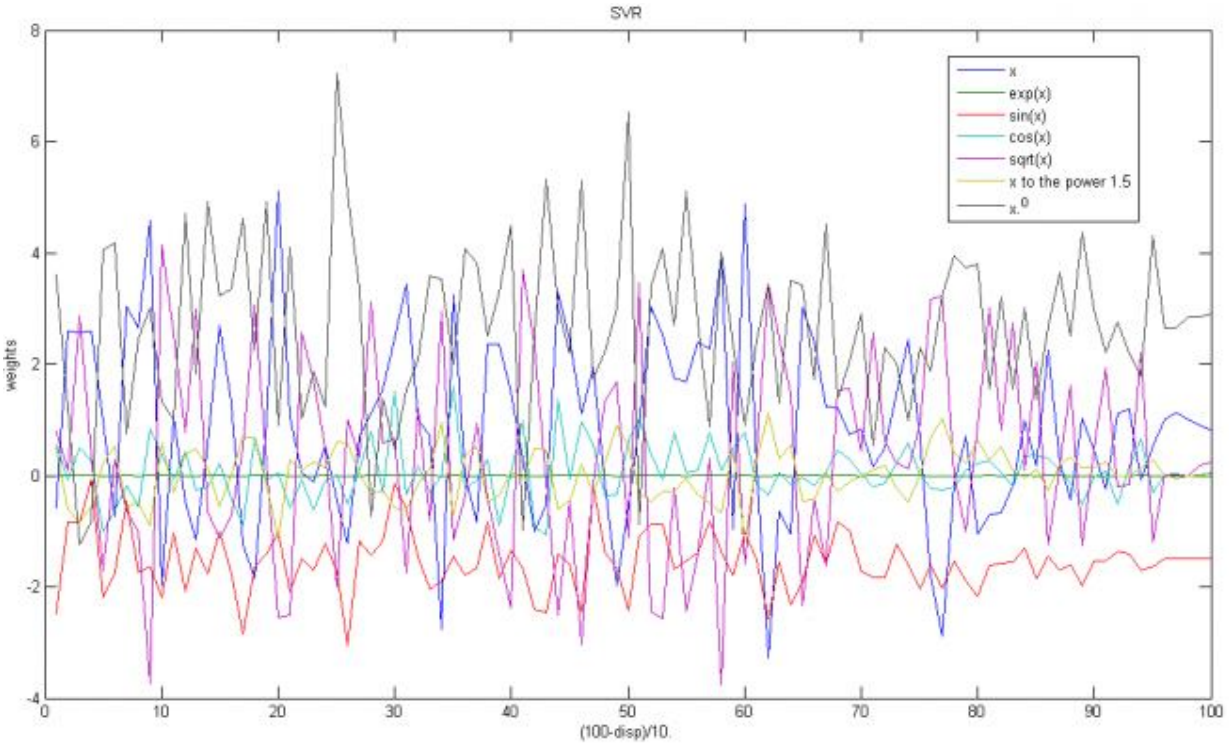

Рис. 2.23 Залежність ваг відповідних функцій від параметра

Висновки до розділу 2

За результатами другого розділу зроблено наступні висновки:

1. Визначено алгоритм для оптимізації і автоматизації задачі формування рекомендованого контенту.
2. Проаналізовано роботу алгоритму на тестових даних, що дозволяє приступити до реалізації системи .

РОЗДІЛ 3
ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

3.1. Розробка архітектури програмної системи

Потрібно розробити систему для аналізу вихідних даних для формування рекомендацій для заповнення календаря. Система повинна зберігати дані про наявні задачі (назва, дата початку виконання, дата кінця виконання, тривалість, вагомість, категорія, періодичність), інформацію про постачальників даних (назва, опис, електрона адреса, категорія, стан), інформацію про користувачів та їхні можливості (ім'я, прізвище, адрес (місто, країна, вулиця, поштовий індекс, будинок), електрона пошта, права доступу), дані про цілі(назва, дата початку виконання, дата кінця виконання, тривалість, категорія), інформацію про категорії (назва, тип), інформацію про рекомендацію (назва, час , назва постачальника, тривалість, рейтинг). Для кращого функціонування слід вести облік рейтинг вподобань користувача (пріоритет, популярність).

Архітектура системи зображена на рисунку 3.1.

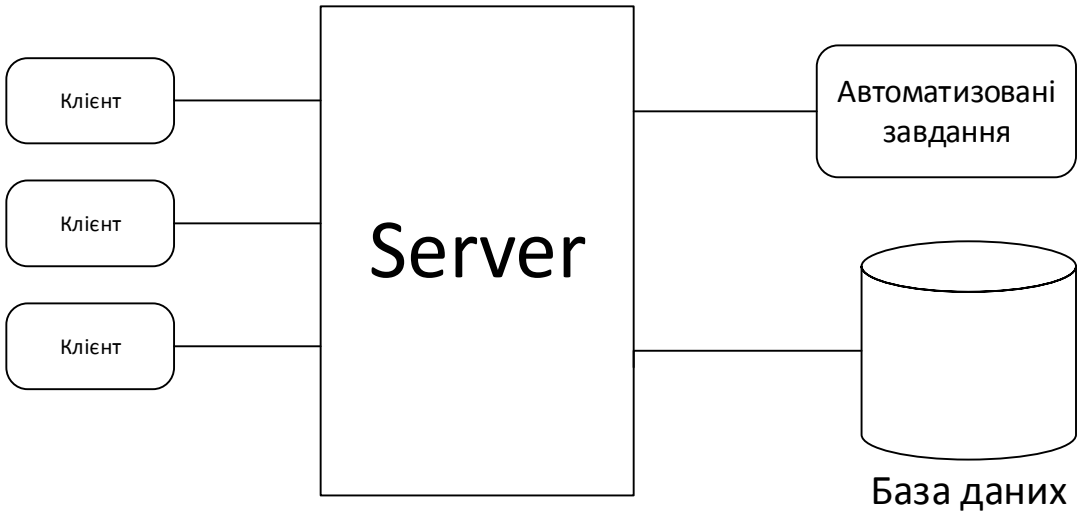

Рис.3.1. Архітектура системи

Так як цей продукт є мобільними додатком із використання веб-служби, то потрібно врахувати всі можливості росту аплікацій, адже вони із зростом популярності можуть призвести до певних проектних змін, тому слід врахувати

фактор масштабованості. Доцільним було рішення про розмежування логіки і використання спеціальних елементів для формування слабкої зв’язності, а також для покращення швидкодії використовувати асинхронні операції та сторінкове відображення елементів.

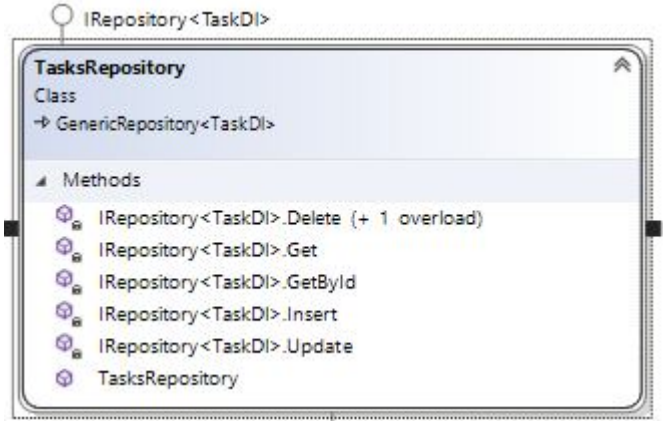

Рис.3.2 Клас TaskRepository

Клас TaskRepository (рис. 3.2) – цей клас є елементом доступу до бази даних, тобто містить у собі усі необхідні функції щодо взаємодії із інформаційними сутностями у БД.

Рис. 3.3 Клас TaskController

Клас TaskController (рис. 3.3) відіграє роль елементу комунікаційного елементу між клієнтом та сервером, повертаючи дані у JSON форматі. Він слугує для того, щоб проводити певні взаємодії стосовно задач.

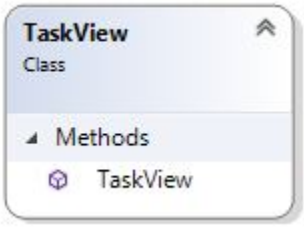

Рис. 3.4 Клас TaskView

Клас TaskView (рис. 3.4) відіграє роль елементу відображення даних, містить у собі усі елементи з котрими взаємодіє користувач.

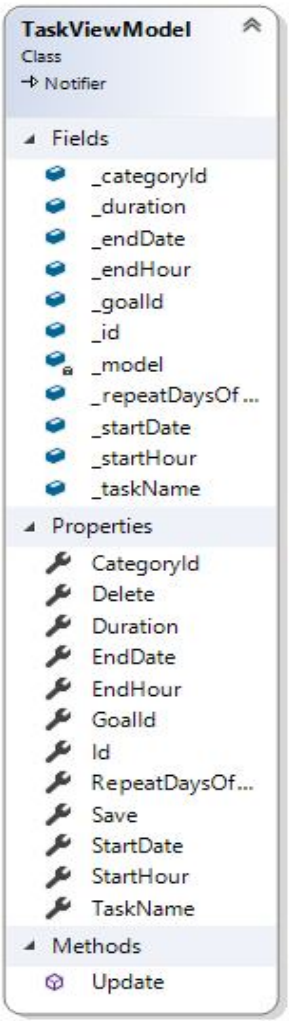

Рис.3.5 КласTaskViewModel

Клас TaskViewModel (рис. 3.5) відіграє роль елементу комунікаційного елементу між рівнем представлення та рівнем маніпулювання даних.

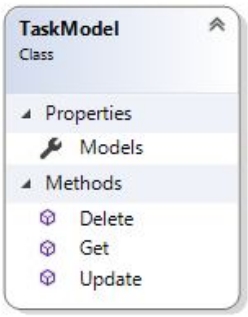

Рис.3.6 КласTaskModel

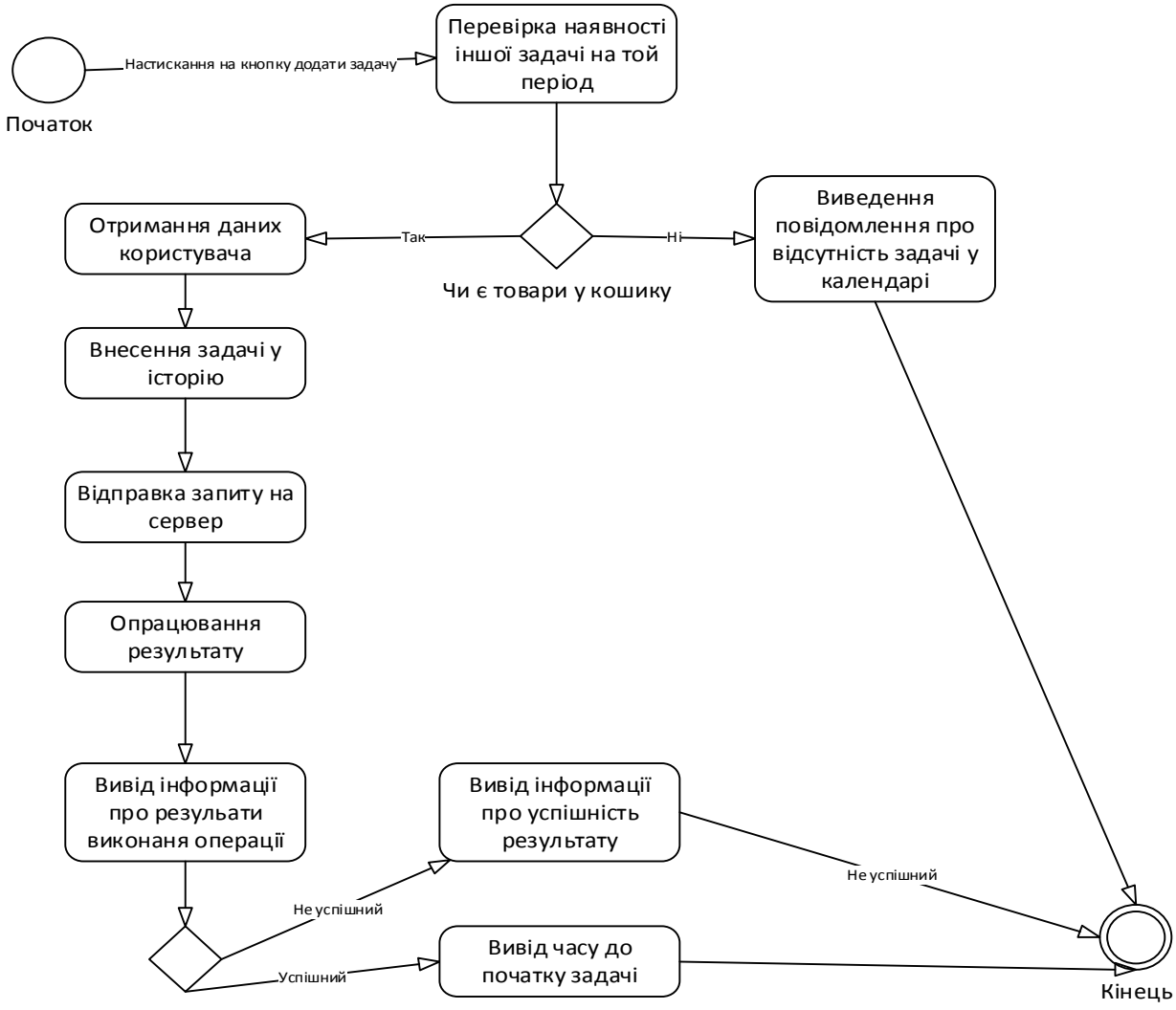

Рис. 3.7. Діаграма діяльності варіанту використання «Створення задачі»

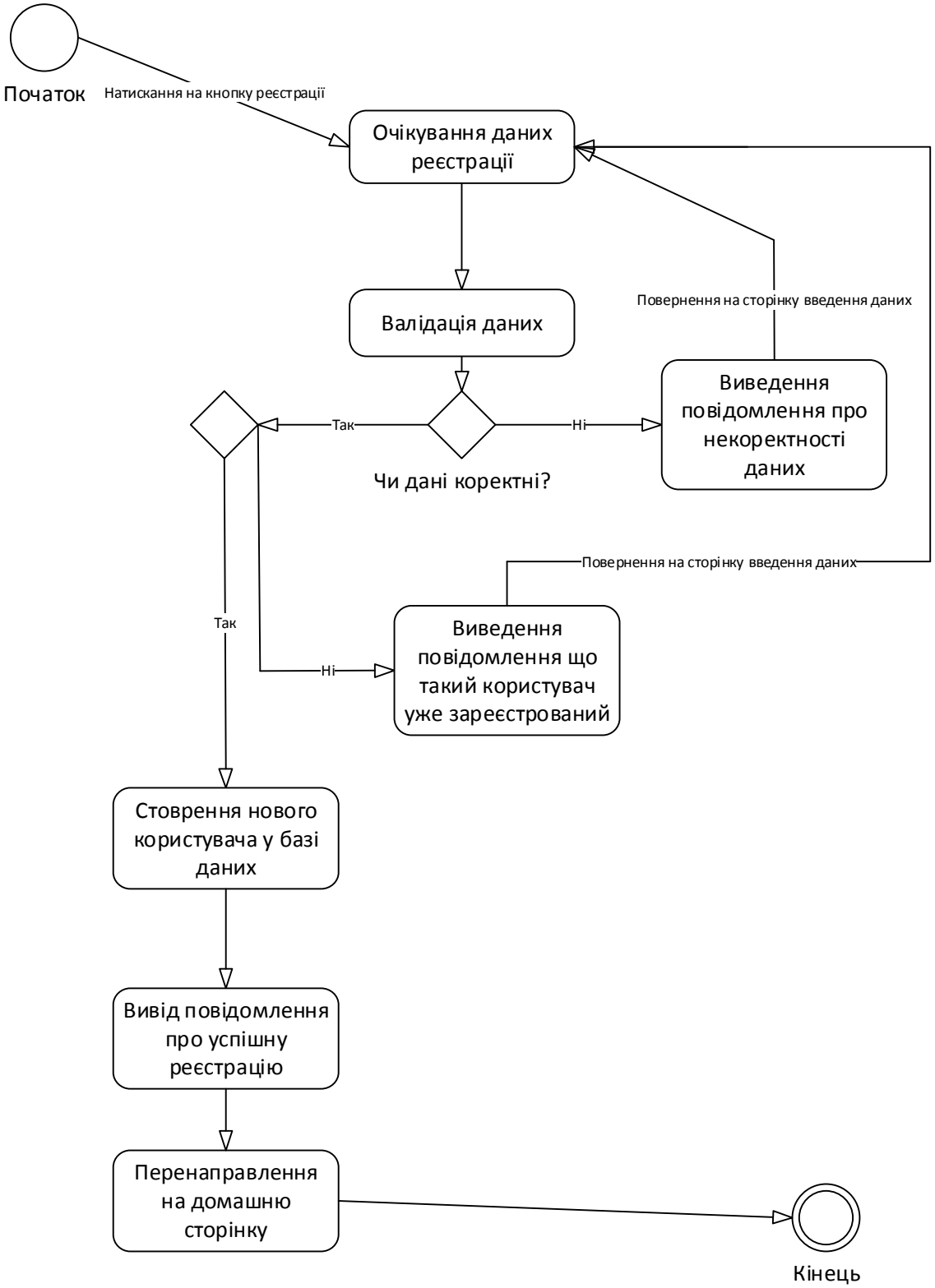

Рис. 3.8 Діаграма діяльності варіанту використання «Реєстрації користувача»

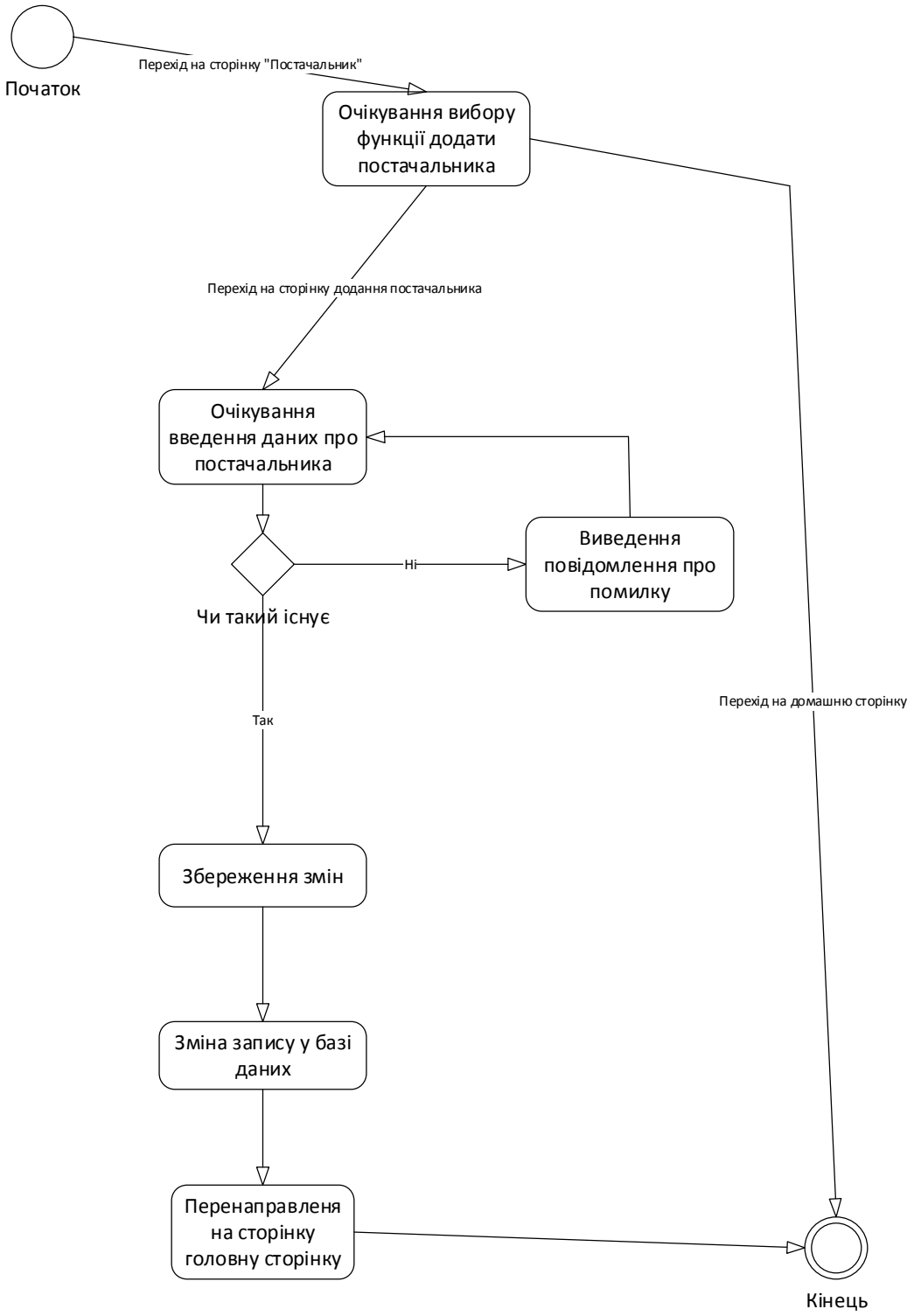

Рис. 3.9 Діаграма діяльності варіанту використання «Блокування користувача»

Клас TaskModel (рис. 3.10) відіграє роль елементу комунікаційного елементу між клієнтом та сервером, приймаючи та повертаючи дані у JSON форматі. Він слугує для того, щоб проводити певні взаємодії стосовно задач.

Рис. 3.10. Класи CategoryConvertor, GaolConverter, TaskConverter

Класи TaskModelConvertor, CategoryModelConverter, GoalModelConverter (рис. 3.10) відіграють роль конвертерів між модулями у системі. Для того щоб система могла краще масштабуватись було виділено окремі рівні.Це дає змогу швидко змінювати модуля. Наприклад,рівень представлення, так як основна логіка розміщена у модулі бізнес логіки і взаємодіє з допомогою абстракцій, що призводить до формування слабкої зв’язності.

Інші класи у системі відіграють роль комунікаторів, або елементів доступу до БД. Також присутні елементи абстракції у вигляді абстрактних класів або інтерфейсів. Інтерфейси слугують для взаємодії із контейнером залежності.

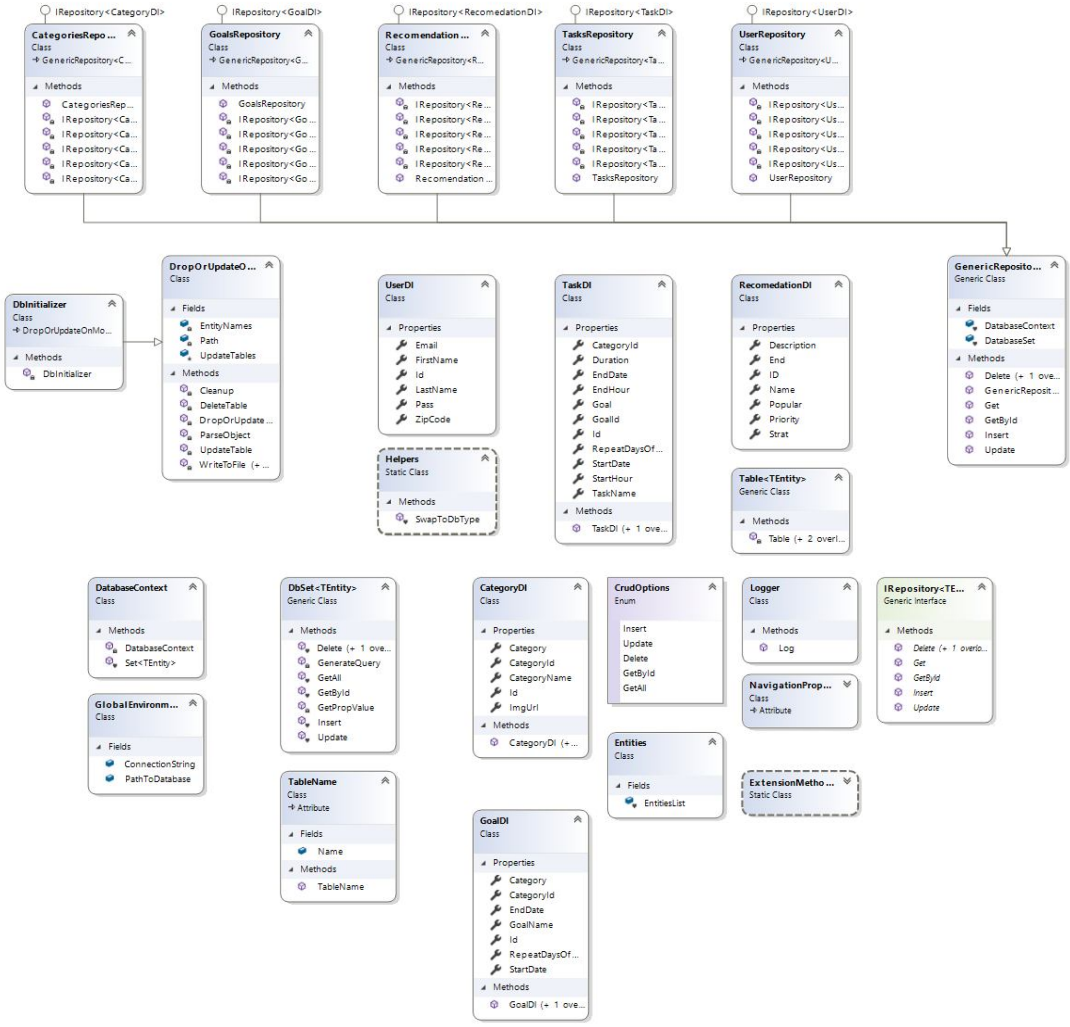

Рис.3.11. Діаграма класів рівня доступу до бази даних.

Для розробки даного програмного продукту було вибрано Xamarin на базі архітектурного паттерну MVVM та із використанням ASP.WebApi якості сервісу постачання даних . Рівень доступу до бази даних було розроблено із використанням технології EntityFramework, яка легко інтегрується під програмні продукти сімейства Asp.Net. Основною мовою програмування було вибрано C#, а також середовище розробки Microsoft VisualStudio 2012 та для розгортання та тестування веб-додатку – InternetInforamtionServices 8 (IIS).

Мова програмування C# є об’єктно-орієнтованою та дає максимально зекономити час розробки програмно продукту. Дана мова програмування

забезпечує безпечну систему типізації. Основним призначенням є розробка програм під платформу .Net. Синтаксис подібний до C++ та до Java.

Технологія Asp.Net розроблена компанією Microsoft, для розробки веб-орієнтованих програмних продуктів. Передуючою технологією була ASP, котра склала більшу частину вже новішої технології ASP.Net.

Для розробки мобільного додатку було вибрано технологію Xamarinосновною перевагою цієї технології є можливість використання під різні платформи, що дає змогу водночас розробляти під ОС Android та IOS. Для впровадження кращої тестованості коду було вибрано архітектурнийпатерн MVVM, адже з його допомогою з легкістю можна розділити логіку що дає можливість нам писати модульні тести задля перевірки коректності роботи .

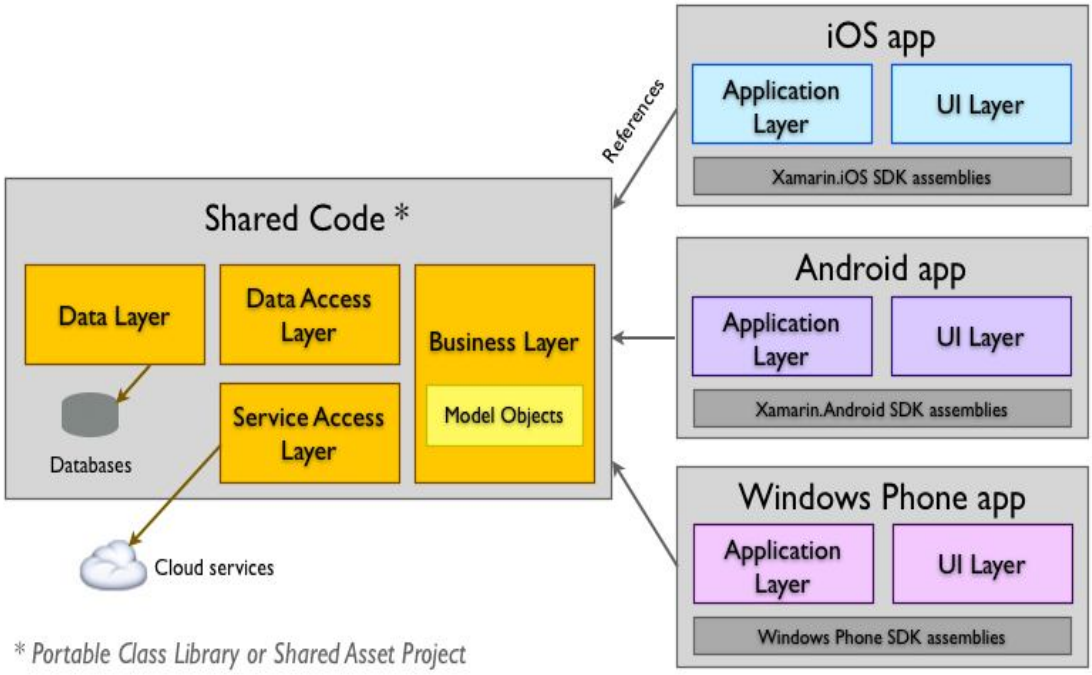

Рис 3.12 Архітектура системи на базі технології Xamarin.

Model-View-ViewModel (MVVM) - це шаблон проектування(рисунок 3.13), що застосовується під час проектування архітектури застосування (додатків). MVVM полегшує відокремлення розробки графічного інтерфейсу від розробки бізнес логіки (бек-енд логіки), відомої як модель (можна також сказати, що це відокремлення представлення від моделі). Модель

представлення є частиною, яка відповідає за перетворення даних для їх подальшої підтримки і використання. З цієї точки зору, модель представлення більше схожа на модель, ніж на представлення і обробляє більшість, якщо не всю, логіку відображення даних. Модель представлення може також реалізовувати патерн медіатор, організовуючи доступ до бек-енд логіки навколо множини правил використання, які підтримуються представленням.

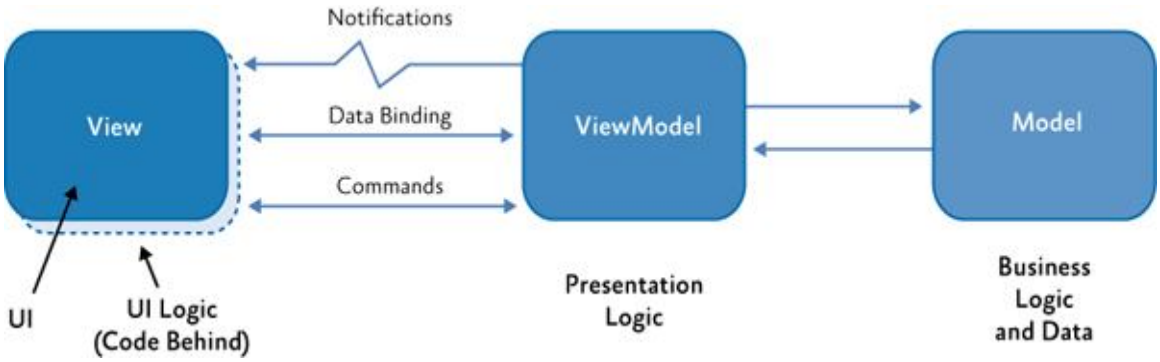

Рис.3.13. Архітектура MVVM

Розглянемо приклад реалізації контролера:

Клас AccountController

```
usingSystem.Web.Mvc;
usingDiplom.DataAccess;
usingDiploma.Models;
usingDomain.Abstraction;
usingDomain.Manager;
usingDomain.Models;

namespaceDiploma.Controllers
{
publicclassAccountController : Controller
   {
privatereadonlyAccountManager _accountManager;

publicAccountController(UnitOfWorkunity)
      {
         _accountManager = new
AccountManager(unity.UserRepository,unity.AddressRepository,unity.PermissionRepository);
      }

publicActionResultIndex()
      {
return Json ();

      }

publicActionResultRegister()
      {
return Json ();
```

```
        }

        [HttpPost]
publicActionResultRegister(UserRegisterModelmodel)
        {
if (ModelState.IsValid)
            {
if (!_accountManager.IsLoginInfoValid(model.UserName, model.Email))
                {
ModelState.AddModelError("", "EmailorLoginwasreserved");
returnView();
                }

var user = newUser
                {
UserName = model.UserName,
Email = model.Email,
Password = model.Password,
SurName = model.SurName,
Name = model.Name,
                };

                _accountManager.Register(user);

            }
return Json (Response);
        }
    }
}
```

В цьому класі використовуються методи, котрі в свою чергу повертають представлення клієнту, прикладом є метод Register та Index. У конструктор класу передається екземпляр UnitOfWork, що вміщує в собі екземпляри реопзиторіїв, які слугують для роботи із БД. Всередині конструктора створюється екземпляр AccountManager, який містить основну бізнес логіку. На рисунку 3.3 наведено UML діаграму класів рівня бізнес-логіки.

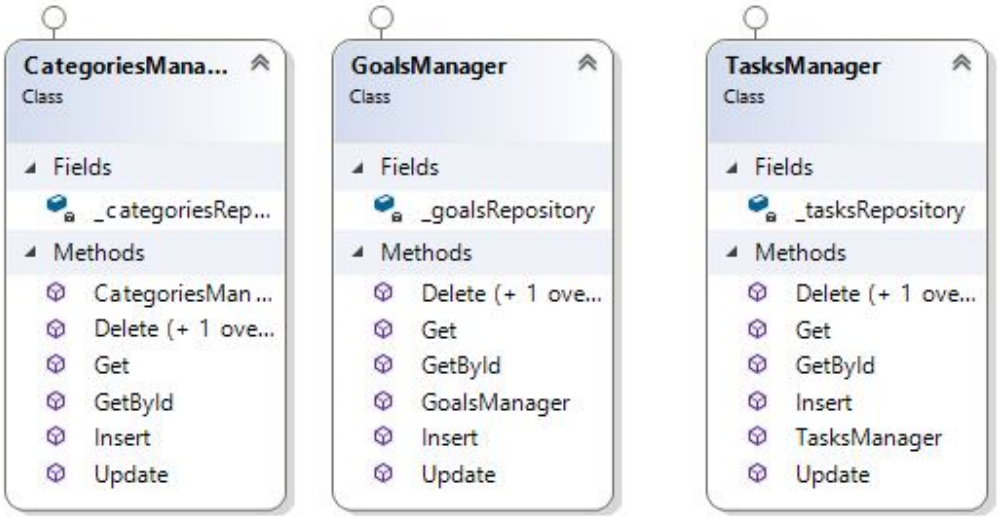

Рис.3.15 Діаграма класів рівня бізнес-логіки

Розглянемо фрагмент коду відображення:

Представлення Rgister

```
<phone:PhoneApplicationPage
   x:Class="LoginApp.Views.SignUpPage"
   xmlns="http://schemas.microsoft.com/winfx/2009/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
   xmlns:phone="clr-namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone"
   xmlns:shell="clr-namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone"
xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
   xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
FontFamily="{StaticResourcePhoneFontFamilyNormal}"
FontSize="{StaticResourcePhoneFontSizeNormal}"
Foreground="{StaticResourcePhoneForegroundBrush}"
SupportedOrientations="Portrait" Orientation="Portrait"
mc:Ignorable="d"
shell:SystemTray.IsVisible="True">

<!--LayoutRootistherootgridwhereallpagecontentisplaced-->
<Grid x:Name="LayoutRoot" Background="White">
<GridMargin="5,0,0,0">
<Grid.RowDefinitions>
<RowDefinitionHeight="Auto"/>
<RowDefinitionHeight="Auto"/>
<RowDefinitionHeight="Auto"/>
<RowDefinitionHeight="Auto"/>
<RowDefinitionHeight="Auto"/>
<RowDefinitionHeight="Auto"/>
<RowDefinitionHeight="Auto"/>
<RowDefinitionHeight="Auto"/>
</Grid.RowDefinitions>
<!--Title-->
<TextBlockText="UserRegistration :" Grid.Row="0" FontSize="40"  Foreground="Black"/>

<!--FirstName -->
<TextBlockText="FirstName:" Grid.Row="1" Foreground="Black"   Margin="0,25,0,0"/>
<TextBoxName="tFirstName    "   BorderBrush="LightGray"   Grid.Row="1"   Margin="100,0,0,0"
GotFocus="Txt_GotFocus"/>

<!--Password-->
<TextBlockText="Password: " Grid.Row="2" Margin="0,25,0,0" Foreground="Black" />
<PasswordBoxName="tPass"   BorderBrush="LightGray"   Grid.Row="2"   Margin="100,0,0,0"
GotFocus="pwd_GotFocus" />

<!--LastName -->
<TextBlockText="LastName: " Grid.Row="3" Margin="0,25,0,0" Foreground="Black" />
<TextBoxName="tLastName    "   BorderBrush="LightGray"   Grid.Row="3"   Margin="100,0,0,0"
GotFocus="Txt_GotFocus"/>

<!--PhoneZipCode -->
<TextBlockText="ZipCode: " Grid.Row="4" Margin="0,25,0,0" Foreground="Black" />
<TextBoxName="TxtPhNo" BorderBrush="LightGray" MaxLength="10"  InputScope="digits" Grid.Row="4"
Margin="100,0,0,0" GotFocus="Txt_GotFocus"/>

<!--Email-->
<TextBlockText="EmaiID: " Grid.Row="5" Margin="0,25,0,0" Foreground="Black" />
<TextBoxName="TxtEmail"   BorderBrush="LightGray"   Grid.Row="5"   Margin="100,0,0,0"
GotFocus="Txt_GotFocus"/>
```

```
        <!--SubmitButton-->
        <ButtonBorderBrush="Transparent"  Background="#FF30DABB"  Content="Submit"  Name="BtnSubmit"
Click="Submit_Click" Grid.Row="6"/>

        </Grid>
        </Grid>

        </phone:PhoneApplicationPage>
```

В цьому коді зображено форму реєстрації нового користувача, де використовується мова розмітки Xam. Для побудови зв'язку між модулем та представленням використовуються спеціальні атрибути такі як x:Class="LoginApp.Views.SignUpPage"

Рівень доступу до БД був розроблений на базі архітектурного паттерну репозиторій із модифікаціями під UnitOfWork, що в свою чергу дає можливість гнучкого масштабування продуту та полегшує процес розробки. Особливістю паттерну репозиторій полягає у доступі до записів, ми використовуємо методи котрі, в свою чергу, вміщають логіку по маніпуляції над даними, а сам клас із методами містить доступ, або саме сховище даних.

Розглянемо архітектуру доступу до БД в деталях. На рисунку 3.4 наведено UML діаграму класів на рівні доступу до даних.

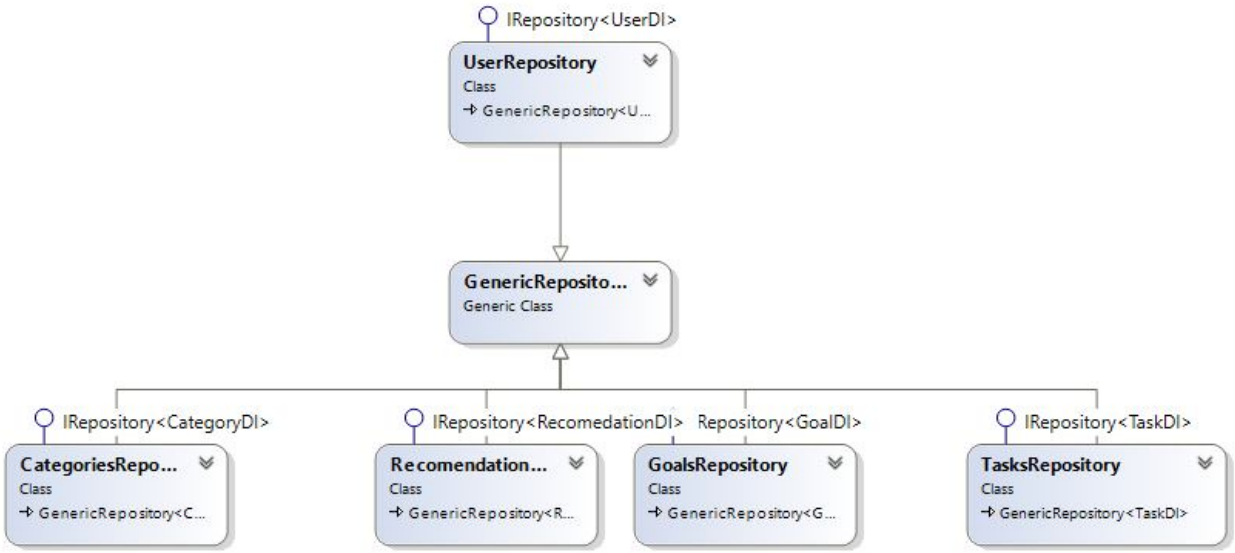

Рис.3.15 UML діаграма класів рівня доступу до даних

Як можна побачити з рисунку 3.15, існує основний клас із реалізацією методів взаємодії із БД, а в дочірніх класах є реалізація методів для взаємодій із рівнем бізнес логіки. Програмний код основних класів кожного із рівнів представлений у додатку А.

3.2. Проектування структури бази даних

Проаналізувавши вербальний опис можна збудувати діаграму елементів та зав’язків (див. рис. 3.16).

На базі отриманої діаграми можна приступати до проектування елементів у БД та побудови ER-діаграми. Взявши усі сутності та звязки ми переносимо їх у діаграму сутностей та звязків, де інформаційними сутностями виступають елементи із діаграми елементів та звязків, котрі містять у собі інші сутності. Отже, можна виділити такі інформаційні сутності як Task, Category, User, Goal, Recomendation.

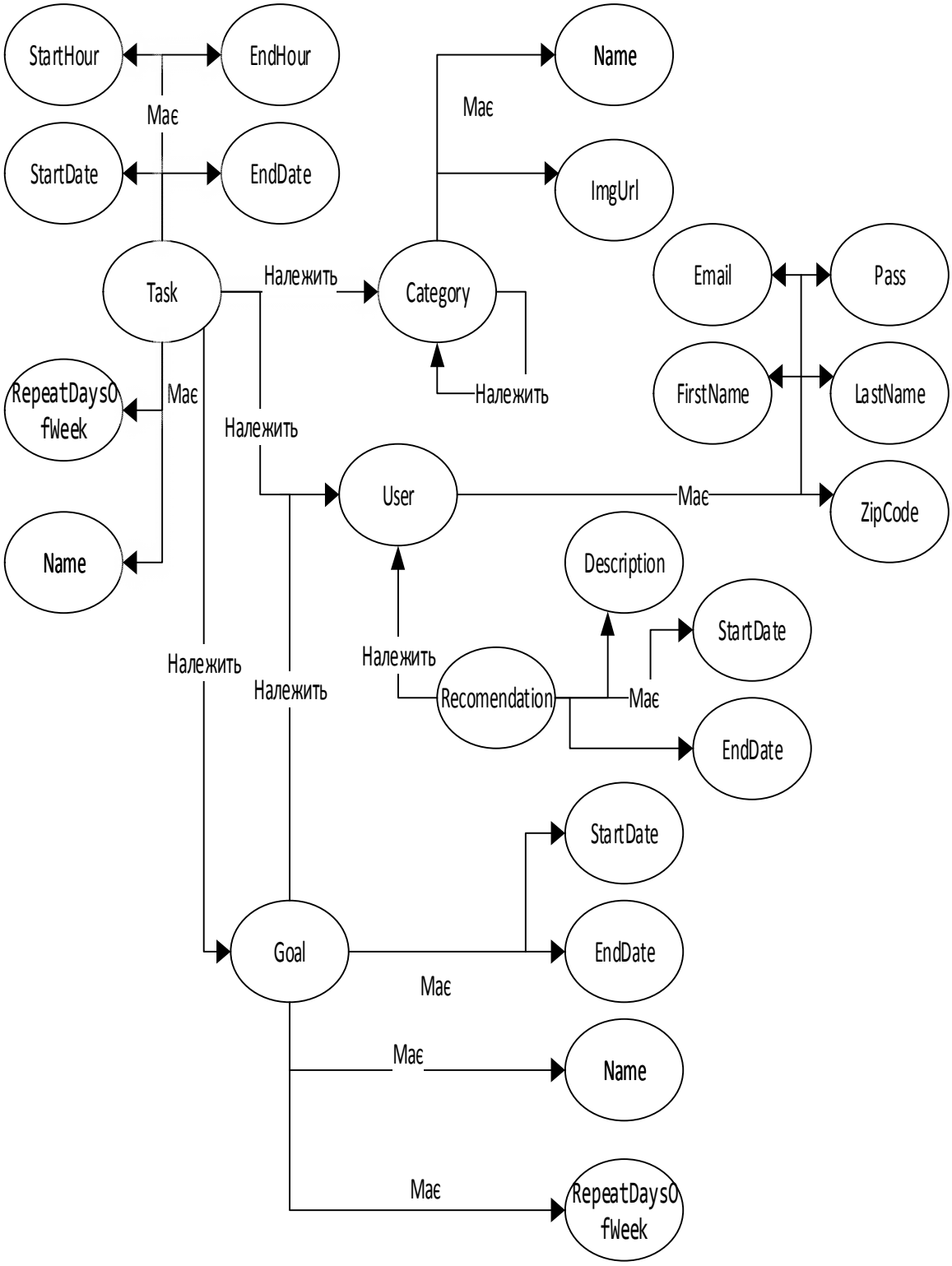

Рис. 3.16 Діаграма елементів і зв’язків

Опис ідентифікаторів та планування індексів представлено у таблицях 3.1 і 3.2 відповідно.

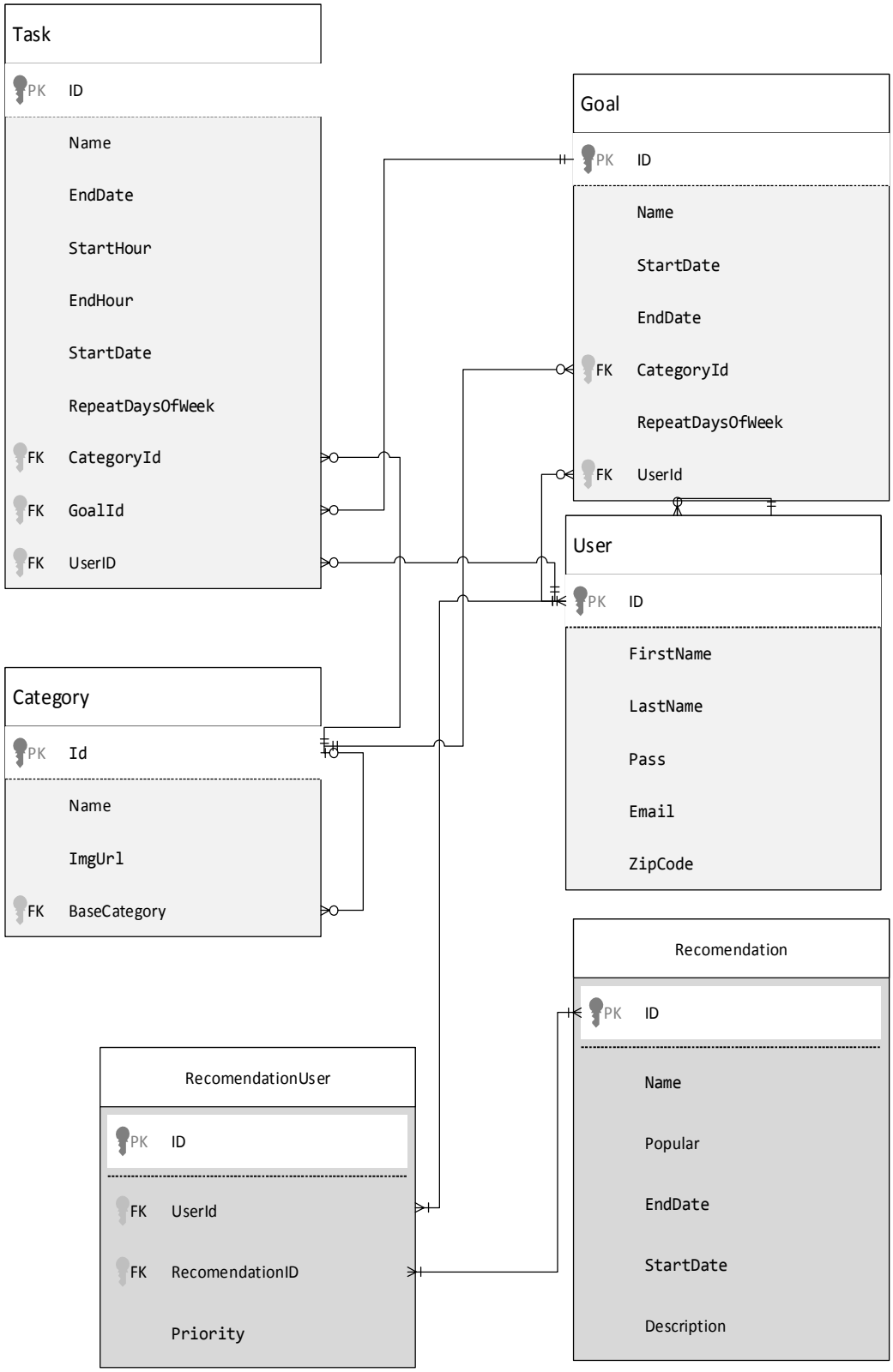

Рис. 3.17 ER-діаграма

Таблиця 3.1

Таблиця ідентифікаторів

Об’єкт	Властивість	Тип	Розмірність	Ідентифікатор
User	Name	varchar	45	Name
	Surname	varchar	45	Surname
	Email	varchar	45	Mail
	Username	varchar	45	Username
	Password	varchar	45	Password
Category	Name	varchar	45	Name
	ImgUrl	varchar	45	ImgUrl
Recomendation	Name	varchar	45	Name
	Popular	double	45	Popular
	EndDate	date		EndDate
	StartDate	date		StartDate
	Descriptrion	varchar	300	Descriptrion
Task	Name	varchar	45	Name
	EndDate	date		EndDate
	StartDate	date		StartDate
	EndHour	date	45	EndHour
	StartHour	date	45	StartHour
	RepeatDaysOfWeek	int		RepeatDaysOfWeek
Goal	Name	varchar	45	Name
	EndDate	date		EndDate
	StartDate	date		StartDate
	RepeatDaysOfWeek	int		RepeatDaysOfWeek

Таблиця 3.2

Таблиця планування індексів

Назва таблиці	Атрибут	Опис
Task	UserID	Foreignkey
Goal	UserID	Foreignkey
Task	GoalID	Foreignkey
Task	CategoryID	Foreignkey
User	Permission_ID	Foreignkey
Recomendation	UserID	Foreignkey

Доступ до БД з використанням технології entityfreamworkбуде здійснюватися за допомогою спеціальних проксі класів, що суттєво полегшить роботу. Це дасть змогу уникнути написання Sql коду та пришвидшить розробку.

Кожній інформаційній сутності буде створений проксі клас із урахуванням усіх обмежень та полів, які зображені у діаграмі зав'язків та сутностей (рис. 3.17). Також із допомогою вистроєного функціоналу віртуалізації даних та побудови графи змін, ми зможемо пришвидшити роботу програми.

По закінченню будуть створенні C# класи із необхідних полів та зав'язків (рис. 3.18).

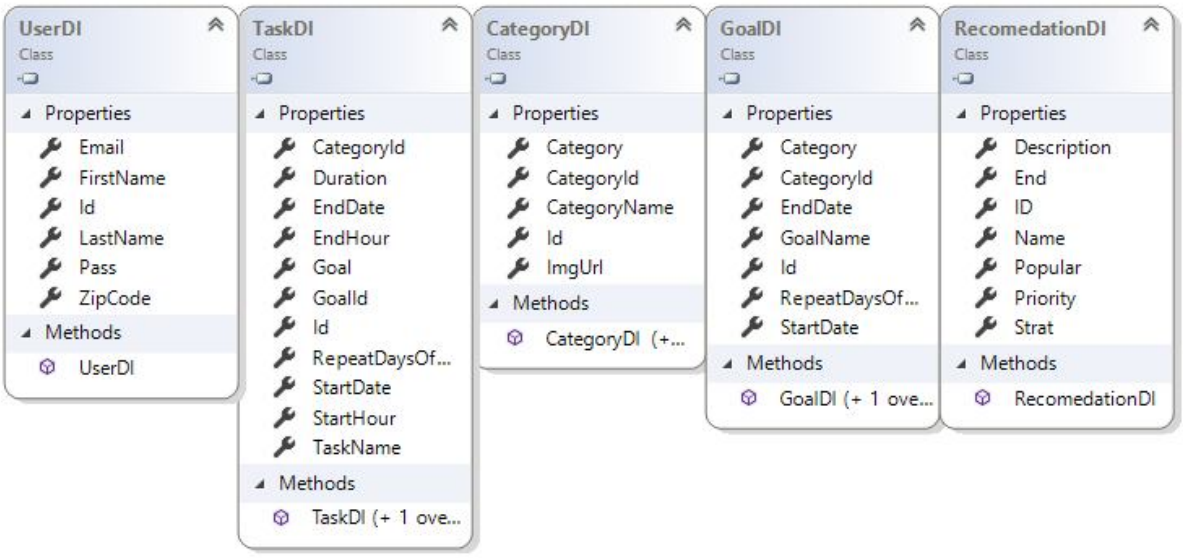

Рис. 3.18. Діаграма класів «Проксі класів»

3.3. Програмування бази даних

Для реалізації баз даних було використано технологію EntityFrameworkcodefirst, що дало змогу конвертувати вже існуючі класи рівня доступу до бази даних у таблички. Дана технологія була розроблена компанією Microsoft. Для формування схеми бази даних використовуються класи написані із допомогою мови програмування C# або VisulBasic.

Базою даних є SQL Server 2008 R2. Основною відмінністю від попередніх СУБД є підтримка структурованих і напівструктурованих даних. Також ця версія на відміну від попередніх має більш кращу систему налаштування використання ресурсів, що дає змогу витримувати більше навантаження та економити витрати. Виділяють засоби управління на основі політик, розширені можливості з складання звітів і проведенню аналізу, а також розвинені засоби управління інтелектуальними ресурсами підприємства.

Для формування нової бази даних потрібно створити клас та унаслідувати його від базового класу контексту, котрий надається в збірці EntityFramework. Таблички із сутностями зображені у вигляді колекції типу DBSet. Для прикладу розглянемо реалізації контексту.

Клас DatabaseContext

```
using System;

namespace TimeZilla.DataLayer.Database
{
    public class DatabaseContext
    {
        static DatabaseContext()
        {
Activator.CreateInstance<DbInitializer>();
        }

        internal DbSet<TEntity> Set<TEntity>() where TEntity : class,new()
        {
            return new DbSet<TEntity>();
        }

    }
}
```

У даному прикладі в конструкторі відбувається встановлення методу, котрий при першому створенні бази даних буде виконувати та записувати

базові значення в базу даних, приклад реалізації даного класу наведений нижче. Також можна побачити основні таблички. Вони записані у вигляді властивостей та мають тип DbSet.

3.4 Тестування та дослідна експлуатація

Етап тестування є одним із найважливіших етапів у розробці програмного забезпечення, адже помилки в роботі програми можуть призвести до великих матеріальних витрат. За способом виконання тестів розрізняють автоматизоване та ручне тестування. Основна відмінність між цим підходами полягає у самому способі тестування програмного продукту: при автоматизованому підході використовуються спеціальні програмні продукти типу Silenium та Ranorex. Написання автоматизованих тестів є клопіткою справою та потребує багато часу та грошей, адже потрібно найти спеціалістів із правильними навичками, або самому потратити час на дослідження даної області.

На відміну від автоматизованого тестування, ручне тестування скоротить суттєво витрати на етапі тестування і зекономить час. Проте в майбутньому при розширенні програмного продукту ми стикнемось із проблемою повторного тестування, що призведе то повторного виконання тієї ж роботи, яка при автоматизованому тестуваннізайме декілька хвилин. Але при кардинально сильних змінах представлення програми або функціонал буде потребувати супровід тих тестів.

На сьогодні тестування програмного забезпечення – один з найдорожчих етапів життєвого циклу програмного забезпечення, на нього відводиться від 45% до 55% загальних витрат. Зазвичай, для проведення тестування застосовуються методи структурного («білий ящик») та функціонального («чорний ящик») тестування [7]. Можна виділити наступні методи тестування:

- модульне тестування;

- функціональне тестування;
- тестування графічного інтерфейсу;
- тестування безпеки.

Першим етапом тестування було модульне. Суть модульного тестування полягає в окремому тестуванні кожного модуля коду програми. Для проведення модульного тестування було використано програмний продукт Microsoft VisualStudio, в якому було створено тестовий проект консольного типу, до якого підключено усі модулі та вручну було проведено тестування, суть якого у написанні програмного коду та перевірки отриманого результату.

При функціональному тестуванні вихідний код програми недоступний. Суть полягає в перевірці відповідності роботи програми до її специфікації. Критерієм повноти слід вважати перебір всіх можливих вхідних даних, що практично здійснити надзвичайно важко.

Метою тестування веб-орієнтованого додатку є знайти його слабкі місця у безпеці та недоліки відображення. Наданий момент є велика кількість браузерів, котрі мають різні функції та по-різному трактують вже існуючі елементи, тому слід перевірити правильність відображення програми на різних браузерах, тобто провести кросбраузерне тестування.

Після проведення функціонального тестування було виправлено ряд суттєвих дефектів, які впливали на функціональність системи в цілому.

У таблиці 3.3 наведено опис дефектів, що були виявлені під час функціонального тестування.

Таблиця 3.3

Дефекти, які виявлені під час функціонального тестування

Дефект	Опис дефекту	Вирішення дефекту
При введенні пустої інформації про подію, система опрацьовує ті дані і зберігає у базі даних	Дефект полягає у тому що система повинна проводити перевірку даних, щоб не призвести до пошкодження цілісності даних	Додавання перевірки даних перед збереженням.
При авторизації користувачам не видно функції «Рекомендації»	Дефект полягає у тому що є проблеми у стилях компонента	Виправлення атрибуту стиля
Можна зареєструвати користувача із вже існуючим логіном	Дефект полягає у тому, що немає перевірки на існування вже існуючого користувача	Потрібно добавити перевірку при додаванні на існування користувача із такою інформацією

Також надзвичайно важливу роль в розробці додатків відіграє безпека, адже при дефектах у безпеці можна понести великі матеріальні збитки. Тому даний етап тестування потребує певних навичок у галузі інформаційної безпеки, що дасть змогу виявити потенціальні загрози.

Для тестування безпеки ресурсу було використано програмний продукт Fiddler. Основний інтерфейс даної програми представлений на рисунку 3.20.

Для тестування безпеки ми модифікуємо протоколи, котрі приходять до користувача, обходячи систему перевірки на браузері (рис. 3.21).

Результат тестування веб-орієнтованої програмної системи є позитивним.

Наступним етапом тестування була перевірка правильності відображення інтерфейсу користувача. Через існування великої кількості браузерів, це є досить важкою роботою, адже потрібно переглядати правильність відображення у браузерах заданими при формуванні вимог. Також деякі браузери не мають можливостей як інші, тому потрібно розробляти програмний продукт, котрий буде враховувати усі ці аспекти.

Під час тестування відображення було виявлено ряд дефектів представлених у таблиці 3.21.

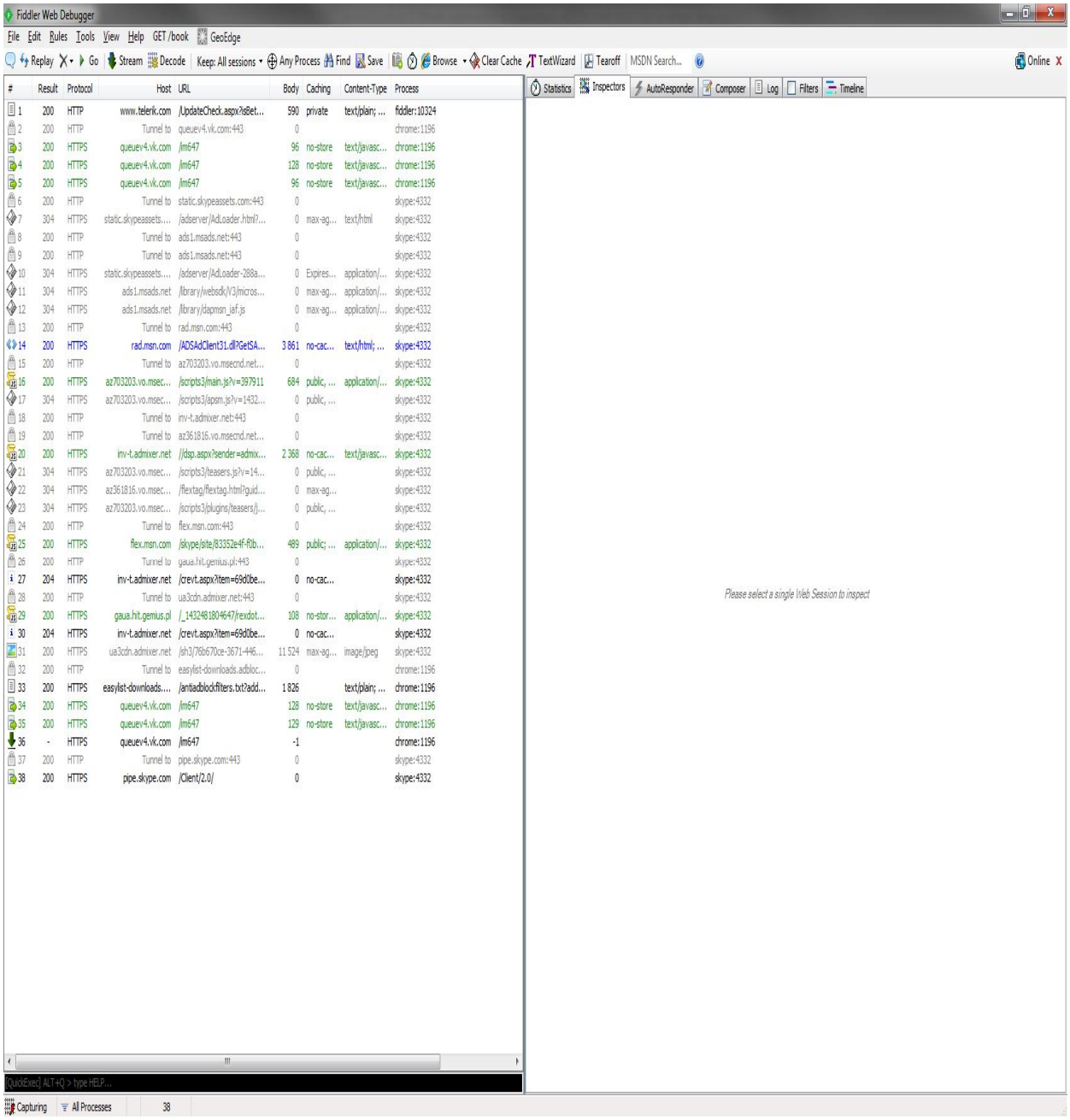

Рис. 3.20 Основне вікно роботи із Fiddler

Таблиця 3.5

Дефекти, які виявлені під час тестування інтерфейсу

Дефект	Опис дефекту	Вирішення дефекту
При відображені картинки категорії рамка пропадає	Дефект полягає у неправильному конвертері пріоритетів відображення	Потрібно виправити конвертер пріоритетів відображення та встановити правильну стратегію відображення
При відображенні всього списку подій, нижні елементи відображаються під кнопками	Дефект полягає у тому що не правильно пораховані розміри контейнера.	Потрібно від корегувати обмеження для розмірів.

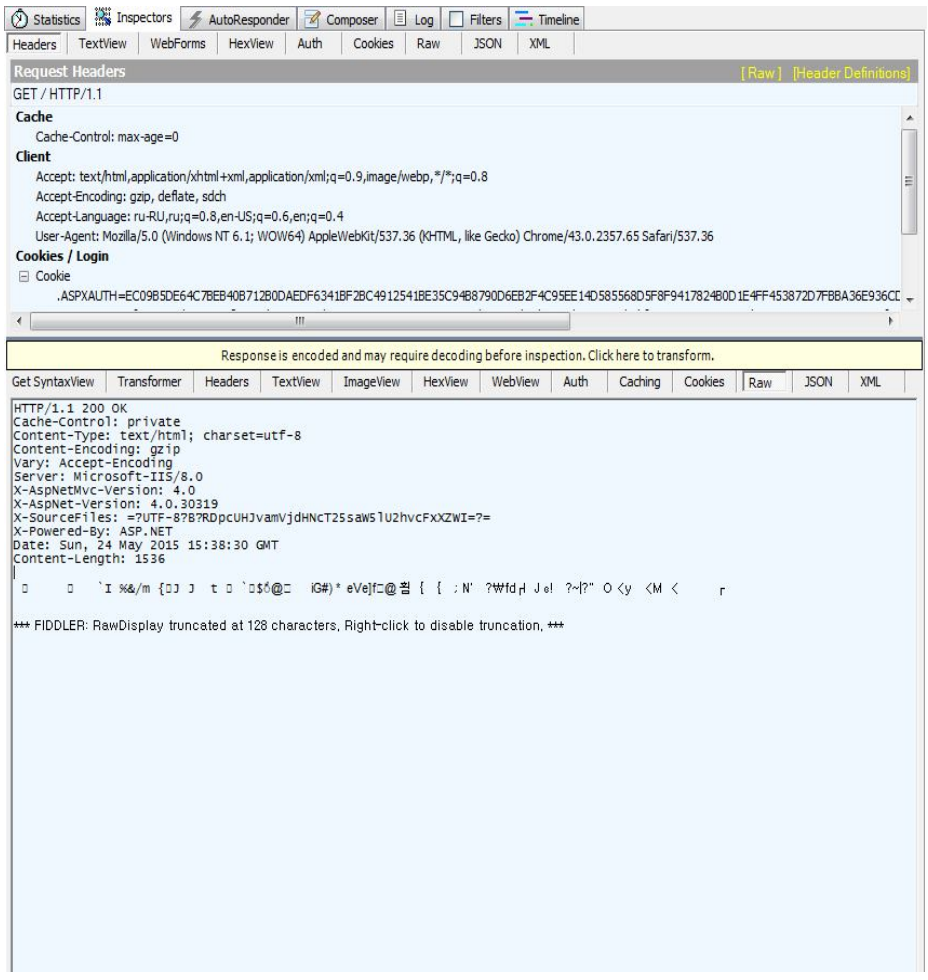

Рис. 3.21 Вікно модифікації протоколу.

Після проведення тестування інтерфейсу було виправлено ряд суттєвих дефектів, які впливали на відображення системи.

3.5 Розгортання програмного продукту

Для розгортання програмної системи необхідно задовольнити вимоги, що наведені нижче:

- Для веб сервісі:
  - операційна система Microsoft Windows 8.1/10;
  - SQL Server 2012;
  - web-сервер IIS 7/8;
  - Microsoft .NET Framework4.5
- Для мобільного додатку:
  - операційна система Android 6.0 Marshmallow

Для формування апаратних вимого потрібно провести аналіз кількості користувачів, котрі будуть використовувати розроблену систему. Для невеликої кількості (до 1000 одночасно) користувачів достатньо наступної конфігурації для апаратних ресурсів:

- Для веб сервіса:
  - процесор з тактовою частотою не менше 2 Ghz;
  - оперативна пам’ять в межах 16Gb;
  - система потребує великої кількості місця на жорсткому диску для зберігання бази даних та зображень. Тому об’єм жорсткого диску має бути не менше 1Tb;
  - рекомендована швидкість з’єднання становить від 100 Mb/s.
- Для мобільного додатку:
  - процесор з тактовою частотою не менше 2 Ghz;
  - оперативна пам’ять в межах 2Gb;
  - необхідно пам’яті для розгортання 256мб

Для розгортання системи на сервері, необхідно попередньо її збільшити

та перенести усі модулі програмного продукту у відповідну папку на серверів. Після перенесення всіх елементів системи слід вказати, що ця папка є джерелом веб-сайту, клікнувши правою клавішею миші і вибравши функцію «Addweb-site». Після встановлення папки як веб-сайт, появиться діалогове вікно, де потрібно встановити шлях до папки із контентом програми. Після проходження усіх попередніх кроків необхідно опублікувати веб-сайт за допомогою функції «Publish» в Microsoft Visual Studio.для публікації.

Для розгортання мобільного додатку, необхідно підключити пристрій до комп'ютера та перемістити файл для інсталяції на мобільний пристрій. Після переносу додатку слід відкрити інсталяційний файл з мобільного пристрою. В результаті буде проведена збереження.

Для роботи програмно продукту потрібно налаштувати середовище згідно попередніх вимог. Також середовище роботи повинно включати в собі бібліотеки наведені в таблиці 3.5.

Таблиця 3.5

Набір бібліотек для коректної роботи програми

№	Назва бібліотеки	Призначення
1.	System.Web.Mvc.dll	головна бібліотека фреймворку ASP.NET MVC
2.	System.Web.Razor.dll	бібліотека для роботи з формуванням інтерактивного та гнучкого користувацького інтерфейсу
3.	TimZilla.Domain.dll	бібліотека для визначення сутностей предметної області
4.	TimZilla.DAL.dll	бібліотека для реалізації доступу до даних
5.	TimZilla.BLL.dll	бібліотека для реалізації всієї бізнес-логіки
6.	Web.config	файл налаштувань програмної системи

Для роботи системи та можливості доступу до бази даних необхідно вказати стрічку з'єднання із сервером бази даних в файлі Web.config. Приклад з'єднання стрічки наведений нижче:

<addname="DataBaseContext" providerName="System.Data.SqlClient" connectionString="Server=.;Database=TimeZilla; IntegratedSecurity=True; MultipleActiveResultSets=True" />

Програмний код основних класів кожного із рівнів представлений у додатку А.

Рис. 3.22 Вікно перегляду рекомендацій по категорії «Концерти»

Рис. 3.23 Вікно перегляду інформації про рекомендацію

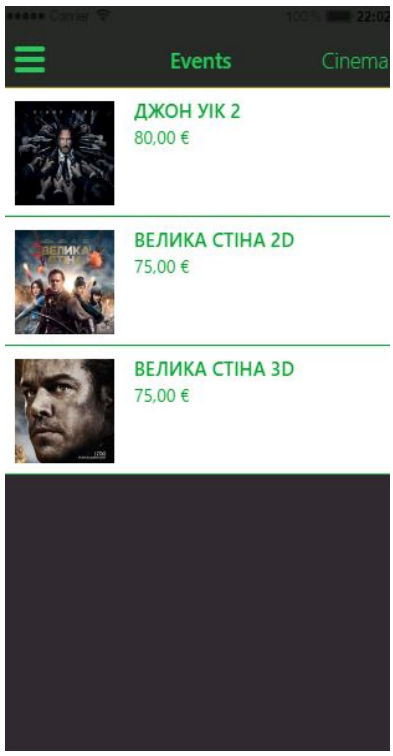

Рис. 3.22 Вікно перегляду рекомендацій по категорії «Кіно»

Висновки до 3 розділу

1. Спроектовано та розроблено програмну систему «TimeZilla».
2. Проведено модульне, інтеграційне, функціональне та тестування графічного інтерфейсу.
3. Розроблено технологічну інструкцію для користувача та системного адміністратора.

ВИСНОВКИ

В результаті виконання магістерської роботи була розв’язана задача розробки програмних засобів вирішення задачі тайм менеджменту із використанням алгоритмів ранжування для формування рекомендацій Розроблений модуль надає простий механізм вирішення задач тайм менеджменту, що дозволяє швидко будувати різні плани. Він також надає можливості генерацій рекомендацій враховуючи побажання користувача та можливість інтеграції у готові комплексні системи.

Розроблений продукт «TimeZilla» покращить спосіб формування планів та власних цілей. Зручний інтерфейс дає можливість користувачеві слідкувати за виконанням поставлених цілей та розроблена система рекомендацій дасть можливість формувати найбільш коректніший графік.

Для досягнення мети в магістерській роботі вирішено наступні задачі:

1) Проведено аналіз та порівняння існуючих систем генерації рекомендацій та встановлено, що вони мають ряд недоліків: орієнтовані під конкретні системи мають обмежені механізми із роботою різних типів даних, не пристосовані до варіації шумів . Основна проблема таких засобів полягає у відсутності системи налаштування згідно потреб користувача. Визначено основні проблеми, які виникають в процесі генерації рекомендацій. На основі зробленого аналізу недоліків існуючих систем здійснено постановку задачі дослідження для підвищення ефективності генерації рекомендацій .

2) Проведено теоретичні дослідження основних підходів, які відносяться до формалізації предметної області. Запропонований механізм вирішення задач тайм менеджменту, а саме процедура генерації рекомендацій із використанням методу опорних векторів

3) На основі результатів аналізу функціональних, нефункціональних та архітектурних вимог здійснено проектування та програмну реалізацію розроблених методів та алгоритмів, та системи в цілому, проведене відповідне

тестування (модульне, інтеграційне, функціональне, тестування графічного інтерфейсу).

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Kernel perecptron [Електронний ресурс]. – Режим доступу: https://en.wikipedia.org/wiki/Kernel_perceptron
2. Bloesch, A. & Halpin, T. 1997, ‘Conceptual queries using ConQuer-II’, Proceedings of the 16th International Conference on Conceptual Modeling ER'97 (Los Angeles), Springer LNCS 1331 (Nov.) 113- 126.
3. Data mining [Електронний ресурс]. – Режим доступу: https://en.wikipedia.org/wiki/Data_mining
4. Halpin, T.A. & Bloesch, A.C. 1999, ‘Data modeling in UML and ORM: a comparison’, Journal of Database Management, vol. 13, no. 2, Idea Group Publishing, Hershey PA, pp. 20-30.
5. Halpin, T.A. & Ritson, P.R. 1992, ‘Fact-oriented modelling and null values’, Research and Practical Issues in Databases, eds B. Srinivasan & J. Zeleznikov, World Scientific, Singapore.
6. Halpin, T.A. 1998, ‘ORM/NIAM Object-Role Modeling’, Handbook on Information Systems Architectures, eds P.Bernus, K. Mertins & G. Schmidt, Springer-Verlag, Berlin, pp. 81-101.
7. IEEE Std. 610.12:1990. IEEE Standard Glossary of Software Engineering Terminology.
8. Spectral clustering [Електронний ресурс]. – Режим доступу: https://en.wikipedia.org/wiki/Spectral_clustering
9. Линейныйний классифаикатор [Електронний ресурс]. – Режим доступу: https://ru.wikipedia.org/wiki/Линейный_классификатор
10. Метод опорных векторов [Електронний ресурс]. – Режим доступу: https://ru.wikipedia.org/wiki/Метод_опорных_векторов
11. Ranking SVM [Електронний ресурс]. – Режим доступу: https://en.wikipedia.org/wiki/Ranking_SVM
12. База даних [Електронний ресурс]. – Режим доступу: http://www.unicyb.kiev.ua/~boiko/pr2k/dbintro.htm

13. Берко А. Ю., Верес О. М., Пасічник В. В. Системи баз даних і знань. Книга 1. Організація баз даних і знань : навч. посіб. Львів : Магнолія, 2011. – 456 с.

14. Бізнес-процес [Електронний ресурс]. – Режим доступу: http://www.rusnauka.com/6_PNI_2012/Economics/6_102471.doc.htm

15. Білас О. Є. Якість програмного забезпечення та тестування : навч. посібн. - Львів : Видавництво Львівської політехніки, 2013. - 216 с.

16. Вищий навчальний заклад [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Вищий_навчальний_заклад

17. Вікіпедія [Електронний ресурс]. – Режим доступу: http://uk.wikipedia.org/wiki/

18. Дивак М. П., Шпінталь М.Я., Шевчук Р.П., Козак О.Л., Пукас А.В., Спільчук В.М., Гончар Л.І. Методичні рекомендації до виконання та захисту дипломної роботи на здобуття освітньо-кваліфікаційного рівня магістр за спеціальностями: 8.05010301 “Програмне забезпечення систем” та 8.05010302 “Інженерія програмного забезпечення” // Тернопіль : Економічна думка, 2011. – 31 с.

19. Дин Леффингуэлл, Дон Уидриг. Принципы работы с требованиями к программному обеспечению. Унифицированный подход. – М.: Вильямс, 2012. – 448 с.

20. Joachims, T. (2003) Optimizing Search Engines using Clickthrough Data // Proceedings of the ACM Conference on Knowledge Discovery and Data Mining

21. Коротун Т.М. Совершенствование процесса тестирования ПО // Проблемы программирования. – 1998. - № 3 – С.59-64.

22. Майерс Г. Искусство тестирования программ. – М: Финансы и статистика. - 1982. – 176 с.

23. Support Vector Machine algorithm [Електронний ресурс]. – Режим доступу: http://www.machinelearning.ru/wiki/index.php?title=%D0%9C%D0%B0%D1%88%D0%B8%D0%BD%D0%B0_%D0%BE%D0%BF%D0%BE%D1%80%D0%BD%D1%

8B%D1%85_%D0%B2%D0%B5%D0%BA%D1%82%D0%BE%D1%80%D0%BE%D0%B2

24. Павловская Т. C#. Програмирование на языке высокого уровня: Учебник для вузов. - СПб. : Питер, 2010. – 432 с.

25. Тарасов О. В., Федько В. В, Лосєв М. Ю. Організація баз даних та знань. Проектування баз даних : навч.-практ. Посіб. - Х. : ХНЕУ, 2011. – 200 с.

26. Томас Х. Кормен, Чарльз И. Лейзерсон, Рональд Л. Ривест, Клиффорд Штайн. Алгоритмы: построение и анализ. - 2-е изд. - М.: «Вильямс», 2006. - 1296 с.