МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

Факультет Магістерської підготовки

Кафедра Інформаційних технологій

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка програмного комплексу для розрахунку та візуалізації складових водно-сольового балансу водойм»

Виконав студент 2 курсу групи МІС-19
спеціальності 122 Комп'ютерні науки

Ліхачов Кирило Дмитрович

Керівник к.геог.н., доц.
Кузніченко Світлана Дмитрівна

Рецензент к.т.н., доц.
Гнатовська Ганна Арнольдівна

Одеса 2020

МІНІСТЕРСТВО ОСВІТИ І НАУКИУ УКРАЇНИ
ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

Факультет Магістерської підготовки
Кафедра інформаційних технологій
Рівень вищої освіти магістр
напрямок 122 Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри _____
_____
" 26 " жовтня 2020 р.

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Ліхачову Кирилу Дмитровичу
(прізвище, ім'я, по батькові)

1. Тема роботи «Розробка програмного комплексу для розрахунку та візуалізації складових водно-сольового балансу водойм»
керівник роботи к.геогр.н., доцент Кузніченко Світлана Дмитрівна
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом закладу вищої освіти від "16" жовтня 2020р. №194 «С»
2. Строк подання студентом проекту 7 грудня 2020 р.
3. Вихідні дані до роботи методика розрахунку складових водного і сольового балансів водойми Катлабух
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Аналітична частина
1.3 Модель водно-сольового балансу водойми
2. Проектна частина
2.2 Проектування інтерфейсу користувача ПП
2.3 Проектування сховища даних
2.4 Проектування архітектури ПП
2.5 Проектування компонентів, що реалізують розрахунок
3. Проектні та технічні рішення

5. Перелік графічного матеріалу
1 Модель розрахунку водного і сольового балансів оз. Катлабух
2 Діаграма варіантів використання ПП
3 Діаграма сутностей БД
4 Діаграма пакетів ПП

6. Консультанти розділів проекту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання "26" жовтня 2020 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту	Термін виконання етапів проекту	Оцінка виконання етапу	
			у %	за 4-х бальною шкалою
1	Дослідження літературних джерел за темою кваліфікаційної роботи	26.10	100	відмінно
2	Дослідження моделі водно-сольового балансу водойми	30.10	100	відмінно
3	Аналіз методики розрахунку складових водно-сольового балансу водойми	05.11	100	відмінно
4	Формування програмних та технічних вимог до програмної системи	10.11	100	відмінно
5	Обґрунтування вибору програмних засобів розробки	15.11	100	відмінно
6	Рубіжна атестація	19.11	100	відмінно
7	Проектування програмної системи розрахунку водно-сольового балансу водойми	23.11	100	відмінно
8	Опис технічних рішень розробки	30.11	100	відмінно
9	Опис програмного продукту. Аналіз результатів дослідження. Формування висновків	03.12	100	відмінно
	Подання роботи на кафедру	07.12		
	Перевірка на плагіат	08.12		
	Рецензування	16.12		
	Інтегральна оцінка виконання етапів календарного плану (як середня по етапам)		**100**	**відмінно**

Студент _____ Ліхачов К. Д.
(підпис) (призвище та ініціали)

Керівник роботи _____ Кузніченко С. Д.
(підпис) (призвище та ініціали)

АНОТАЦІЯ

на магістерську кваліфікаційну роботу

«Розробка програмного комплексу для розрахунку та візуалізації складових водно-сольового балансу водойм»,

студента Ліхачова Кирила Дмитровича

Кваліфікаційна магістерська робота: 79 с., 11 табл., 20 рис., 20 джерел.

Ключові слова: програмна система, моделювання, водно-сольовий баланс, об’єктна-орієнтоване програмування, C#, SQLITE.

Мета дослідження – розробка програмного комп’ютерного комплексу для моделювання складових водних і сольових балансів озера Катлабух за різних режимів його водогосподарського використання.

Об’єкт дослідження – модель водно-сольового балансу озера Катлабух.

Предмет дослідження – методи та алгоритми моделювання складових водно-сольового балансу озера Катлабух.

Задачі дослідження: провести дослідження моделі розрахунку водно-сольового балансу; сформувати програмні та технічні вимоги до програмного продукту, сховища даних і інтерфейсу користувача; обґрунтувати вибір програмних засобів створення програмного продукту і його компонентів; виконати проектування модулю розрахування водно-сольових балансів, сховища даних і архітектури програмного продукту; провести тестування основних компонентів компонентів програмного продукту.

Результати, їх новизна, теоретичне та практичне значення: розроблений програмний комплекс дозволяє ефективно розрахувати водно-сольовий режим озера Катлабух, графічно представити одержані результати, виконати моделювання водного балансу водойми за різних умов її водогосподарської експлуатації з метою надання обґрунтованих техніко-економічних рекомендацій стосовно приведення озера до оптимального гідроекологічного стану.

SUMMARY

for a master's degree

"Development of a Software Package for Calculation and Visualization of Components of the Water-Salt Balance for Water Bodies",

students of Kyrylo Likhachov

Qualifying master's thesis: 79 pages, 11 tables, 20 figures, 20 sources.

Keywords: software system, modeling, water-salt balance, object-oriented programming, C #, SQLITE.

The purpose of the study is to develop a software computer system for modeling the components of water and salt balances of Lake Katlabukh under different modes of its water use.

The object of research is a model of water-salt balance of Lake Katlabukh.

The subject of research – methods and algorithms for modeling the components of the water-salt balance of Lake Katlabukh.

Research objectives: to conduct a study of the model for calculating the water-salt balance; to form software and technical requirements to the software product, data warehouse and user interface; justify the choice of software to create a software product and its components; perform the design of the module for calculating water-salt balances, data storage and software architecture; to test the main components of the components of the software product.

Results, their novelty, theoretical and practical significance: the developed software allows to effectively calculate the water-salt regime of Lake Katlabukh, graphically present the results, model the water balance of the reservoir under different conditions of its water management in order to provide sound technical and economic recommendations for bringing the lake to the optimal hydroecological condition.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

БД – база даних

ВБВЧ – водний баланс витратна частина

ВБПЧ – водний баланс прибуткова частина

ДГРШ – диспетчерський графік роботи шлюзів

ПЗ – програмне забезпечення

ПП – програмний продукт

РВСБ – розрахунок водно-сольового балансу

СБВЧ – сольовий баланс витратна частина

СБПЧ – сольовий баланс прибуткова частина

СКБД – система керування базами даних

ТЗ – технічне завдання

.netmodule – формат файлу, що створюється компілятором MSVC з параметром / LN (створити модуль MSIL) або компоновщиком з / NOASSEMBLY (створення модуля MSIL).

API – (Application Programming Interface) інтерфейс програмування, інтерфейс створення додатків.

AVX – розширення системи команд x86 для мікропроцесорів Intel і AMD, запропоноване Intel в березні 2008

Bitbucket – веб-сервіс для хостингу проектів і їх спільної розробки, заснований на системах контролю версій Mercurial і Git

BitKeeper – це програмний інструмент для розподіленого контролю версій вихідного коду комп'ютера.

CVS – Concurrent Versions System, централізована система управління версіями

Direct3D – набір API функцій для взаємодії з відео картою

DirectCompute – API, який входить до складу DirectX (набору API від Microsoft), який призначений для роботи на IBM PC-сумісних комп'ютерах під управлінням операційних систем сімейства Microsoft Windows

DirectX – це набір API, розроблених для вирішення завдань, пов'язаних з програмуванням під Microsoft Windows. Найбільш широко використовується при написанні комп'ютерних ігор

ECMA - 262 3rd Edition -December 1999 – стандарт, який визначає мову програмування загального призначення ECMAScript

JSON (JavaScript Object Notation) – текстовий формат обміну даними, заснований на JavaScript

Mercurial – кроссплатформена розподілена система керування версіями, розроблена для ефективної роботи з дуже великими репозиторіями коду

Microsoft Visual Studio – лінійка продуктів компанії Microsoft, що включають інтегроване середовище розробки програмного забезпечення і ряд інших інструментальних засобів

Mono – проект зі створення повноцінного втілення системи .NET Framework на базі вільного програмного забезпечення

MySQL – вільна реляційна система управління базами даних

Perforce – комерційна система управління версіями

SGML, Standard Generalized Markup Language – стандартний узагальнений мову розмітки

Silverlight – це програмна платформа для написання і запуску багатофункціональних інтернет-додатків RIA

SSE, Streaming SIMD Extensions – потокове SIMD-розширення процесора

Subversion, SVN – вільна централізована система управління версіями

Visual SourceSafe – файл-серверна система управління версіями

UX, user experience design – дизайн користувальницького досвіду

ВСТУП

Озеро Катлабух відділено греблею від Кілійського рукава, на якому встановлено шлюз Желявський для регулювання надходження Дунайської води до озера, але сучасні економічні проблеми не дають змоги забезпечити достатній водообмін. Крім того, здійснюється примусовий водообмін Катлабуха із близько розташованим озером Сафьян (через канал Громадський). Враховуючи замулення підвідних каналів в баровій частині, водообмін досить слабкий. В останній час ситуація на озері Катлабух погіршилась ще більше, оскільки рівень води озера досягнув критично низьких відміток, підвищилася мінералізація води, що унеможливлює її використання навіть для поливу земель.

В таких умовах актуальним є дослідження гідроекологічного режиму озера Катлабух з метою контролю мінералізації і якості води в ньому. При цьому необхідним є щорічний розрахунок водних та сольових балансів з місячним розрахунковим інтервалом часу, причому як в кожному поточному році, так і за багаторічний період. Крім того важливим є здійснення часового моделювання водно-сольових режимів за різних умов експлуатації водогосподарського використання ресурсів озера. Такі дослідження потребують великої кількості вихідних гідрометеорологічних даних, детальних розрахунків приходної та витратної частин балансів, оцінки точності одержаних результатів, модельних розрахунків, представлення щорічних і багаторічних розрахункових величин у вигляді графіків, діаграм, залежностей.

Мета магістерської роботи – розробка програмного комп'ютерного комплексу для моделювання складових водних і сольових балансів озера Катлабух за різних режимів його водогосподарського використання [1][1)].

[1)] [1] Романова Є.О., Шакірзанова Ж.Р., Медведєва Ю.С. Водний та сольовий баланси озера Катлабух за різних умов експлуатації водойми.

Для досягнення поставленої мети в роботі необхідно вирішити наступні завдання:

- ознайомитися з вимогами до ПП;
- виконати аналіз моделі водно-сольового балансу озера Катлабух, а також методи та алгоритми моделювання її складових;
- обрати стек технологій, використання яких допоможе досягти поставленої мети;
- спроектувати модуль розрахування водно-сольових балансів, сховища даних і архітектури ПП;
- розробити інтерфейс користувача, що відповідає висунутим вимогам;
- провести тестування компонентів ПП.

1 АНАЛІТИЧНА ЧАСТИНА

1.1 Вимоги до програмного продукту

Розроблений програмний продукт – це програмний продукт, який реалізовано на платформі .NET для використання на персональних комп'ютерах, що керуються операційною системою MS Windows, і має назву «CatlabuhApp», що повинен буде замінити стару систему проведення і збереження розрахунку, яка базувалась на використанні програмного додатку MS Office Excel.

Програмний продукт буде застосовуватися в якості допоміжного інструменту у операціях розрахування і аналізу водно-сольового балансу озера Катлабух, і відповідає наступним основним вимогам, вказаним у ТЗ:

- ПП повинен автоматизувати процес розрахування водно-сольового балансу;
- ПП повинен зберігати результати розрахування водно-сольового балансу;
- ПП повинен мати зручний інтерфейс користувача, для перегляду результатів розрахування водно-сольового балансу;

В кінцевому вигляді програмний продукт матиме вигляд архіву з розширенням .zip(.rar), що включатиме файли програми і керівництво користувача. Цей архів користувач завантажить та розпакує на своєму персональному комп’ютері.

1.2 Загальні відомості

Прісноводне озеро Катлабух, як і інші озера посушливого регіону Придунав’я в межах України є одним з поверхневих джерел постачання води на господарські потреби і зрошення сільськогосподарських культур регіону. В середині минулого століття озеро було зв’язано з р. Дунай природними протоками,

які з'єднували водойму з річкою. На той час гідрологічний режим рівнів води в озері визначався динамікою рівнів води в р.Дунай. Це сприяло доброму водообміну та підтримки задовільної якості води у водоймі.

В сучасний період озеро Катлабух відділено від річки Дунай смугою заболоченої суші, з'єднане з нею природними протоками чи штучними каналами, на яких побудовані гідротехнічні споруди для регулювання водообміну і пропуску риби. Свого часу, завдяки заборам води на зрошування і доброго водообміну в озері стан якості води в ньому відносно сольового складу був задовільний. При зменшенні у 90-ті року минулого століття об'ємів зрошення і заборів води з озера, водообмін води погіршився, що призвело до підвищення мінералізації води озера, яка перевищує допустимі норми для питної та зрошувальної води.

Тому актуальною проблемою є дослідження водного і сольового режимів озера Катлабух за різних умов експлуатації водойми. При цьому гідрологічний режим озера маловивчений, а деякі компоненти водного і сольового балансу не мають систематичних спостережень, що потребує розробки методів їх оцінки та визначення.

Метод водного балансу є одним з основоположних наукових методів, при дослідженнях гідрологічного режиму водосховищ, озер і ставків, відомим в науковій літературі. Рівняння водного балансу водойм дають можливість докладно вивчити та зіставити між собою складові приходної та витратної частин водних балансів, що є необхідним при плануванні використання вод, визначенні заходів щодо задоволення потреб у воді господарства і населення. На основі водних балансів розробляються й сольові баланси, які дозволяють оцінювати мінералізацію та якість води водойм для різних водогосподарських потреб[1][1)].

[1)] [1] Романова Є.О., Шакірзанова Ж.Р., Медведєва Ю.С. Водний та сольовий баланси озера Катлабух за різних умов експлуатації водойми.

Для виконання за цим методом розрахунку водно-сольового балансу, необхідна обробка великої кількості вхідних даних. При цьому операції над даними і їх розрахування власноруч займає велику кількість часу.

Раніше для рішення цієї задачі використовувався табличний процесор для роботи з електронними таблицями, створений корпорацією Microsoft – Microsoft Office Excel. Не зважаючи на відносну простоту роботи з електронними таблицями, їх використання для розрахунків водно-сольових балансів має ряд недоліків:

- розрахунок, який складається з двох залежних одна від одної частин, розподілений на два файли .xls (.xlsx) і втрата одного з них спричинить за собою неможливість проведення повного розрахунку і його подальшого аналізу. Також ця залежність потребує від користувача переносити частину даних, необхідних для розрахунку, від одного файлу до іншого;
- вихідні дані для розрахунків знаходяться в одному файлі, тобто існує імовірність випадкового редагування іншої таблиці;
- табличний процесор MS Excel не дозволяє перевіряти дані, що вводяться, і структурувати таблиці, необхідним для поставленої задачі чином.

Тому підходячи до проектування і розробки ПП треба приділити увагу на позбавлення цих недоліків[1][1)].

1.3 Модель водно-сольового балансу водойми

Розрахунок водного і сольового балансів озера Катлабух складається з двох частин: розрахунок водного балансу і розрахунок сольового балансу. В свою чергу кожен баланс розрахунку також складається з двох частин: прибут-

[1)] [1] Романова Є.О., Шакірзанова Ж.Р., Медведєва Ю.С. Водний та сольовий баланси озера Катлабух за різних умов експлуатації водойми.

кової частини і витратної частини. Модель розрахунку можна переглянути на рис. 1.

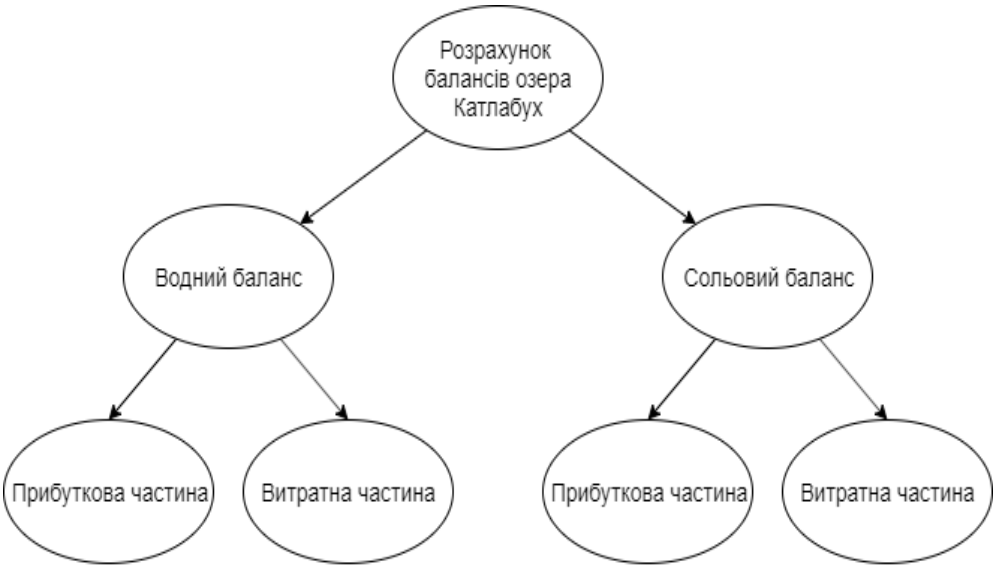


Рисунок 1 – Модель розрахунку водного і сольового балансів оз. Катлабух

Розглянемо складові кожної частини розрахунку водно-сольового балансу. На таблицях 1 і 2 перелічено всі компоненти водного балансу, які приймають участь у розрахунку. Частина цих складових є вхідними даними для проведення операції розрахування і є початковою точкою всього розрахунку, а саме:

- рівні води на початку і кінця місяця;
- опади, що були виміряні на метеостанціях Болград і Ізмаїл;
- річний об’єм припливу ґрунтових вод;
- дефіцит вологості, або якщо даних дефіциту немає можна використовувати показники випаровування;
- збори води на зрошування;

Осібно стоїть диспетчерський графік роботи шлюзів, дані якого можуть бути розраховані при вказані місяців роботи або використовуватися як вхідні дані.

Таблиця 1 – Складові ВБПЧ

Позначення складової	Опис	Одиниці вимірювання
H1	Рівень води на початку місяця	мБС
H2	Рівень води наприкінці місяця	мБС
H ср.	Середній рівень води за місяць	мБС
W1	Об'єм води в озері на початку місяця	млн $м^3$
W2	Об'єм води в озері наприкінці місяця	млн $м^3$
ΔW	Зміни об'ємів води в озері за місяць	млн $м^3$
F(H ср.)	Площа водного дзеркала	$км^2$
P Ізм.	Опади виміряні на м/ст Ізмаїл	мм
P Болг.	Опади виміряні на м/ст Болград	мм
Vp	Об'єм атмосферних опадів	млн $м^3$
Vr	Об'єм річкового стоку	млн $м^3$
Vb	Об'єм бічного припливу	млн $м^3$
Vg	Об'єм припливу ґрунтових вод	млн $м^3$
Vdr	Об'єм надходження дренажних вод	млн $м^3$
∑P	Сума прихідної частини	млн $м^3$
VD+	Приплив води з р. Дунай	млн $м^3$
Voz+	Підрімка рівнів води озер Лунг-Сафьян	млн $м^3$
ΔVні	Нев'язкі рівняння водного балансу	млн $м^3$

Таблиця 2 – Складові ВБВЧ

Позначення складової	Опис	Одиниці вимірювання
1	2	3
d	Дефіцит влогості	гПа
lg(d)	Логарифм дефіциту вологості	-
lg(E)	Логарифм випаровування	-
E	Випаровування	мм
Etr	Транспірація	мм

Продовження таблиці 2

1	2	3
VE	Об'єм випаровування	млн $м^3$
Vtr	Об'єм транспірації	млн $м^3$
Vf	Об'єм фільтраціїї	млн $м^3$
Vz	Збір води на зрошування	млн $м^3$
∑R	Сума витратної частини	млн $м^3$
VD-	Скиди води з озера в р. Дунай	млн $м^3$
Voz-	Підримка рівнів води озер Лунг-Сафьян	млн $м^3$

Перейдемо до огляду складових сольової частини розрахунку, які можна переглянути на таблицях 3 і 4. Як було раніше сказано, розрахунок сольової частини залежить від даних водного балансу. В представленні таблиць прибуткової і витратної частини, є опис складових водного балансу, які виказують залежність балансів, а саме:

- об’єми води в озері на початку і наприкінці місяця;
- об’єми атмосферних опадів;
- об’єми річкового стоку;
- об’єми бічного припливу;
- об’єми припливу ґрунтових вод;
- об’єми надходжень дренажних вод;
- об’єми випаровування;
- об’єми транспірації;
- об’єми фільтрації;
- об’єми зборів води на зрошування;
- дані об’ємів води з ДГРШ;

Таблиця 3 – Складові СБПЧ

Позначення складової	Опис	Одиниці вимірювання
W1, W2, Vp, Vr, Vb, Vg, Vdr, VD+, Voz+	Дивись таблицю 1.	млн $м^3$
S1	Середня по озеру мінералізація на початку місяця	$кг/м^3$
S2	Середня по озеру мінералізація наприкінці місяця	$кг/м^3$
Sp	Мінералізація атмосферних опадів	$кг/м^3$
Sr	Мінералізація поверхневого стоку	$кг/м^3$
Sb	Мінералізація бічного стоку	$кг/м^3$
Sg	Мінералізація ґрунтового стоку	$кг/м^3$
Sdr	Мінералізація дренажних вод	$кг/м^3$
SD+	Мінералізація Дунайської води	$кг/м^3$
Soz+	Мінералізація води на підтримку рівнів води в системі озер Лунг-Сафьян	$кг/м^3$
C1	Концентрація солей по озеру на початку місяця	10^6 т.
C2	Концентрація солей по озеру на початку місяця	10^6 т.
Cp	Надходження солей з опадами	10^6 т.
Cr	Надходження солей з поверхневим стоком	10^6 т.
Cb	Надходження солей з бічним припливом	10^6 т.
Cg	Надходження солей з ґрунтовими водами	10^6 т.
Cdr	Надходження солей з дренажними водами	10^6 т.
CD+	Надходження солей з Дунайською водою	10^6 т.
Coz+	Надходження солей з системи озер Лунг-Сафьян	10^6 т.
∑pCi+	Сума прихідної частини	10^6 т.

Окремо слід виділити середню по озеру мінералізацію на початку січня місяця. Ця складова, також входить до переліку вхідних даних і береться з показника середньої по озеру мінералізації наприкінці грудня попереднього року.

Таблиця 4 – Складові СБВЧ

Позначення складової	Опис	Одиниці вимірювання
VE, Vtr, Vf, Vz, VD-, Voz-	Дивись таблицю 2.	млн $м^3$
Sf	Мінералізація фільтрації	$кг/м^3$
Sz	Мінералізація збору води на зрошування	$кг/м^3$
SD-	Мінералізація скидів води у р. Дунай	$кг/м^3$
Soz-	Мінералізація води на підтримку рівнів води в системі озер Лунг-Сафьян	$кг/м^3$
Cf	Втрата солей з фільтрацією	10^6 т.
Cz	Втрата солей на зрошуванні	10^6 т.
CD-	Скиди води разом із солями у р. Дунай	10^6 т.
Coz-	Вивід солей з водою на підтримку рівнів води у системі озер Лунг-Сафьян	10^6 т.
EpCi-	Сума витратної частини	10^6 т.

Розглянувши складові частин водного і сольового балансів, а також виділивши їх вхідні дані, тепер можна оглянути правила розрахування всіх інших складових, які мають назву вихідні дані. Правила розрахування вихідних даних водного балансу прибуткової частині відображені в таблиці 5.

Таблиця 5 – Правила розрахування ВБПЧ

Позначення складової	Формула розрахунку і-місяця	Формула розрахунку суми	Формула розрахунку відсотків
1	2	3	4
H ср.	$\frac{H1 + H2}{2}$	-	-
W1	$H1 * 67.098 + 17.278$	-	-

Продовження таблиці 5

1	2	3	4
W2	$H2 * 67.098 + 17.278$	-	-
ΔW	$W2 - W1$	-	-
Vp	$\frac{F(H\ ср.) * P\ Ізм.}{1000}$	$\sum_{i=1}^{12} Vp$	$\frac{\sum Vp}{\sum P + \sum VD + \sum Voz} * 100$
Vr	$\frac{\sum Vr}{100} * k1$	$\sum_{i=1}^{12} Vr$	$\frac{\sum Vr}{\sum P + \sum VD + \sum Voz} * 100$
Vb	Vr * 0.23	$\sum_{i=1}^{12} Vb$	$\frac{\sum Vb}{\sum P + \sum VD + \sum Voz} * 100$
Vg	$\frac{\sum Vg}{100} * k2$	$\sum_{i=1}^{12} Vg$	$\frac{\sum Vg}{\sum P + \sum VD + \sum Voz} * 100$
Vdr	$Vz * 0.2$	$\sum_{i=1}^{12} Vdr$	$\frac{\sum Vdr}{\sum P + \sum VD + \sum Voz} * 100$
∑P	Vp + Vr + Vb + Vg + Vdr	∑Vp + ∑Vr + ∑Vb + ∑Vg + ∑Vdr	%Vp + %Vr + %Vb + %Vg + %Vdr
VD+	$-(\sum P - \sum R - \Delta W)$	$\sum_{i=1}^{12} VD$	$\frac{\sum VD}{\sum P + \sum VD + \sum Voz} * 100$
Voz+	$-(\sum P - \sum R - \Delta W)$	$\sum_{i=1}^{12} Voz$	$\frac{\sum Voz}{\sum P + \sum VD + \sum Voz} * 100$
ΔVні	$-(\sum P - \sum R - \Delta W)$	$\sum_{i=1}^{12} \Delta Vні$	$\frac{\sum \Delta Vні}{\sum P + \sum VD + \sum Voz} * 100$

У розрахуванні таких складових як Vr і Vg присутні коефіцієнти k1 і k2. Це коефіцієнти внутрішньорічного розподілу поверхневого стоку і припливу ґрунтових вод відповідно. В залежності від водності року, який буває многоводним, середньоводним або маловодним, беруться відповідні значення коефіцієнтів.

нтів для кожного місяця. Значення коефіцієнтів можна переглянути в таблицях 6 та 7.

Таблиця 6 – Внутрішньорічний розподіл поверхневого стоку

Місяці / Водність	I	II	III	IV	V	VI	VII	VIII	IX	X	XI	XII
Багатоводний	1.8	24	15.7	4.5	3.1	6.8	5	15.7	8.1	7.3	4.4	3.5
Середньоводний	3.3	8.1	13.2	6.3	7.8	6.6	10.7	14.8	6.6	6.9	7.6	8.1
Маловодний	9.2	14.6	7.3	5.6	11.8	5.6	12.5	3.8	9.3	5.4	7	7.9

Таблиця 7 – Внутрішньорічний розподіл припливу ґрунтових вод

Місяці / Водність	I	II	III	IV	V	VI	VII	VIII	IX	X	XI	XII
Багатоводний	8	6.4	10.3	10.7	11.4	11.3	8	6.4	6.3	7.2	8.4	9.2
Середньоводний	7.4	10.3	12.6	12.3	12.5	6.5	5.2	5.8	6.5	7.7	7.4	5.8
Маловодний	3.9	4.7	5.11	5.2	5.4	6	8	9.3	11	13.4	13.5	14.5

Окремо розраховуються площа водного дзеркала і річне значення річкового стоку. Значення площі водного дзеркала можна отримати в залежності від середнього рівня води(див. рис. 2).

В Одеському державному екологічному університеті під керівництвом проф. Є. Д. Гопченка та проф. Н. С. Лободи була розроблена математична модель річного стоку, яка базується на використанні метеорологічних даних і відноситься до моделей типу «клімат-стік». Теоретичним базисом моделі «клімат-стік» є запропонована в модифікація рівняння водно-теплового балансу В.С. Мезенцева. За одержаним по методиці «клімат-стік» модулі природного річного стоку $\bar{q}$= 0.27 л/(с·км^2). При загальній площі водозборів всіх річок, що впадають в озеро Катлабух (F = 1060 км^2) середня багаторічна річна витрата води $\overline{Q}$ ста-

новить 0.29 м^3/с, що в об’ємах дорівнює 9.15 млн. м^3.

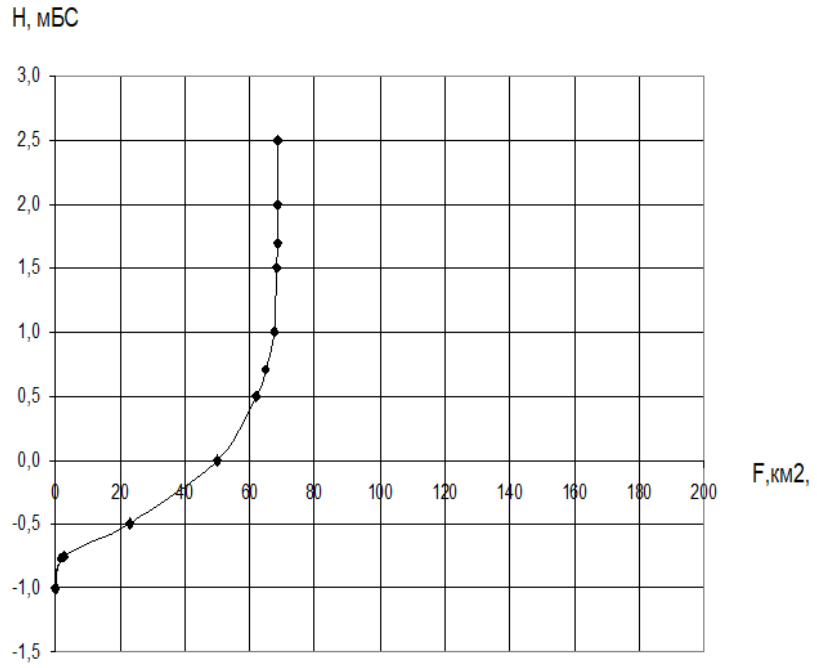


Рисунок 2 – Залежність площини водної поверхні від рівнів води в озері Катлабух

Річний стік з площі водозборів малих річок визначався з урахуванням забезпеченості водності року за формулою

$$Q_r = k_p \cdot \overline{Q},$$

де k_p – коефіцієнт, що враховує забезпеченість водності року ймовірністю p%, $\overline{Q}$ – середня багаторічна річна витрата води, м^3/с.

В цій роботі вважалось, що забезпеченість річного стоку дорівнює забезпеченості річних опадів, яка визначена за кривою забезпеченості річних опадів по м/ст Болград. При цьому, встановлений зв’язок середнього шару стоку води за рік поста-аналога р. Когильник – м. Котовськ з сумою річних атмосферних опадів на м/ст Болград, як основного чинника формування річкового стоку (при значущому коефіцієнті кореляції зв'язку – 0.42).

Об'єм надходження річного стоку невеликих річок V_r (млн.м^3) розраховувався за виразом

$$V_r = Q_r \cdot 86400 \cdot 365 / 10^6,$$

де Q_r - річний стік з площ водозборів річок басейну оз. Катлабух, м^3/с; 86400 – кількість секунд в одній добі; 365 – кількість діб у році[1].

Тепер розглянемо правила розрахування вихідних даних витратної частини водного балансу(див. таб. 8).

Таблиця 8 – Правила розрахування ВБВЧ

Позначення складової	Формула розрахунку і-місяця	Формула розрахунку суми	Формула розрахунку відсотків
1	2	3	4
lg(d)	lg(d)	-	-
lg(E)	$k1 * lg(d) + k2$	-	-
E	$10^{lg(E)}$	$\sum_{i=1}^{12} E$	-
Etr	$\sum Etr / 100 * k3$	$\sum E * 1.14 - \sum E$	-
VE	$\frac{F(H\ cp.) * E}{1000}$	$\sum_{i=1}^{12} VE$	$\frac{\sum VE}{\sum R + \sum VD + \sum Voz} * 100$
Vtr	$\frac{Etr * F(H\ cp.) * 0.3}{1000}$	$\sum_{i=1}^{12} Vtr$	$\frac{\sum Vtr}{\sum R + \sum VD + \sum Voz} * 100$
Vf	$\frac{Hcp. * 0.55}{\overline{\sum Hcp.}}$	$\sum_{i=1}^{12} Vf$	$\frac{\sum Vf}{\sum R + \sum VD + \sum Voz} * 100$
Vz	-	$\sum_{i=1}^{12} Vz$	$\frac{\sum Vz}{\sum R + \sum VD + \sum Voz} * 100$
$\sum R$	$VE + Vtr + Vf + Vz$	$\sum VE + \sum Vtr + \sum Vf + \sum Vz$	$\frac{\sum R}{\sum R + \sum VD + \sum Voz} * 100$

Продовження таблиці 8

1	2	3	4
VD-	$-(\sum P - \sum R - \Delta W)$	$\sum_{i=1}^{12} VD$	$\frac{\sum VD}{\sum R + \sum VD + \sum Voz} * 100$
Voz-	$-(\sum P - \sum R - \Delta W)$	$\sum_{i=1}^{12} Voz$	$\frac{\sum Voz}{\sum R + \sum VD + \sum Voz} * 100$

Звернемо увагу на розрахунок таких складових, як: lg(E), E, Etr. Ці складові розрахуваються лише не на кожен місяць. Складові lg(E) і E розраховуються для місяців з квітня по вересень включно, а складова Etr – з травня по жовтень включно. Якщо розрахунок витратної частини починати з випаровування(E), то значення складових d, lg(d) і lg(E) не розраховуються.

У рівняннях складових lg(E) і Etr присутні коефіцієнти k1, k2, і k3. Ці значення цих коефіцієнтів можна переглянути в таблиці 9.

Таблиця 9 – Коефіцієнти розрахування складових lg(E), Etr

Місяці / Коефіцієнти	IV	V	VI	VII	VIII	IX	X
k1	0.445	0.56	0.896	0.47	0.488	0.364	-
k2	1.498	1.53	1.216	1.67	1.677	1.653	-
k3	-	7	23	27	25	15	3

Окремо слід виділити розрахунок складових, що представляють значення ДГРШ. Всі значення розраховуються за одним правилом і записуються у складову ΔVні, проте ці значення переносяться у складові VD± і Voz± в залежності від того, коли працювали шлюзи і у якому режимі, тобто наповнювали озеро чи навпаки скидали воду. Слід зазначити, що складові VD± відображають роботу Желявського шлюзу, а складові Voz± − роботу Громадського шлюзу.

Тепер перейдемо до розгляду правил розрахування складових сольового балансу розрахунку балансів озера Катлабух. На таблицях 10 і 11 відображені правила розрахування прибуткової частини і витратної частини відповідно.

Таблиця 10 – Правила розрахування СБПЧ

Позначення складової	Формула розрахунку і-місяця	Формула розрахунку суми	Формула розрахунку відсотків
1	2	3	4
W1, W2, Vp, Vr, Vb, Vg, Vdr, VD+, Voz+	див. табл. 5	див. табл. 5	див. табл. 5
S1	$S2_{i-1}$	-	-
S2	$\frac{C2}{W2}$	-	-
Sp	0.22	-	-
Sr	$5.12 * \exp(Vr * -0.021)$	-	-
Sb	$Sr * 0.57$	-	-
Sg	2.6	-	-
Sdr	2	-	-
SD+	0.39	-	-
Soz+	0	-	-
C1	$C2_{i-1}$	-	-
C2	C1 + (∑pCi+) - (∑pCi-)	-	-
Cp	$Vp * Sp$	$\sum_{i=1}^{12} Cp$	$\frac{\sum Cp}{\sum(\sum pCi+)} * 100$
Cr	$Vr * Sr$	$\sum_{i=1}^{12} Cr$	$\frac{\sum Cr}{\sum(\sum pCi+)} * 100$
Cb	$Vb * Sb$	$\sum_{i=1}^{12} Cb$	$\frac{\sum Cb}{\sum(\sum pCi+)} * 100$

Продовження таблиці 10

1	2	3	4
Cg	$Vg * Sg$	$\sum_{i=1}^{12} Cg$	$\frac{\sum Cg}{\sum(\sum pCi+)} * 100$
Cdr	$Vdr * Sdr$	$\sum_{i=1}^{12} Cdr$	$\frac{\sum Cdr}{\sum(\sum pCi+)} * 100$
CD+	$VD * SD$	$\sum_{i=1}^{12} CD$	$\frac{\sum CD}{\sum(\sum pCi+)} * 100$
Coz+	$Voz * Soz$	$\sum_{i=1}^{12} Coz$	$\frac{\sum Coz}{\sum(\sum pCi+)} * 100$
∑pCi+	Cp + Cr + Cb + Cg + Cdr + CD + Coz	∑Cp + ∑Cr + ∑Cb + ∑Cg + ∑Cdr + ∑CD + ∑Coz	$\frac{\sum(\sum pCi+)}{\sum(\sum pCi+)} * 100$

Таблиця 11 – Правила розрахування СБВЧ

Позначення складової	Формула розрахунку і-місяця	Формула розрахунку суми	Формула розрахунку відсотків
1	2	3	4
VE, Vtr, Vf, Vz, VD-, Voz-	див. табл. 8	див. табл. 8	див. табл. 8
Sf	S1	-	-
Sz	S1	-	-
SD-	$S1 * 0.82$	-	-
Soz-	S1	-	-
Cf	$Vf * Sf$	$\sum_{i=1}^{12} Cf$	$\frac{\sum Cf}{\sum(\sum pCi-)} * 100$
Cz	$Vz * Sz$	$\sum_{i=1}^{12} Cz$	$\frac{\sum Cz}{\sum(\sum pCi-)} * 100$

Продовження таблиці 11

1	2	3	4
CD-	$VD * SD$	$\sum_{i=1}^{12} CD$	$\frac{\sum CD}{\sum(\sum pCi-)} * 100$
Coz-	$Voz * Soz$	$\sum_{i=1}^{12} Coz$	$\frac{\sum Coz}{\sum(\sum pCi-)} * 100$
EpCi-	Cf + Cz + CD + Coz	$\sum Cf + \sum Cz + \sum CD + \sum Coz$	$\frac{\sum(\sum pCi-)}{\sum(\sum pCi-)} * 100$

1.4 Аналіз програмних засобів розробки

Майбутній ПП буде використовуватися переважно на персональних комп'ютерах під керуванням ОС Windows. Тому слід розглянути платформи розробки програмних додатків для ОС Windows від корпорації Microsoft. Всього їх існує чотири основних платформи:

- Універсальна платформа Windows;
- WPF (.NET);
- Windows Forms (.NET);
- Win32.

Всі ці платформи додатків дозволяють створювати такі програми, як Word, Excel і Photoshop, які працюють в класичній ОС Windows і використовують всі переваги конкретних функцій середовища. Однак деякі з цих платформ мають деякі характеристиками і краще підходять для деяких типів додатків[2][1)].

UWP, WPF і Windows Forms. Ці платформи надають керовані середовища виконання (виконавча Windows для UWP і .NET для Windows Forms і WPF) з безліччю переваг насамперед для підвищення ефективності роботи розробника, застосування складного і настроюється призначеного для користувача інтер-

[1)] [2] Выбор платформы для приложения. URL: https://docs.microsoft.com/ru-ru/windows/apps/desktop/choose-your-platform. (дата звернення 03.09.2020).

фейсу і безпеки додатків.

Ці платформи особливо добре підходять для бізнес-додатків, оскільки підтримують візуальні конструктори і розмітку призначеного для користувача інтерфейсу для швидкого створення призначеного для користувача інтерфейсу.

Win32 API (також званий Windows API) – це вихідна платформа для власних Windows-додатків на мові C/C ++, яким потрібен прямий доступ до Windows і обладнання. Він надає першокласний інтерфейс розробки, що не залежить від керованого середовища виконання, такий як .NET і WinRT. Завдяки цьому API Win32 стає оптимальною платформою для додатків, яким потрібна найвищий рівень продуктивності і прямий доступ до системного обладнання[2].

UWP – це передова платформа для додатків та ігор, призначених для Windows 10. Це платформа з широкими можливостями налаштування, яка використовує розмітку XAML для відділення призначеного для користувача інтерфейсу (подання) від коду (бізнес-логіки). UWP підходить для класичних програм, для яких потрібно розширений призначений для користувача інтерфейс, настройка стилів і сценарії з великим об'ємом графіки. UWP також має вбудовану підтримку Системи дизайну Fluent для роботи з UX за замовчуванням і забезпечує доступ до API Windows Runtime (WinRT). Використовуючи Fluent, UWP автоматично підтримує стандартні методи введення, такі як рукописний ввід, сенсорний ввід, ігровий планшет, клавіатура і миша.

UWP можна використовувати не тільки для створення класичних додатків для комп'ютерів під управлінням Windows – UWP також є єдиною платформою, підтримуваної додатками Xbox, HoloLens і Surface Hub. UWP - це новітня, ведуча платформа додатків[2][1)].

WPF – це загальновизнана платформа для керованих додатків для Windows з доступом до всіх компонентів платформи .NET Core або повної пла-

---

[1)] [2] Выбор платформы для приложения. URL: https://docs.microsoft.com/ru-ru/windows/apps/desktop/choose-your-platform. (дата звернення 03.09.2020).

тформи .NET Framework. Вона також використовує розмітку XAML для відділення призначеного для користувача інтерфейсу від коду. Ця платформа створена для класичних програм, для яких потрібно розширений призначений для користувача інтерфейс, настройка стилів і сценарії з великим об'ємом графіки. Навички розробки WPF схожі на навички розробки UWP, тому міграція з WPF в додатки UWP простіше, ніж міграція з Windows Forms.

Windows Forms – це вихідна платформа для керованих додатків для Windows на основі спрощеної моделі призначеного для користувача інтерфейсу з доступом до всіх компонентів платформи .NET Core або повної платформи .NET Framework. Це дозволяє розробникам швидко приступити до створення додатків, навіть новачкам на платформі. Це платформа швидкої розробки додатків на основі форм з великою вбудованою колекцією візуальних і не візуальних елементів управління, що підтримують перетягування. Windows Forms не використовує XAML, тому при прийнятті рішення про розширення програми в UWP буде виконана повна повторна запис призначеного для користувача інтерфейсу.

Використання API Win32 з C++ дозволяє досягти найвищого рівня продуктивності та ефективності завдяки підвищенню контролю над цільовою платформою за допомогою некерованого коду, що можливо в керованому середовищі виконання, такий як WinRT і .NET. Однак такий рівень контролю над виконанням програми вимагає більшої обачності і уваги для правильного виконання, і дозволяє отримувати вигоду з продуктивності розробки для продуктивності середовища[2][1)].

Ось кілька основних особливостей інтерфейсу API Win32 і C++ пропозицій, що дозволяють створювати високопродуктивні додатки:

- оптимізація на рівні обладнання, що включає ретельний контроль за виділенням ресурсів, час існування об'єктів, макет даних, вирівнювання,

[1)] [2] Выбор платформы для приложения. URL: https://docs.microsoft.com/ru-ru/windows/apps/desktop/choose-your-platform. (дата звернення 03.09.2020).

байтову упаковку і багато іншого;

- доступ до наборів інструкцій, орієнтованих на продуктивність, наприклад SSE і AVX, за допомогою вбудованих функцій;
- ефективне і строго типізовані універсальне програмування за допомогою шаблонів;
- ефективні та надійні контейнери і алгоритми;
- DirectX, зокрема Direct3D і DirectCompute (зверніть увагу, що UWP також пропонує між програмний взаємодія з DirectX)[2][1)].

Роботу з усіма переліченими платформами підтримує інтегроване середовище програмування Visual Studio, також від корпорації Microsoft.

Microsoft Visual Studio – лінійка продуктів компанії Microsoft, що включають інтегроване середовище розробки програмного забезпечення і ряд інших інструментальних засобів. Дані продукти дозволяють розробляти як консольні додатки, так і додатки з графічним інтерфейсом, в тому числі з підтримкою технології Windows Forms, а також веб-сайти, веб-додатки, веб-служби як в рідному, так і в керованому кодах для всіх платформ, підтримуваних Windows, Windows Mobile, Windows CE, .NET Framework, Xbox, Windows Phone .NET Compact Framework і Silverlight[3][2)].

Visual Studio включає в себе редактор вихідного коду з підтримкою технології IntelliSense і можливістю найпростішого рефакторінга коду. Вбудований відладчик може працювати як відладчик рівня вихідного коду, так і відладчик машинного рівня. Решта вбудовуються інструменти включають в себе редактор форм для спрощення створення графічного інтерфейсу додатку, веб-редактор, дизайнер класів і дизайнер схеми бази даних. Visual Studio дозволяє створювати і підключати сторонні додатки (плагіни) для розширення функціональності

[1)] [2] Выбор платформы для приложения. URL: https://docs.microsoft.com/ru-ru/windows/apps/desktop/choose-your-platform. (дата звернення 03.09.2020).

[2)] [3] Microsoft Visual Studio. URL: https://ru.wikipedia.org/wiki/Microsoft_Visual_Studio. (дата звернення 03.09.2020).

практично на кожному рівні, включаючи додавання підтримки систем контролю версій вихідного коду (як, наприклад, Subversion і Visual SourceSafe), додавання нових наборів інструментів (наприклад, для редагування і візуального проектування коду на предметно-орієнтованих мовах програмування) або інструментів для інших аспектів процесу розробки програмного забезпечення (наприклад, клієнт Team Explorer для роботи з Team Foundation Server)[3][1)].

Однією з перелічених вимог до ПП є – зберігання результатів операцій розрахування і існує багато шляхів рішення цієї проблеми. Результати розрахувань можна зберігати як в базі даних так і в окремих файлах.

SQLite – це вбудований двигун бази даних SQL. На відміну від більшості інших баз даних SQL, у SQLite немає окремого серверного процесу. SQLite читає і записує безпосередньо у звичайні файли диска. Повна база даних SQL з кількома таблицями, індексами, тригерами та переглядами міститься в одному файлі диска. Формат файлу бази даних є кросплатформенним, тобто можна вільно копіювати базу даних між 32-бітовою та 64-бітовою системами або між архітектурами big-endian та little-endian. Ці функції роблять SQLite популярним вибором як формат файлу додатків[4][2)].

MySQL – вільна система керування реляційними базами даних. MySQL був розроблений компанією «ТсХ» для підвищення швидкодії обробки великих баз даних. Ця система керування базами даних (СКБД) з відкритим кодом була створена як альтернатива комерційним системам. MySQL з самого початку була дуже схожою на mSQL, проте з часом вона все розширювалася і зараз MySQL – одна з найпоширеніших систем керування базами даних. Вона використовуєть-

[1)] [3] Microsoft Visual Studio. URL: https://ru.wikipedia.org/wiki/Microsoft_Visual_Studio. (дата звернення 03.09.2020).

[2)] [4] About SQLite. URL: https://www.sqlite.org/about.html. (дата звернення 03.09.2020).

ся, в першу чергу, для створення динамічних веб-сторінок, оскільки має чудову підтримку з боку різноманітних мов програмування[5][1)].

XML (Extensible Markup Language) – запропонований консорціумом World Wide Web Consortium (W3C) стандарт побудови мов розмітки ієрархічно структурованих даних для обміну між різними застосунками, зокрема, через Інтернет. Є спрощеною підмножиною мови розмітки SGML. XML-документ складається із текстових знаків, і придатний до читання людиною. Стандарт XML визначає набір базових лексичних та синтаксичних правил для побудови мови описання інформації шляхом застосування простих тегів. Цей формат достатньо гнучкий для того, аби бути придатним для застосування в різних галузях[6][2)].

JSON (JavaScript Object Notation) – простий формат обміну даними, зручний для читання і написання як людиною, так і комп'ютером. Він заснований на підмножині мови програмування JavaScript, визначеного в стандарті ECMA-262 3rd Edition - December 1999. JSON – текстовий формат, повністю незалежний від мови реалізації, але він використовує угоди, знайомі програмістам C-подібних мов, таких як C, C++, C#, Java, JavaScript, Perl, Python і багатьох інших. Ці властивості роблять JSON ідеальним мовою обміну даними[7][3)].

JSON заснований на двох структурах даних:

- колекція пар ключ/значення. У різних мовах, ця концепція реалізована як об'єкт, запис, структура, словник, хеш, іменований список або асоціативний масив;
- упорядкований список значень. У більшості мов це реалізовано як масив, вектор, список або послідовність.

---

1) [5] MySQL. URL: https://uk.wikipedia.org/wiki/MySQL. (дата звернення 05.09.2020).
2) [6] XML. URL: https://uk.wikipedia.org/wiki/XML. (дата звернення 05.09.2020).
3) [7] Введение в JSON. URL: https://www.json.org/json-ru.html. (дата звернення 05.09.2020).

Це універсальні структури даних. Майже всі сучасні мови програмування підтримують їх в будь-якій формі. Логічно припустити, що формат даних, незалежний від мови програмування, повинен бути заснований на цих структурах[7][1)].

Під час розробки ПП обв'язко знадобиться зберегти результати роботи тих чи інших етапів розробки ПЗ. Для цих цілей треба звернути увагу на системи контролю версій. . Існують різні системи для контролю версій. Ви, можливо, про них чули: SVN, Mercurial, Perforce, CVS, Bitkeeper і інші. Самою популярною серед розробників є Git.

Git – це набір консольних утиліт, які відстежують і фіксують зміни в файлах (найчастіше мова йде про вихідний код програм, але можна використовувати його для будь-яких файлів на наш смак). Проект був створений Лінус Торвальдс для управління розробкою ядра Linux, перша версія випущена 7 квітня 2005 року. За допомогою Git є можливість відкотитися на більш стару версію вашого проекту, порівнювати, аналізувати, зливати зміни і багато іншого. Цей процес називається контролем версій.

Git є розподіленим, тобто не залежить від одного центрального сервера, на якому зберігаються файли. Замість цього він працює повністю локально, зберігаючи дані в папках на жорсткому диску, які називаються репозиторієм. Тим не менш, ви можете зберігати копію сховища онлайн, це сильно полегшує роботу над одним проектом для кількох людей. Для цього використовуються сайти на кшталт github і bitbucket[8][2)].

[1)] [7] Введение в JSON. URL: https://www.json.org/json-ru.html. (дата звернення 05.09.2020).

[2)] [8] Git за полчаса: руководство для начинающих. URL: https://proglib.io/p/git-for-half-an-hour. (дата звернення 05.09.2020).

1.5 Обґрунтування вибору програмних засобів розробки

За результатами проведення аналізу було прийнято рішення. Розробляти ПП для проведення операцій розрахування водного і сольового балансів озера Катлабух, використовуючи платформу Windows Forms, мову програмування C# та СКБД SQLite. Цей вибір обумовлений наступними чинниками:

- велика швидкість розробки на обраній платформі;
- зручність використання БД SQLite, як "кишенькового" сховища результатів розрахування;
- використання обраної мови програмування перекриває всі потреби у реалізації моделі розрахування водно-сольового балансу.

2 ПРОЕКТНА ЧАСТИНА

2.1 Постановка завдання

В першу чергу оглянемо основні вимоги, що були висунуті до програмного продукту, а потім визначимо ряд задач, які необхідно вирішити для успішної розробки ПП. Отож основними вимогами, що були висунуті є:

1. Перегляд значень окремих складових розрахунку водно-сольового балансу у чотирьох таблицях в залежності від обраного року розрахунку.
2. Виконання розрахунку значень водного і сольового балансів озера Катлабух, надавши користувачу змогу поетапного проведення розрахунку:
   - ввести початкові данні;
   - розрахувати водний баланс, не враховуючи ДГРШ;
   - розрахувати водний баланс разом з ДГРШ;
   - розрахувати сольовий баланс.
3. Редагування даних розрахунку водного і сольового балансів, повторне проведення розрахунку.
4. Редагування значень, які приймають участь під час проведення розрахунку.
5. Збереження даних розрахунку водно-сольового балансу і тих значень, що приймають участь у проведенні розрахунку.
6. Експорт складових розрахунку до програмного продукту для перегляду електронних таблиць Microsoft Excel.
7. Надати можливість користувачу ПП формувати і редагувати діаграми на основі даних, що були розраховані. Серед діаграм повинні бути:
   - лінійний графік;
   - гістограма;
   - кругова діаграма.

8. Вивід зображення отриманої діаграми у графічний файл(.png, .jpg, .bmp).
9. Інтерфейс користувача повинен містити українську локалізацію.
10. ПП повинен мати довідкові матеріали з використання.
11. Інтерфейс користувача повинен мати рясну кількість підказок.

На основі розглянутих вимог до ПП, можна виділити основні за виділити наступний ряд задач:

1. Спроектувати інтерфейс користувача ПП.
2. Спроектувати і розробити сховище даних.
3. Спроектувати і розробити модель архітектури ПП.
4. Спроектувати і розробити компоненти, що будуть реалізовувати алгоритми проведення розрахункових обчислень.

Після виконання вище перелічених задач отримуємо повне уявлення того, як повинен виглядати ПП.

2.2 Проектування інтерфейсу користувача ПП

Спираючись на вимоги до ПП, можна сформувати функціонал програми, який зображено на діаграмі варіантів використання(див. рис. 1). На цій діаграмі можна побачити основні операції, які будуть доступні користувачу, і операції візьме які на себе візьме ПП. Наприклад така операція як збереження даних розрахунку буде автоматичним, тобто коли користувач буде створювати новий розрахунок і натискатиме кнопку "Провести розрахунок", то після завершення розрахункових операцій, програма створить нові записи у базі даних і занесе туди результати розрахунку. Такий самий алгоритм дій збереження даних буде виконуватися і при зміні даних розрахунків водного і сольового балансів. Це наддасть користувачу зберегти час, який би він витратив на натиск відповідної кнопки. Також такий механізм автоматичного збереження розрахованих або

змінених даних спростить передачу даних в окремі елементи відображення цих даних.

Переглядаючи діаграму варіантів використання ПП, можна виділити дві ярко виражені основні частини майбутнього інтерфейсу користувача. Перша частина – це робочий простір, де користувач буде виконувати дії над розрахунком, такі як перегляд даних розрахунку, змінення даних вже існуючого розрахунку, видалення і створення нового розрахунку, тобто виклик операції виконання розрахункових дій над тими даними, що введе користувач.

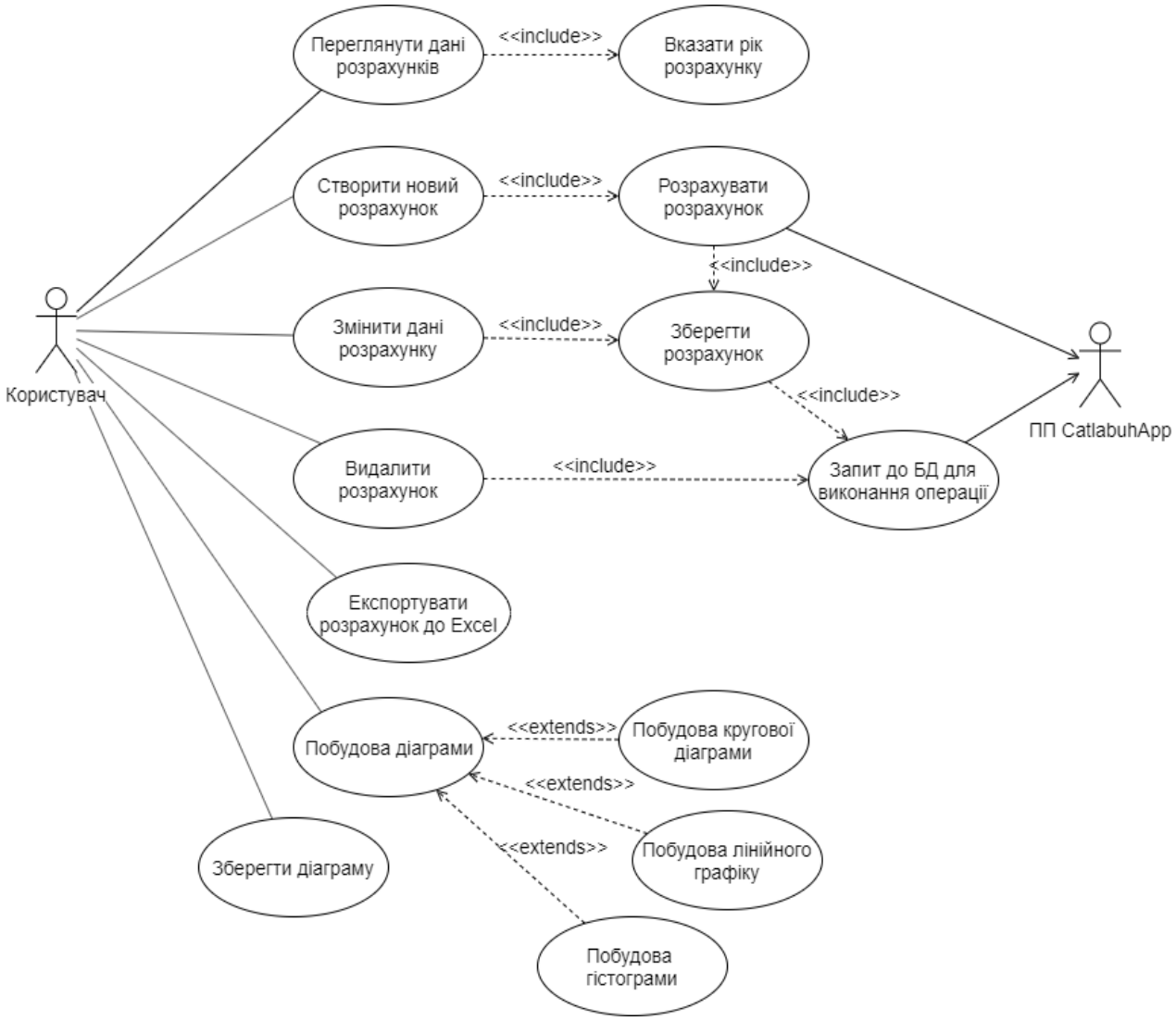


Рисунок 3 – Діаграма варіантів використання ПП

Також до цього списку слід додати експорт розрахунку до MS Excel. Для більш комфортної праці користувача з цією частиною інтерфейсу, розіб’ємо її на три зони: зону вхідних даних, зону ДГРШ і зону результуючих даних, що буде відображати весь розрахунок в цілому. Функціонал робочого простору, де користувач буде виконувати дії над розрахунком, буде відповідно розподілено(див. рис. 2).

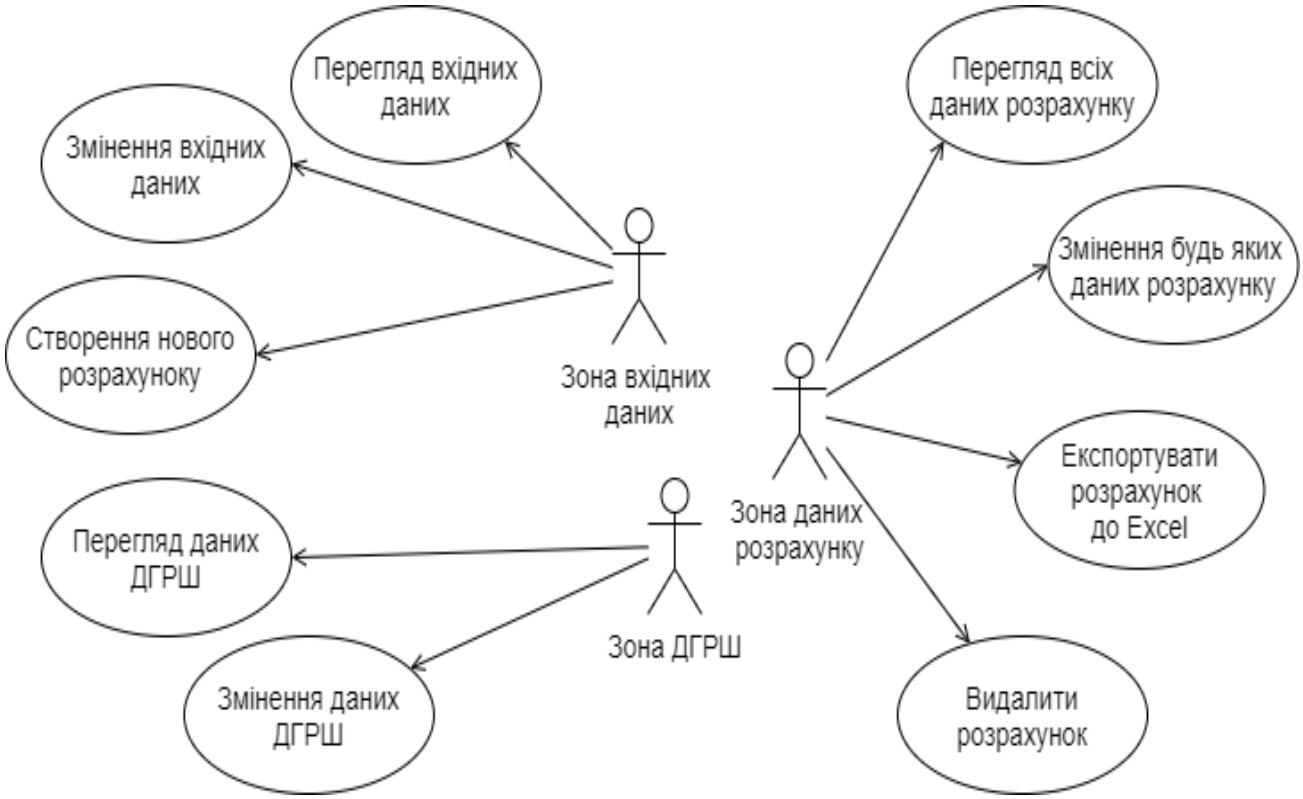


Рисунок 4 – Діаграма варіантів використання розподіленого простору для виконання дій над розрахунком

Другою частиною буде той робочий простір, де користувач буде використовувати з операції для праці з діаграмами. В цій частині користувачу треба надати можливість обирати, який тип діаграми він хоче побудувати, налаштовувати дані, що зображує діаграма, і змінення зовнішнього вигляду діаграми. Імпорт

зображення отриманої діаграми буде виконуватися з цього ж робочого простору.

2.3 Проектування сховища даних

Однією з ключових вимог до ПП є збереження результатів розрахунків водного і сольового балансів, було прийнято рішення зберігати всі дані розрахунку і допоміжні дані, що не є частиною розрахунку, але приймають участь у його проведені. Оглянемо більш детально модель розрахунку.

Розрахунок водного і сольового балансу озера Катлабух за конкретний рік являє собою чотири таблиці складових двох частин одного розрахунку. В такій формі доволі зручно аналізувати, проте це не доцільно чітко слідувати цій моделі при створенні сутностей сховища даних, тому треба виділити частини даних відповідно до їх ролей у розрахунку. Про аналізувавши алгоритм проведення розрахунку можна виділити основні сутності:

– вхідні дані розрахунку;
– дані ДГРШ;
– вихідні дані, які в свою чергу поділяються на дані частини балансів.

Кожна сутність відображає стан розрахунку відповідного року, тому до проектованого сховища додано і таку сутність, як Рік розрахунку.

Також треба пам’ятати про ті сутності, що будуть зберігати допоміжні дані. І тут може виникнути питання, а чи треба взагалі їх зберігати, якщо можна залишити їх у коді програми. Звергнемо увагу на те, що при розрахунку водного і сольового балансів озера Катлабух використовують ті допоміжні значення, які географічно відносяться до об’єкта розрахунків, тобто озера Катлабух. Але ця модель може використовуватися для розрахунків водно-сольових балансів і інших озер, і тому треба залишити вікно для подальшого узагальнення ПП в разі потреби.

Допоміжними даними, що беруть участь у розрахунку, є:

– таблиця внутрішньорічного розподілу для поверхневого стоку і припливу ґрунтових вод;
– таблиця ординат кривих трьох параметричного гама-розподілення;
– таблиця залежності площини водної поверхні від рівнів води в озері Катлабух;
– коефіцієнти формул розрахунку.

На діаграмі сутностей зображені всі вище перелічені сутності і їх відносини між собою(див. рис. 5). При розгляді діаграми, можна побачити сутності про які раніше не згадували, це: сутність Months і сутність CalculationInfo.

Зупинимося на першій сутності. Отже, кожна сутність-компонент розрахунку має певну кількість атрибутів, що відображують складові тієї чи іншої частини розрахунку і розрахунок виконується для певного року. Тому, щоб не виділяти один запис для кожного року і тим самим не створювати таблиці із безлічі атрибутів, було прийнято рішення зберігати дані одного розрахунку у чотирнадцяти записах відповідних таблиць. Чому саме чотирнадцять записів, якщо в році дванадцять місяців? Така кількість записів зумовлена тим, що окрім розрахунку водно-сольового балансу на кожен місяць року, ще виконується розрахунок сумарних і процентних значень для всього року. Таким чином отримаємо більш зручні для сприймання розробником таблиці БД.

А комбінація сутностей Month і Year утворюють зіставний первинний ключ для сутностей, що відображають вхідні, вихідні дані і дані ДГРШ.

Перейдемо до сутності CalculationInfo. Ця сутність необхідна тим, що є дані, які потребують того, щоб їх зберігали для кожного року розрахунку. Наприклад, цього потребує значення водності року, що розраховується, і значення показника середньої мінералізації на кінці грудня попереднього року, що використовується вхідне значення для розрахунку сольового балансу.

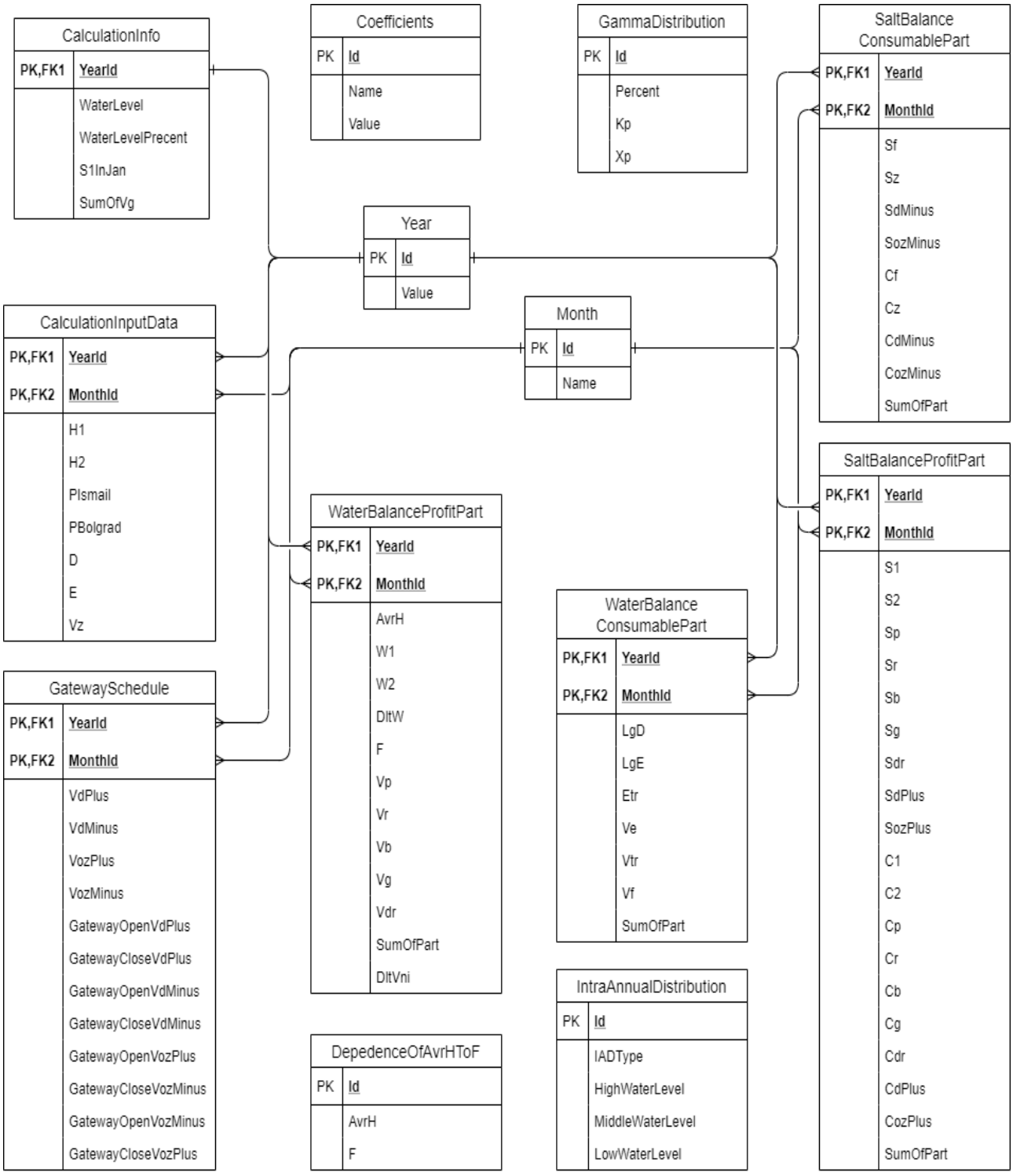


Рисунок 5 – Діаграма сутностей БД

Тому сутність CalculationInfo являє собою місцем збереження додаткових даних про розрахунок певного року. Так як записів в таблицях

WaterBalanceProfitPart, WaterBalanceConsumablePart, SaltBalanceProftitPart і SaltBalanceConsumablePart чотирнадцять для певного року розрахунку, то роль зв’язувального елемента для сутностей раніше перелічених сутностей і CalculationInfo буде виконувати запис у таблиці Years.

Розібравшись із сутностями БД проекту треба зазначити, що зокрема сутностей до моделі БД необхідно додати чотири представлення. Ці представлення будуть відображати таблиці розрахунку, так як для відтворення таблиць розрахунку буде потрібно об’єднати дані з різних сутностей БД відповідно до таблиць 1, 2, 3 і 4(див. рис. 6).

Рисунок 6 – Діаграма сутностей представлень таблиць розрахунку

2.4 Проектування архітектури ПП

Важливою частиною у розробці будь якого ПП відіграє організаційна структура програми. За наявності доброї архітектури сприяє подальшому супроводу і розширенню ПП, на відміну від погано спроектованої програми.

Основна ідея при проектуванні структури даного ПП стало відділення візуальної частини, тобто інтерфейсу користувача, від даних при умові, щоб можна було легко замінити візуальні частини. Ця ідея не нова і має безліч рішень у вигляді шаблонів проектування, таких як Model-View-Controller, Model-View-Presenter та Model-View-ViewModel.

Model-View-Controller – архітектурний шаблон, який використовується під час проектування та розробки програмного забезпечення[9][1)].

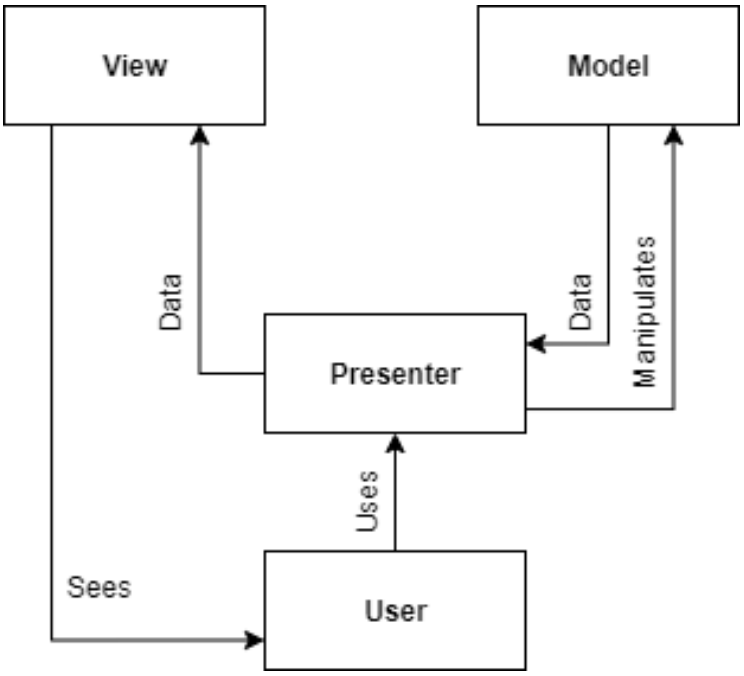


Рисунок 7 – Патерн MVC

[1)] [9] Модель-вид-контролер – Вікіпедія. URL: https://uk.wikipedia.org/wiki/Модель-вид-контролер. (дата звернення 05.09.2020).

Цей шаблон передбачає поділ системи на три взаємопов'язані частини: модель даних, вигляд (інтерфейс користувача) та модуль керування. Застосовується для відокремлення даних (моделі) від інтерфейсу користувача (вигляду) так, щоб зміни інтерфейсу користувача мінімально впливали на роботу з даними, а зміни в моделі даних могли здійснюватися без змін інтерфейсу користувача.

Мета шаблону – гнучкий дизайн програмного забезпечення, який повинен полегшувати подальші зміни чи розширення програм, а також надавати можливість повторного використання окремих компонентів програми. Крім того використання цього шаблону у великих системах сприяє впорядкованості їхньої структури і робить їх більш зрозумілими за рахунок зменшення складності.

У рамках архітектурного шаблону MVC програма поділяється на три окремі, але взаємопов'язані частини з розподілом функцій між компонентами. Модель (Model) відповідає за зберігання даних та їх структуру. Вигляд (View) відповідальний за представлення цих даних користувачеві, тобто інтерфейс програми. Контролер (Controller) керує компонентами, отримує сигнали у вигляді реакції на дії користувача (зміна положення курсора миші, натискання кнопки, ввід даних в текстове поле) і передає дані у модель[9][1)].

– Модель є центральним компонентом шаблону MVC і відображає поведінку застосунку, незалежну від інтерфейсу користувача. Модель стосується прямого керування даними, логікою та правилами застосунку.
– Вигляд може являти собою будь-яке представлення інформації, одержуване на виході, наприклад графік чи діаграму. Одночасно можуть співіснувати кілька виглядів (представлень) однієї і тієї ж інформації, наприклад гістограма для керівництва компанії й таблиці для бухгалтерії.

[1)] [9] Модель-вид-контролер – Вікіпедія. URL: https://uk.wikipedia.org/wiki/Модель-вид-контролер. (дата звернення 05.09.2020).

– Контролер одержує вхідні дані й перетворює їх на команди для моделі чи вигляду.

Модель інкапсулює ядро даних і основний функціонал їхньої обробки і не залежить від процесу вводу чи виводу даних.

Вигляд може мати декілька взаємопов'язаних областей, наприклад різні таблиці і поля форм, в яких відображаються дані.

У функції контролера входить відстеження визначених подій, що виникають в результаті дій користувача. Контролер дозволяє структурувати код шляхом групування пов'язаних дій в окремий клас. Наприклад у типовому MVC-проекті може бути користувацький контролер, що містить групу методів, пов'язаних з управлінням обліковим записом користувача, таких як реєстрація, авторизація, редагування профілю та зміна пароля.

Зареєстровані події транслюються в різні запити, що спрямовуються компонентам моделі або об'єктам, відповідальним за відображення даних. Відокремлення моделі від вигляду даних дозволяє незалежно використовувати різні компоненти для відображення інформації. Таким чином, якщо користувач через контролер внесе зміни до моделі даних, то інформація, подана одним або декількома візуальними компонентами, буде автоматично відкоригована відповідно до змін, що відбулися[9][1)].

Model-View-Presenter – шаблон проектування, похідний від MVC, що відділяє візуальне відображення та поведінку обробки подій у різні класи, а саме: Представлення (View) та Пред'явник (Presenter)(див. рис. 8).

Модель являє собою клас для визначення даних, які будуть відображатися або над якими будуть проводитися інші дії у інтерфейсі користувача[10][2)].

[1)] [9] Модель-вид-контролер – Вікіпедія. URL: https://uk.wikipedia.org/wiki/Модель-вид-контролер. (дата звернення 05.09.2020).

[2)] [10] Model-View-Presenter – Вікіпедія. URL: https://uk.wikipedia.org/wiki/Model-View-Presenter. (дата звернення 05.09.2020).

Клас Представлення управляє елементами на сторінці, та направляє події до класу Пред'явника.

Пред'явник містить логіку реагування на події, оновлює Модель (бізнес-логіки і даних з програми) і, в свою чергу, маніпулює станом Представлення. Для полегшення тестування Пред'явника, він повинен мати посилання на інтерфейс Представлення замість посилання на конкретну реалізацію. Як наслідок, ви можете легко замінити діюче Представлення на макет для виконання тестів[10][1)].

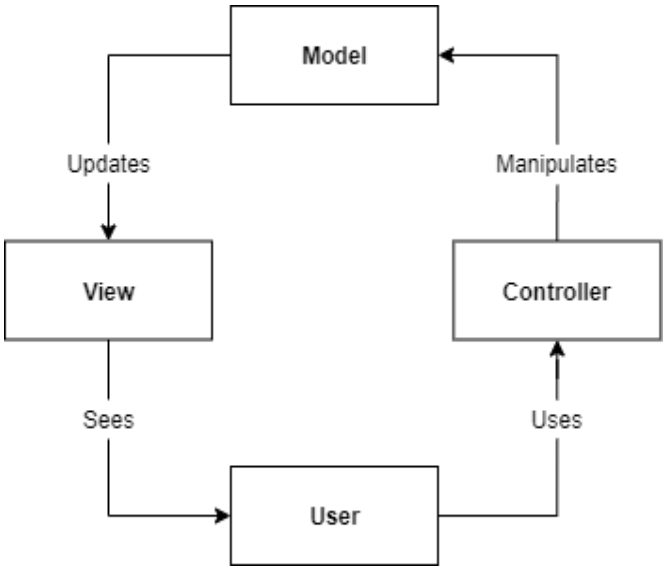


Рисунок 8 – Патерн MVP

Model-View-ViewModel – це шаблон проектування, що застосовується під час проектування архітектури застосунків (додатків). Публічно вперше був представлений Джоном Госсманом (John Gossman) у 2005 році як модифікація шаблону Presentation Model(див. рис. 9)[11][2)].

[1)] [10] Model-View-Presenter – Вікіпедія. URL: https://uk.wikipedia.org/wiki/Model-View-Presenter. (дата звернення 05.09.2020).

[2)] [11] Model-View-ViewModel – Вікіпедія. URL: https://uk.wikipedia.org/wiki/Model-View-ViewModel. (дата звернення 05.09.2020).

MVVM орієнтований на такі сучасні платформи розробки, як Windows Presentation Foundation та Silverlight від компанії Microsoft.

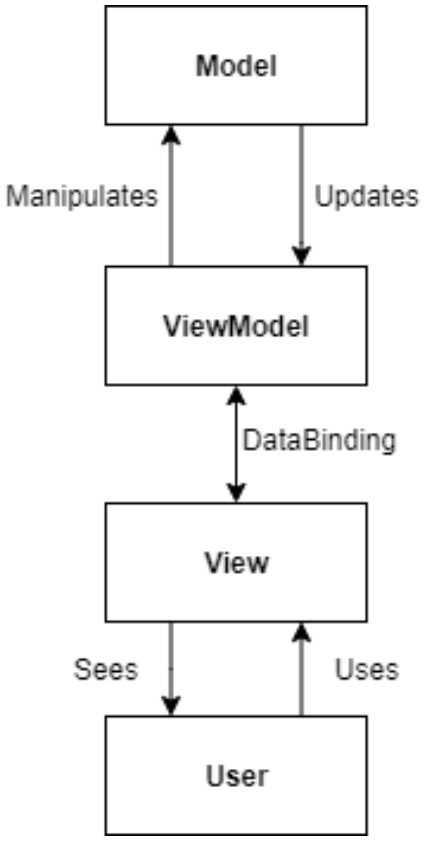


Рисунок 9 – Патерн MVVM

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 оброблює 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[11][1)].

Шаблон MVVM ділиться на три частини:

[1)] [11] Model-View-ViewModel – Вікіпедія. URL: https://uk.wikipedia.org/wiki/Model-View-ViewModel. (дата звернення 05.09.2020).

Модель (Model), як і в класичному шаблоні MVC, Модель являє собою фундаментальні дані, що необхідні для роботи застосунку.

Вид/(Вигляд) (View) як і в класичному шаблоні MVC, Вигляд — це графічний інтерфейс, тобто вікно, кнопки тощо.

Модель вигляду (ViewModel, що означає «Model of View») з одного боку є абстракцією Вигляду, а з іншого надає обгортку даних з Моделі, які мають зв'язуватись. Тобто вона містить Модель, яка перетворена до Вигляду, а також містить у собі команди, якими може скористатися Вигляд для впливу на Модель. Фактично ViewModel призначена для того, щоб:

– Здійснювати зв'язок між моделлю та вікном;

– Відслідковувати зміни в даних, що зроблені користувачем;

– Відпрацьовувати логіку роботи View (механізм команд)[11][1)].

За основу візьмемо патерн MVP, так як він повністю перекриває всі потреби для організації ПП. Проте зробимо декілька правок і розділимо програму на дві частини: ядро(Core) і інтерфейс користувача(UI), тобто візуальну частину.

Ядро програми буде повністю побудовано на патерні MVP, але реалізація інтерфейсів представлень буде знаходитися окремо від їх інтерфейсів. Тобто вся бізнес-логіка програми буде знаходитися окремо від візуальної частини і наддасть змогу змінювати візуальну частину на будь-яку. Таким чином така компоновка розв’язує руки вибору платформи для візуальної частини, тому що один й той самий ПП може бути розгорнуто як звичайний програмний додаток, що встановлено на персональному комп’ютері користувача, або як WEB-додаток, і в незалежності від платформи програма буде вести себе однаково. Все що вимагається від візуальної частини – це реалізація відповідних інтерфейсів.

[1)] [11] Model-View-ViewModel – Вікіпедія. URL: https://uk.wikipedia.org/wiki/Model-View-ViewModel. (дата звернення 05.09.2020).

І на основі цієї інформації у ядро ПП буде входити такі пакети:

– DataAccess – це пакет, в якому буде знаходитися контекст БД розрахунків водного і сольового балансів, а також клас, який створить БД, якщо її не буде знайдено ПП або при першому запуску програми. Причиною створення цих класів є націляння використовувати бібліотеку Entity Framework в замін ADO.NET. Вибір Entity Framework наддасть високий рівень абстрагування від самої БД і працювати с даними в незалежності від типу сховища;
– Models – це пакет, де будуть знаходитися базовий інтерфейс для всіх моделей і класи, що його реалізують. Кожен екземпляр моделі буде відображати запис у відповідній таблиці БД, а вся таблиця буде у вигляді списку і знаходитиметься у класі WSBCContext. Такий вигляд рівня даних обумовлено тим, що планується використовувати бібліотеку EntityFramework і підхід Code First.

  Presenters – це пакет, де будуть знаходитися базовий інтерфейс пред'явника і класи, що його реалізують. Кожен клас матиме посилання на представлення за яким він закріплений, відповідатиме за обробку подій свого представлення і виклику спеціальних методів для оновлення стану як моделей так і представлення.
– Services – це пакет, де будуть знаходитися базовий інтерфейс сервісів пред'явника і класи, що його реалізують, будуть використовуватися для розвантаження класів пред'явників. Наприклад, в один із сервісів може реалізовувати логіку формування документу для експорту до Excel.
– Views – це пакет, де будуть знаходитися лише базовий інтерфейс представлення і інтерфейси представлень. При реалізації похідних інтерфейсів, представлення повинно буде мати екземпляр свого пред'явника, щоб забезпечити зв'язок для обробки та оновлення даних.

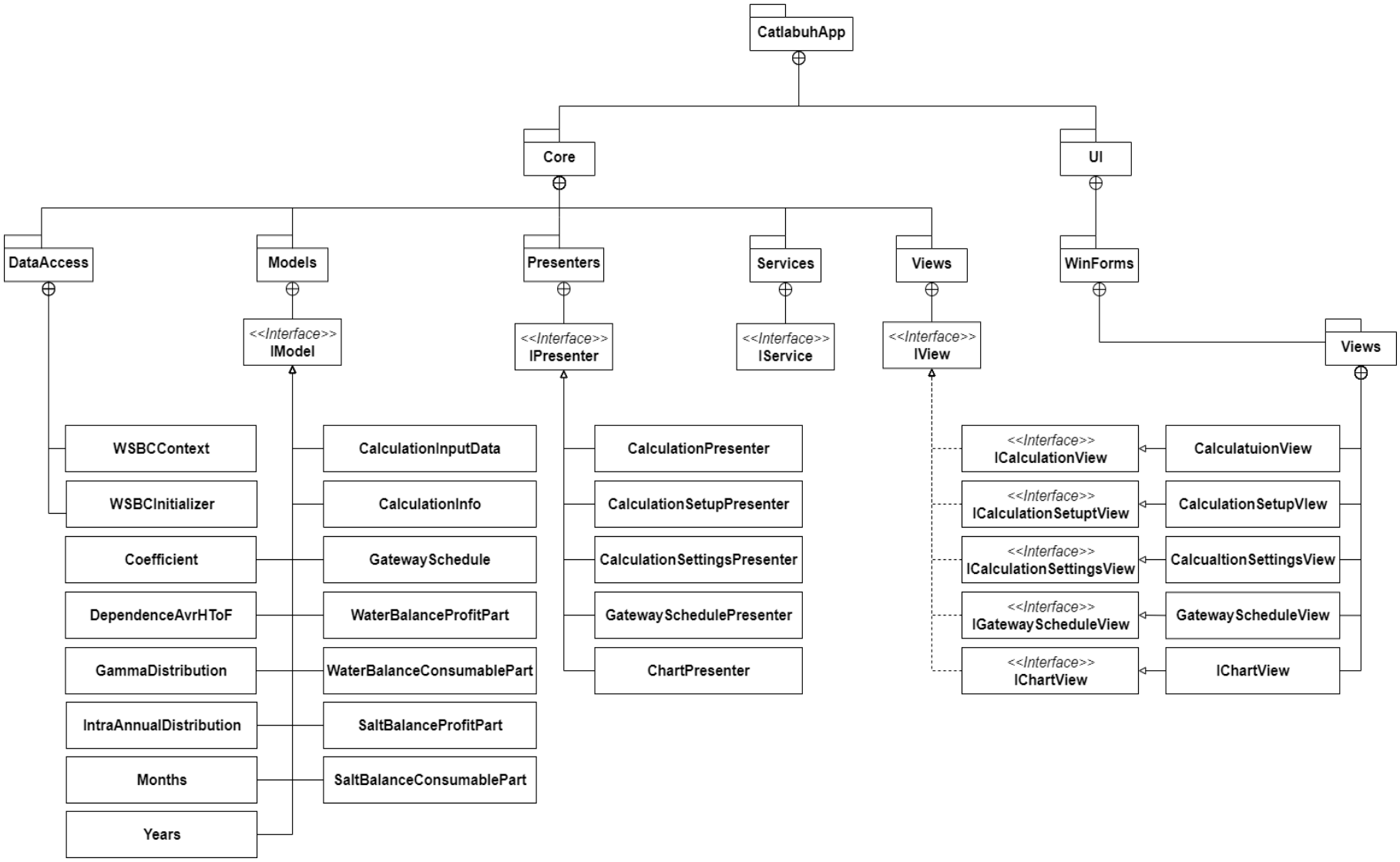


Рисунок 10 – Діаграма пакетів ПП

Слід звернути увагу, що на діаграмі зображено тільки основні пакети, проте в ході розробки і реалізації тих чи інших рішень діаграма буде поповнюватися і іншими допоміжними пакетами.

2.5 Проектування компонентів, що реалізують розрахунок

Проведення розрахунку водного і сольового балансів, не представляє нічого складного, але потребує велику кількість вхідних даних. Тут стає питання як представити ці дані?

Спробуємо дати відповідь на з першого погляду просте питання.

Відомо, що розрахунок водного і сольового балансів складається з трьох частин: вхідні дані, дані ДГРШ і вихідних даних. І згідно діаграми пакетів(див. рис. 10), маємо ряд класів-моделей даних, що зберігаються у відповідному сховищі, які можна було б використати для маніпуляції з проведення розрахунку, а сам процес розрахунку віддати пред'явнику чи його сервісу. При використані такого підходу буде реалізовано менше класів, що заощадить час розробки, але позбавить ПП модульної незалежності компонентів, яку бажано все ж таки зберегти.

В такому випадку з'являється інший варіант, а саме винести модель розрахунку водного і сольового балансу, не плутати з моделями даних БД, у окремий пакет. Для цього потребується реалізувати класи і структури об'єктів розрахунку. І таким чином буде збережено модульність компонентів. А діаграма пакетів компоненту, що розробляється, матиме наступний вигляд(див. рис. 11).

В пакеті Structs знаходяться структури, що описують ДГРШ(GatewaySchedule) і вхідні дані розрахунку(InputData). Вхідні дані містять інформацію про показання рівнів води на початку(H1) і кінця місяців(H2), опадів виміряних на м/ст(P), дефіциту вологості(D), випаровування(E), збору води на зрошування(Vz), сумарні значення опадів(SumOfP) і об'ємів припливу ґрунтових вод(SumOfVg), а також середнє значення міне-

ралізації по озеру на початку місяця(S1InJan). Окремо можна виділити списки К1, К2 і К3 – ці списки містять значення, які враховуються при розрахунку логарифму випаровування і транспірації(див. табл. 8).

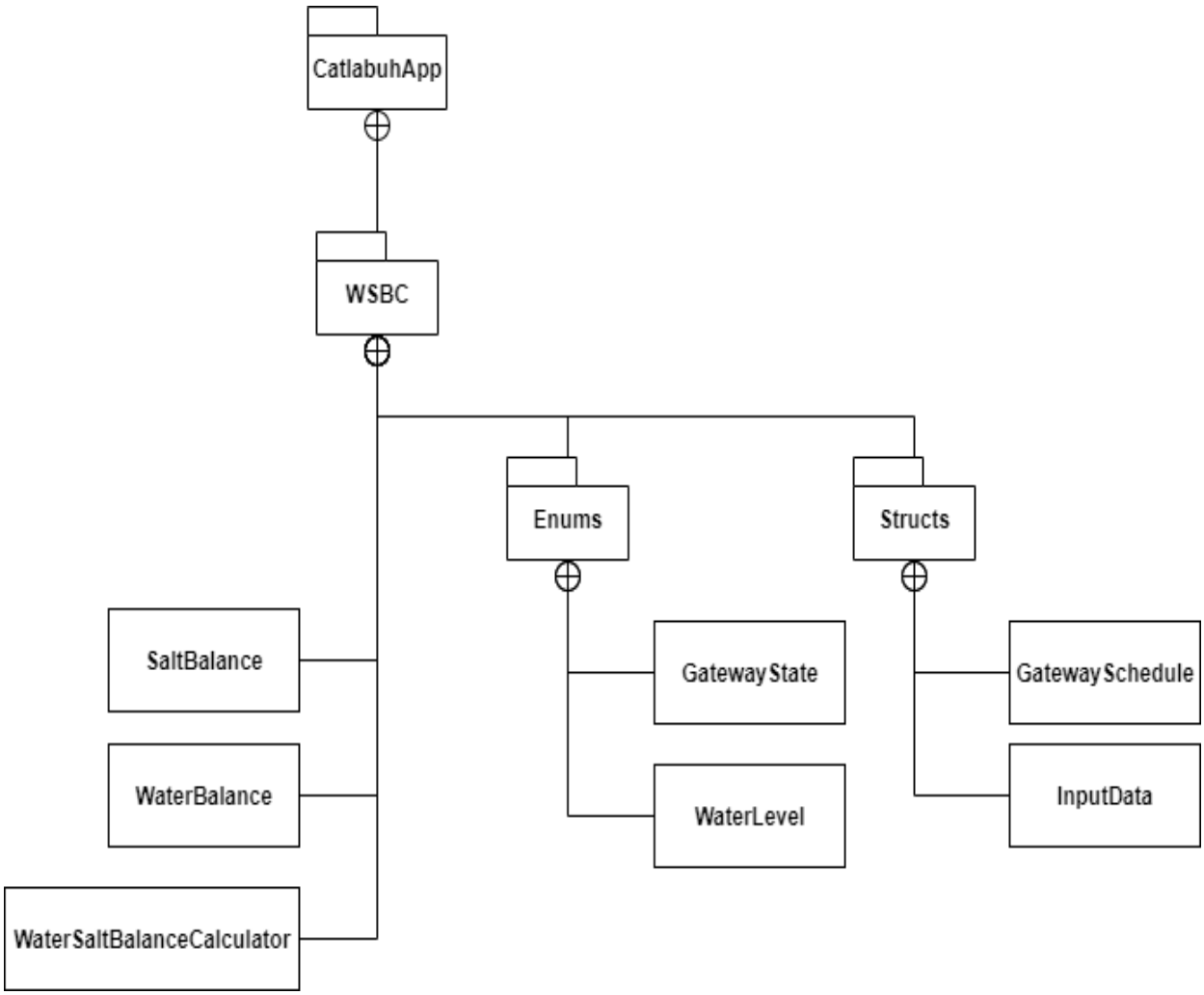


Рисунок 11 – Діаграма пакету WSBC

Більш детальну структуру пакету розрахунку зображено на діаграмі класів(див. рис. 12).

Пакет WSBC містить WaterSaltBalanceCalculator, цей об’єкт являє собою публічний клас, через який буде виконуватися всі операції по розрахунку, щоб не робити з нього один "всемогутній" клас, частини операцій розрахування буде делеговано класам WaterBalance і SaltBalance. Ці два класи по-

винні мати методи, що розраховують водний і сольовий баланси відповідно, а також бути захищеними від доступу зовні пакету.

Щоб здійснювати налаштування проведення розрахунку в класі присутні три булевих прапора, які відповідають за розрахунок значень випаровування (IsCalculateE), розрахунок значень об'ємів у ДГРШ(IsDistributeGSValues), тобто потрібно буде розподілити значення DltVni по складовим ДГРШ(цей варіант можливий лише якщо були вказані місяці, коли працювали шлюзи), і чи взагалі враховувати дані з ДГРШ про виконанні розрахунку водного і сольового балансів(IsGSTakenIntoAccount).

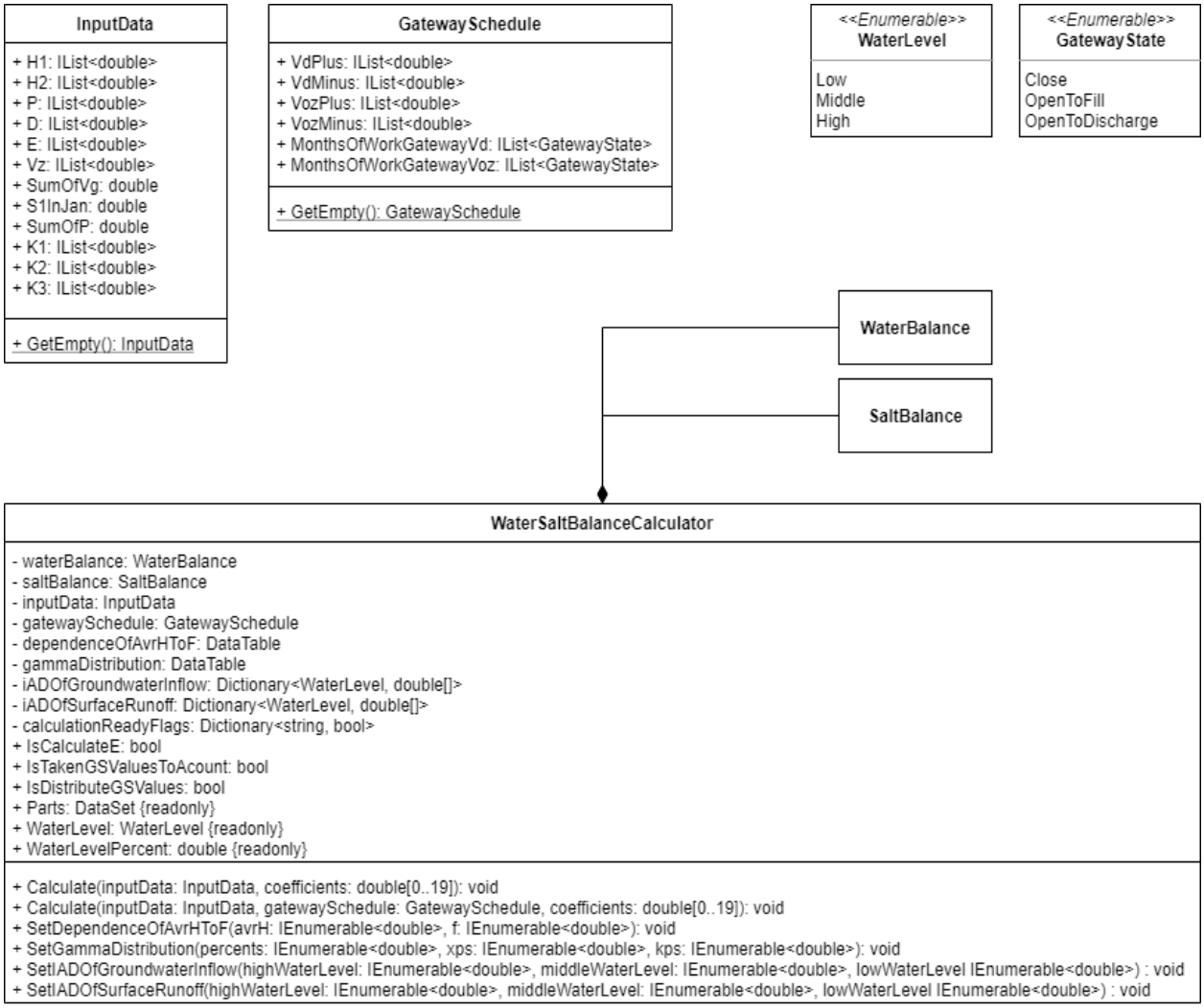


Рисунок 12 – Діаграма класів пакету WSBC

Для досягнення більшої незалежності від цільової платформи пакет WSBC потрібно буде розробляти як бібліотеку .NET Standart.

.NET Standard – це офіційна специфікація для API .NET, які повинні бути доступні у всіх реалізаціях .NET. .NET Standard була створена для того, щоб підвищити узгодженість в екосистемі .NET[12][1)].

Бібліотеки .NET Standard замінюють концепції специфічних для платформи і переносяться бібліотек. Вони є специфічними для платформи в тому сенсі, що вони надають всі функціональні можливості базової платформи (без штучних платформ або пересічний платформ). Вони є переносяться в тому сенсі, що вони працюють на всіх підтримуваних платформах.

.NET Standard надає набір бібліотечних контрактів. Реалізації .NET повинні підтримувати кожен контракт повністю або не підтримувати взагалі. Таким чином, кожна реалізація підтримує набір стандартних .NET контрактів. Наслідком з цього є те, що кожна бібліотека класів .NET Standard підтримується на платформах, які підтримують її контрактні залежності.
Контракти .NET Standard не пропонують всі функції платформи .NET Framework (і це не є метою), але вони надають набагато більше API, ніж переносяться бібліотеки класів. Згодом будуть додані додаткові API.

Бібліотеки .NET Standard підтримуються в наступних платформах:

- .NET Core;
- .NET Framework;
- Mono;
- Xamarin.iOS, Xamarin.Mac, Xamarin.Android;
- Універсальна платформа Windows (UWP);
- Windows;
- Windows Phone та Windows Phone Silverlight[13][1)].

[1)] [12] Библиотеки классов .NET | Microsoft Docs. URL: https://docs.microsoft.com/ru-ru/dotnet/standard/class-libraries. (дата звернення 08.10.2020).

[1)] [13] NET Standard | Microsoft Docs. URL: https://docs.microsoft.com/ru-ru/dotnet/standard/net-standard. (дата звернення 08.10.2020).

3 ПРОЕКТНІ ТА ТЕХНІЧНІ РІШЕННЯ

3.1 Технічні рішення при створенні компоненту, що проводить розрахунок

Згідно діаграми класів бібліотеки WSBC(Water Salt Balance Calculator) були створені базові структури, що відображають вхідні дані і дані ДГРШ, а також перечислення стану шлюзів і водності року(див. рис. 12).

Як було сказано раніше, з об’єкту WaterSaltBalanceCalculator прийнято рішення не робити всеосяжний тип, тому логіку розрахування делеговано класам WaterBalance і SaltBalance. Ці класи видимі лише в рамках бібліотеки WSBC і їх стан неможливо побачити зовні, таке рішення надає у безпеку даного компоненту ПП.

Отож перейдемо до огляду типів WaterBalance і SaltBalance, а потім до їх оболонки WaterSaltBalanceCalculator.

Почнемо з WaterBalance. WaterBalance – це клас який відповідає за розрахунок водного балансу і серед полів має масиви, що описують компоненти таблиць 1 та 2.

Додатково тут присутні поля: залежності середнього значення рівня вологості від площі водного дзеркала, трьох параметричного гама-розподілення та внутрішньо-річних розподілів поверхневого стоку і припливу ґрунтових вод, що ініціалізуються у конструкторі класу.

```
private readonly DataTable _dependenceOfAvrHToF;
private readonly DataTable _gammaDistribution;
private readonly Dictionary<WaterLevel, double[]>
_iADOfSurfaceRunoff;
private readonly Dictionary<WaterLevel, double[]>
_iADOfGroundwaterInflow;
```

Окремо були додані приватні булеві прапори, що вказають на завершеність етапу розрахування РВСБ. Наприклад, неможливо почати розрахунок витратної частини водного балансу, якщо не було встановлені значення при-

буткової частини, які встановлюються при виклику метода розрахунку прибуткової частини.

```
private bool _isWaterLevelSetted;
private bool _isSquareOfWaterMirrorSetted;
private bool _isSumOfVrSetted;
private bool _isProfitPartCalculated;
```

У класі задекларовані наступні методи:

- internal void CalculateProfitPart(double sumOfVg, double k1W = 67.098, double k2W = 17.278, double kVb = 0.23, double kVdr = 0.2);
- internal void CalculateConsumablePart(bool isCalculateE, double[] k1, double[] k2, double[] k3, double kSumEtr = 1.14, double kVtr = 0.3, double kVf = 0.55);
- internal void CalculateDltVni(GatewaySchedule schedule, bool isTakenToAcountGSValues, bool isDistributeGSValues);
- internal void SetSquareOfWaterMirror();
- internal void SetSumOfVr(double sumOfP, double q = 0.27, double fr = 1060, double k = 86400, int numOfDaysInYear = 365);
- internal void SetWaterLevel(double sumOfP);
- private void DistributeGSValues(GatewaySchedule schedule).

Перші три метода відповідають саме за проведення розрахунку. Наступні три метода в списку відповідають за перед розрахункові операції, такі як встановлення площі водного дзеркала, річного об’єму річкового стоку та встановлення водності розрахункового року. Останній захищений метод розподіляє значення об’ємів ДГРШ. Серед вхідних параметрів методи мають параметри коефіцієнтів, яким завдані значення за замовчуванням, які при необхідності можуть бути змінені при виклику того чи іншого метода. Таке рішення було прийнято через бажання замовника змінювати ці значення.

SaltBalance – це клас, який відповідає за обробку операцій розрахування сольового балансу РВСБ і серед полів, відповідно до таблиць 3 і 4, має масиви, що відображають дані компонентів сольового балансу, за аналогією з класом WaterBalance.

Для проведення розрахунку сольового балансу спирається на дані водного балансу, тому до полів класу було додано поле-посилання на об’єкт кла-

су WaterBalance і ініціалізація проводиться через конструктор класу SaltBalance.

```
private readonly WaterBalance _waterBalance;
```

В класі задекларовано один єдиний метод, що розраховує прибуткову і витратну частини разом. Це обумовлено тим, щоб розрахувати компоненти наступного місяця потрібно розрахувати всі компоненти прибуткової і витратної частин поточного місяця.

Нижче наведено тіло методу.

```
    for (int i = 0; i < WaterSaltBalanceCalculator.MonthCount;
i++)  {
    s1[i] = (i == 0) ? s1InJan : s2[i - 1];
    c1[i] = (i == 0) ? _waterBalance.w1[0] * s1[0] : c2[i - 1];
    sp[i] = kSp;
    sr[i] = k1Sr * Math.Exp(k2Sr * _waterBalance.vr[i]);
    sb[i] = sr[i] * kSb;
    sg[i] = kSg;
    sdr[i] = kSdr;
    sdPlus[i] = kSdPlus;
    sozPlus[i] = 0;
    cp[i] = _waterBalance.vp[i] * sp[i];
    cr[i] = _waterBalance.vr[i] * sr[i];
    cb[i] = _waterBalance.vb[i] * sb[i];
    cg[i] = _waterBalance.vg[i] * sg[i];
    cdr[i] = _waterBalance.vdr[i] * sdr[i];
    cdPlus[i] = _waterBalance.vdPlus[i] * sdPlus[i];
    cozPlus[i] = _waterBalance.vozPlus[i] * sozPlus[i];
    sumOfProfitPart[i] = cp[i] + cr[i] + cb[i] + cg[i] + cdr[i]
+ cdPlus[i] + cozPlus[i];
    sf[i] = s1[i];
    sz[i] = s1[i];
    sdMinus[i] = s1[i] * kSdMinus;
    sozMinus[i] = s1[i];
    cf[i] = _waterBalance.vf[i] * sf[i];
    cz[i] = _waterBalance.vz[i] * sz[i];
    cdMinus[i] = _waterBalance.vdMinus[i] * sdMinus[i];
    cozMinus[i] = _waterBalance.vozMinus[i] * sozMinus[i];
    sumOfConumablePart[i] = cf[i] + cz[i] + cdMinus[i] +
cozMinus[i];
    c2[i] = c1[i] + sumOfProfitPart[i] - sumOfConumablePart[i];
    s2[i] = c2[i] / _waterBalance.w2[i];
}
```

Розібравшись з класами, що містять логіку розрахування РВСБ, перейдемо до публічного класу-оболонки WaterSaltBalanceCalculator через який порозводиться налаштування і виконання розрахунку.

Згідно до діаграми класів(див. рис. 12) у класі задекларовано шість публічних методів, останні чотири з яких, відповідають за встановлення значень у поля, що відображають допоміжні дані розрахунку. А перші два методи використовуються для виконання розрахунку і мають вигляд перевантаженого методу з назвою Calculate.

Вихідні дані розрахунку зберігаються у публічному полі Parts, який э набор даних і містить колекцію з чотирьох таблиць розрахунку типу DataTable. Після завершення операцій розрахування вихідні дані записуються у таблиці, за допомогою виклику метода приватного метода FillOutputDataTables().

Для навігації по таблицях, пропонується користуватися їх скороченим ім’ям, але щоб не допустити помилок при виклику тієї чи іншої таблиці за іменем, були створені чотири константи, які зберігають правильні імена таблиць. І при виклику таблиць розрахування радиться використовувати саме ці константи.

```
public const string WBPPTableName = "WBPP";
public const string WBCPTableName = "WBCP";
public const string SBPPTableName = "SBPP";
public const string SBCPTableName = "SBCP";
```

3.2 Технічні рішення при створенні шару даних

Для того щоб власноруч не прописувати запроси до БД і робити велику кількість помилок при їх створенні буде використано ORM Entity Framework 6, що наддасть змогу швидко і зручно налаштувати "зв’язок" з БД.

Entity Framework являє спеціальну об'єктно-орієнтовану технологію на базі фреймворка .NET для роботи з даними. Якщо традиційні засоби ADO.NET дозволяють створювати підключення, команди та інші об'єкти для

взаємодії з базами даних, то Entity Framework являє собою більш високий рівень абстракції, який дозволяє абстрагуватися від самої бази даних і працювати з даними незалежно від типу сховища. Якщо на фізичному рівні ми оперуємо таблицями, індексами, первинними і зовнішніми ключами, але на концептуальному рівні, який нам пропонує Entity Framework, ми вже працює з об'єктами.

Перша версія Entity Framework – 1.0 вийшла ще в 2008 році і представляла дуже обмежену функціональність, базову підтримку ORM (object-relational mapping - відображення даних на реальні об'єкти) і один єдиний підхід до взаємодії з бд – Database First. З виходом версії 4.0 у 2010 році багато чого змінилося – з цього часу Entity Framework став рекомендованої технологією для доступу до даних, а в сам фреймворк були введені нові можливості взаємодії з БД – підходи Model First і Code First.

Додаткові поліпшення функціоналу пішли з виходом версії 5.0 в 2012 році. І нарешті, в 2013 році був випущений Entity Framework 6.0, що володіє можливістю асинхронного доступу до даних.

Центральної концепцією Entity Framework є поняття сутності або entity. Сутність представляє набір даних, асоційованих з певним об'єктом. Тому дана технологія передбачає роботу не з таблицями, а з об'єктами і їх наборами.

Будь-яка сутність, як і будь-який об'єкт з реального світу, має низку властивостей. Наприклад, якщо сутність описує людини, то ми можемо виділити такі властивості, як ім'я, прізвище, зріст, вік, вага. Властивості необов'язково представляють прості дані типу int, а й можуть представляти більш комплексні структури даних. І у кожної сутності може бути одна або кілька властивостей, які будуть відрізняти цю сутність від інших і будуть унікально визначати цю сутність. Подібні властивості називають ключами.

При цьому суті можуть бути пов'язані асоціативної зв'язком один-ко-многих, один-ко-одному і багато-до-багатьох, подібно до того, як в реальній базі даних відбувається зв'язок через зовнішні ключі.

Відмінною рисою Entity Framework є використання запитів LINQ для вибірки даних з БД. За допомогою LINQ ми можемо не тільки отримувати певні рядки, що зберігають об'єкти, з БД, а й отримувати об'єкти, пов'язані різними асоціативними зв'язками.

Іншим ключовим поняттям є Entity Data Model. Ця модель зіставляє класи сутностей з реальними таблицями в БД.

Entity Data Model складається з трьох рівнів: концептуального, рівень сховища і рівень зіставлення (маппінга).

На концептуальному рівні відбувається визначення класів сутностей, які використовуються в додатку.

Рівень сховища визначає таблиці, стовпці, відносини між таблицями і типи даних, з якими порівнюється використовувана база даних.

Рівень зіставлення (маппінга) служить посередником між попередніми двома, визначаючи зіставлення між властивостями класу суті і стовпцями таблиць.

Таким чином, ми можемо через класи, визначені у додатку, взаємодіяти з таблицями з бази даних.

Entity Framework передбачає три можливі способи взаємодії з базою даних:

- Database first: Entity Framework створює набір класів, які відображають модель конкретної бази даних;
- Model first: спочатку розробник створює модель бази даних, по якій потім Entity Framework створює реальну базу даних на сервері;
- Code first: розробник створює клас моделі даних, які будуть зберігатися в БД, а потім Entity Framework за цією моделлю генерує базу даних і її таблиці[14][1)].

Проте нажаль Entity Framework 6 не підтримує можливість створювати БД чи таблиці у СКБД SQLite. Для вирішення цієї проблеми було вирішено

[1)] [14] Введение в Entity Framework 6. URL: https://metanit.com/sharp/entityframework/1.1.php. (дата звернення 08.10.2020).

піти наступним шляхом: створити клас, що містить код створення БД ПП і викликати метод, який буде виконувати всю роботу, при умові, якщо БД не існує. А сам процес спілкування додатку з БД буде портікати вже із використанням Entity Framework.

3.2.1 Реалізація підходу Code-First створення до БД

На діаграмі пакетів ПП в пакеті DataAccess зображено клас WSBCIntializer. Цей клас буде відповідати за створення і ініціалізацію БД, якщо її не буде існувати.

Клас має два статичних методи. Перший публічний в межах ядра ПП метод CreateIfNotExists – містить всю логіку створювання БД і повертає булеве значення, що символізує, а чи була потреба в створенні БД. І другий метод LoadConnectionString повертає рядок з'єднання з БД, що вказано у файлі конфігурацій ПП.

Розглянемо метод CreateIfNotExists детальніше. В першу чергу метод перевіряє існування БД, через умову, що наведено нижче.

```
if (!new WSBCContext().Database.Exists())
{
    Directory.CreateDirectory(Directory.GetCurrentDirectory() +
@"\Data");
    SQLiteConnection.CreateFile(Directory.GetCurrentDirectory()
+ @"\Data\users.db");

    using (var cnn = new
SQLiteConnection(LoadConnectionString()))
    {
        cnn.Open();

        string sql = ...;

        var cmd = new SQLiteCommand(sql, cnn);
        cmd.ExecuteNonQuery();

        cnn.Close();
    }
}
```

Якщо БД не існує, тобто умова вірна, то створюється директорія, де буде розміщено файл БД з назвою Data у кореневій директорії ПП. Після чого створюється файл БД, встановлюється з’єднання і виконується команда-запит до БД на створення таблиць і представлень відповідно до діаграм сутностей на рисунках 5 і 6.

Вирішивши одну проблему, настає інша. Тепер треба забезпечити виклик методу створювача БД перед викликом контексту сховища даних, так як контекст посилається на вже існуючу БД.

С тієї мети у пакет представників Presenters(див. рис. 10) буде додано абстрактний клас Presenter, який буде прабатьком для всіх інших представників і реалізує інтерфейс-маркер IPresenter. У класі Presenter оголошено захищене поле контексту, яке ініціалізується у конструкторі класу. А до цього викликається метод створювача БД CreateIfNotExists.

```
public abstarct class Presenter : IPresenter
{
    protected WSBCContext _db;

    protected Presenter()
    {
        WSBCInitializer.CreateDatabaseIfNotExists();

        _db = new WSBCContext();
    }
}
```

Таким чином вирішується проблема створення за необхідністтю БД ПП, так як всі подальші представники, що наслідують клас Presenter, матимуть доступ до контексту, що посилається на існуючу БД.

3.3 Технічні рішення при створенні інтерфейсу до ядра ПП

Ядро програми CatlabuhApp було вирішено робити окремим від візуальної частини компонентом, щоб досягти незалежності від конкретної реалізації інтерфейсу користувача. Але для цього треба забезпечити ядро шаром представлень, який буде розміщуватися у пакеті Views. В цьому пакеті будуть створені інтерфейси представлень через які буде відбуватися взаємодія між ядром і візуальною частиною.

Відповідно до діаграми варіантів використання розподіленого простору для виконання дій над розрахунком(див. рис. 4) всі представлення повинні будуть обробляти дії, що зображені на рисунку 13.

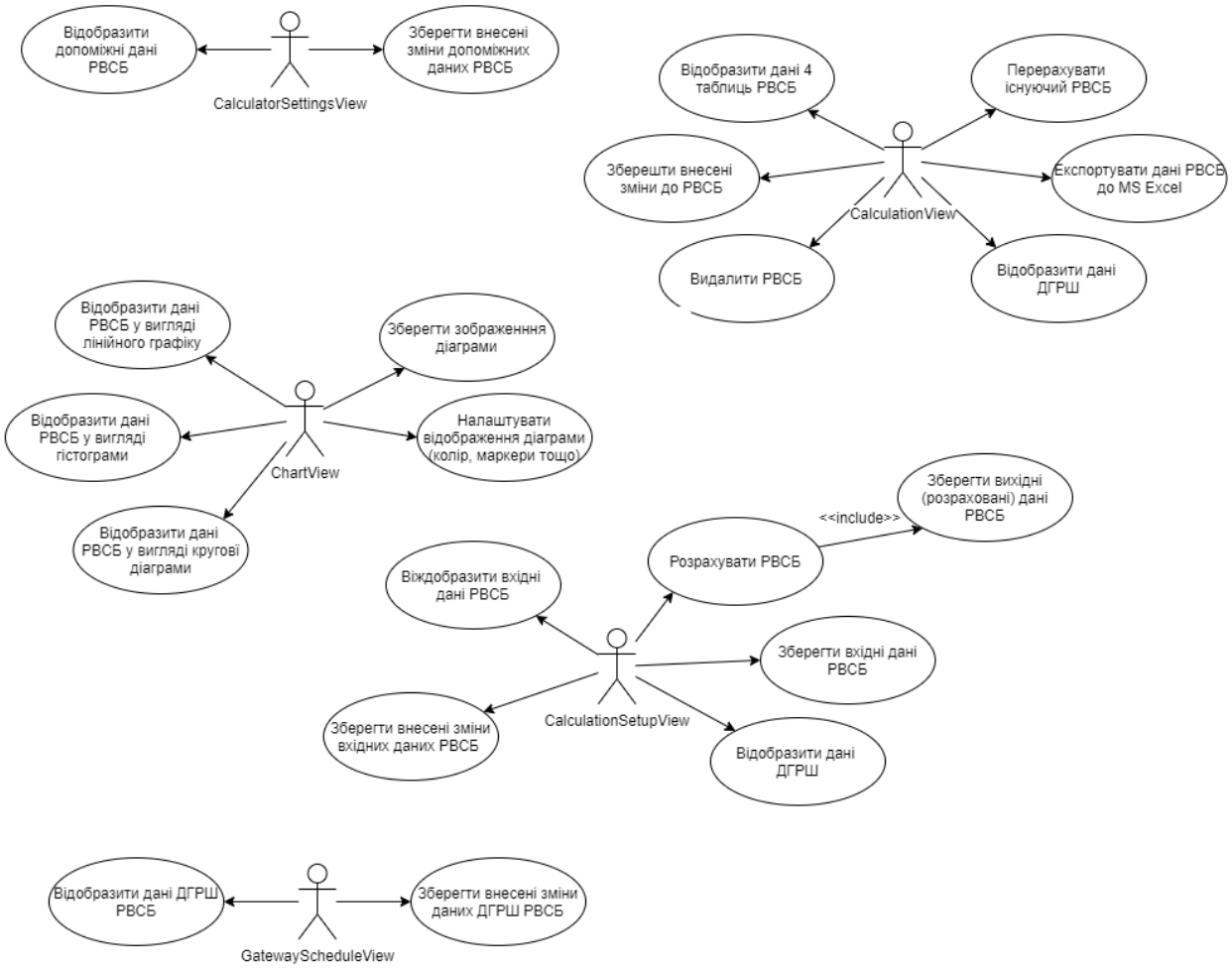


Рисунок 13 – Діаграма варіантів використання представлень ПП

У пакеті Views будуть розміщенні наступні інтерфейси(див. рис. 14):

– IView – базовий інтерфейс представлення;
– ICalculationView – представлення розрахунку;
– ICalculationSetupView – представлення встановлення вхідних даних розрахунку;
– IGatewayScheduleView – представлення ДГРШ;
– ICalculationSettingsView – представлення встановлення допоміжних даних, що беруть участь у розрахунку;
– IChartView – представлення, що оброблятиме налаштування побудови діаграми.

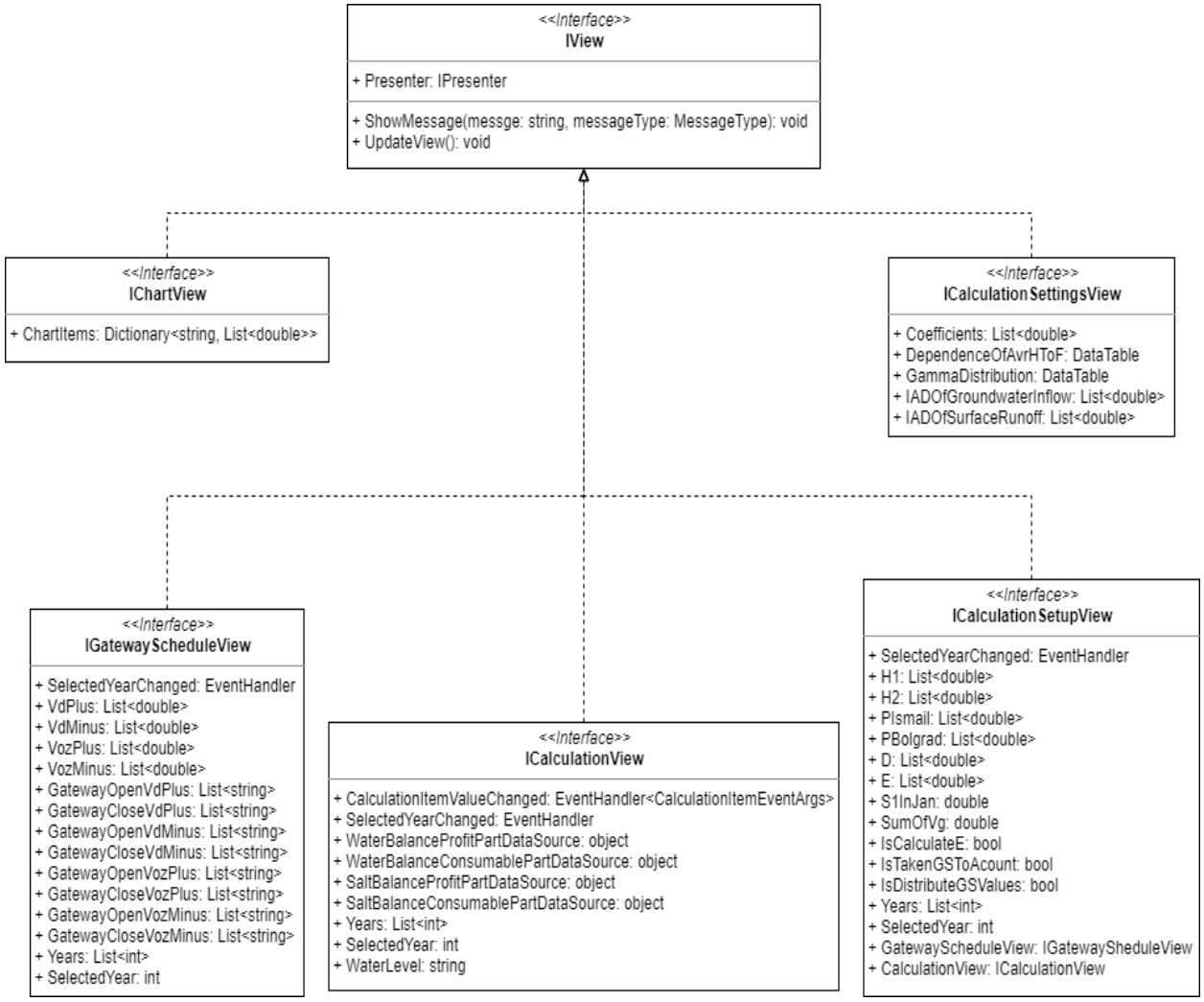


Рисунок 14 – Діаграма класів пакету CatlabuhApp.Core.Views

Базовий інтерфейс має два методи: метод оновлення представлення і метод, що показує користувачу деяке повідомлення. Також тут присутнє публічне поле, що є посиланням на представника представлення, що обробляє запити до БД, тощо. Інші інтерфейси представлень розширюють базовий інтерфейс публічними полями, що відображають дані, які будуть відтворюватися у представленні.

3.4 Технічні рішення при створенні шару представлень

Встановивши, які інтерфейси представлень мають місце бути і спираючись на діаграму використання представлень ПП(див. рис. 13), а також на діаграму пакетів ПП(див. рис. 10) були створені представники представлень, відповідно шаблону проектування, що було обрано.

Насамперед у пакет Presenters було додано базовий інтерфейс представника, який немає жодного методу, тому що цей інтерфейс призначений для маркування цілого ряду класів типом Presenter.

Відповідно до кожного представлення були створені представники, які мають ряд методів, які забезпечують зв’язок між візуальною частиною та ядром програми.

Розглянемо представник CalculationPresenter, що закріплено до представлення ICalculationView. Представник має три приватних поля, ініціалізація яких відбувається у конструкторі класу, який приймає один вхідний параметр, а саме клас, що реалізує представлення розрахунку.

```
private readonly ICalculationView _view;
private readonly CalculatorService _calculator;
private readonly WSBCContext _db;

public CalculationPresenter(ICalculationView view)
{
    _view = view ?? throw new
ArgumentNullException(nameof(view));
    _view.CalculationItemValueChanged +=
View_CalculationItemValueChanged;

    _db = new WSBCContext();
```

```
    _calculator = new CalculatorService();
}
```

Серед полів є посилання на представлення розрахунку, посилання на сервіс розрахування РВСБ і посилання на контекс БД.

У представник задекларовані наступні методи:

- public void LoadData() – завантажує дані чотирьох таблиць розрахунку за обраний рік у представлення;
- public void LoadYears() – завантажує список років, що присутні у БД;
- public void Recalculate(bool isCalculateE, bool isTakenGSValuesToAcount, bool isDistributeGSValues) – перераховує розрахунок за обраний рік, спираючись на вхідні дані розрахунку, що були збережені у БД.
- public void Remove() – видаляє розрахунок за обраний рік з БД.
- public void ExportToExcel(string filePath, WSBCParts[] chosenParts) – створює Excel-файл в якому формує та записує дані таблиць розрахунку, що були обрані. Для реалізації цього методу було встановлено бібліотеку EPPlus 5.4.2, використання якої дало змогу написати універсальний алгоритм створення Excel-файлу та заповнення даними таблиць розрахунку.
- private void View_CalculationItemValueChanged(object sender, CalculationItemEventArgs e) – обробник події зміни значення одного з компонентів таблиці розрахунку, що другим парметром приймає аргументи CalculationItemEventArgs. Об’єкт класу CalculationItemEventArgs містить всю необхідну інформацію, про те який елемент піддався змінам. Більш детальний опис класу можна перглянути на рисунку 15.

Розглянемо взаємодію з БД на прикладі методу LoadData().

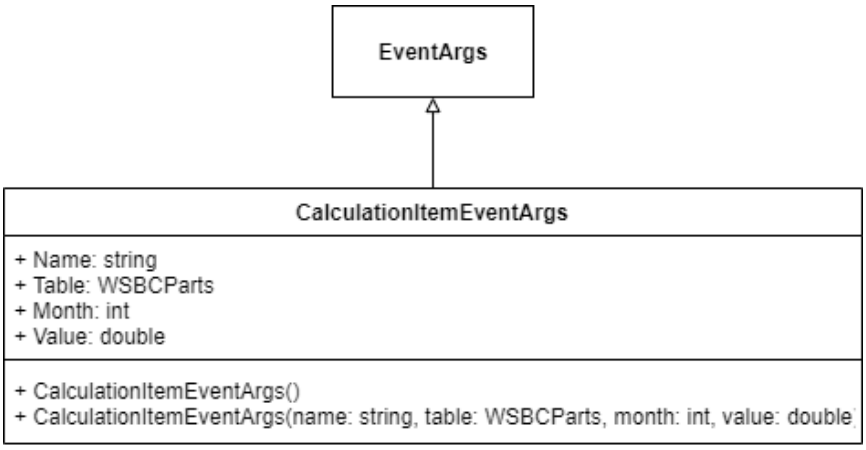


Рисунок 15 – Діаграма класу CalculationItemEventArgs

По перше використовуючи контекст БД перевіряється чи існує у сховищі даних записи про обраний рік у таблицях років та таблицях даних розрахунків, використовуючи представлення View_WaterBalanceProfitPart, так як дане представлення збирає дані з необхідних для даної операції таблиць БД.

```
if (!_db.Years.Any(x => x.Name == _view.SelectedYear) ||
!_db.View_WaterBalanceProfitPart.Any(x => x.YearName ==
_view.SelectedYear))
{
    _view.ShowMessage($"У сховищі даних немає записів розрахунку
за {_view.SelectedYear} рік.", MessageType.Error);
    return;
}
```

Якщо даних розрахунку не знайдено, тобто умова буде вірним, то користувачу буде відображено відповідне повідомлення про те, що в у сховищі даних не було знайдено записів про розрахунок за обраний рік.

Інакше у поля представлення розрахунку, що відображають джерела даних, будуть завантажені дані розрахунку за обраний рік.

```
_view.WaterBalanceProfitPartDataSource =
_db.View_WaterBalanceProfitPart.Where(x => x.YearName ==
_view.SelectedYear).ToList();
_view.WaterBalanceConsumablePartDataSource =
_db.View_WaterBalanceConsumablePart.Where(x => x.YearName ==
_view.SelectedYear).ToList();
```

```
_view.SaltBalanceProfitPartDataSource =
_db.View_SaltBalanceProfitPart.Where(x => x.YearName ==
_view.SelectedYear).ToList();
_view.SaltBalanceConsumablePartDataSource =
_db.View_SaltBalanceConsumablePart.Where(x => x.YearName ==
_view.SelectedYear).ToList();

var info = _db.CalculationInfo.FirstOrDefault(x => x.Year ==
_db.Years.First(y => y.Name == _view.SelectedYear));
string waterLevel;

switch (info.WaterLevel)
{
    case WaterLevel.Low: waterLevel = "Маловодний"; break;
    case WaterLevel.Middle: waterLevel = "Середньоводний";
break;
    default: waterLevel = "Багатоводний"; break;
}

_view.WaterLevel = $"{waterLevel}, {info.WaterLevelPercent}%";
```

Останнім кроком є виклик методу оновлення представлення.

```
_view.UpdateView();
```

3.5 Технічні рішення при створенні сервісу розрахування РВСБ

Для розрахунку РВСБ була створена бібліотека CatlabuhApp.WSBC, яка має публічний клас калькулятора. Згідно до діаграми використання представлень ПП(див. рис. 13) можна звернемо увагу на те, що операція розрахування, за деякими несуттєвими відмінностями, присутня у двох представленнях, а саме у представлені розрахунку(ICalculationView) і представлені налаштуванню вхідних даних розрахунку (ICalculationSetupView). Отже не важко прийти до думки, що операції налаштування, розрахування і збереження вихідних даних розрахунку відокремити у окремому класі.

Для таких цілей згодиться клас сервісів, що матиме назву CacluclationService і буде розміщений у пакеті CatlabuhApp.Core.Services. Розглянемо цей сервіс. В класі CacluclationService оголошено змінні калькулятора РВСБ і контекст БД, що ініціалізуються у конструкторі за замовчуванням.

```
internal CalculatorService()
{
    _db = new WSBCContext();
    _calculator = new WaterSaltBalanceCalculator();

_calculator.SetDependenceOfAvrHToF(_db.DependenceOfAvrHToF.Selec
t(x => x.AvrH),
        _db.DependenceOfAvrHToF.Select(x => x.F));


_calculator.SetGammaDistribution(_db.GammaDistribution.Select(x
=> x.Percent),
        _db.GammaDistribution.Select(x => x.Xp),
        _db.GammaDistribution.Select(x => x.Kp));

    var groundwaterInflow = _db.IntraAnnualDistribution.Where(x
=> x.IADType == IntraAnnualDistributionType.GroundwaterInflow);


_calculator.SetIADOfGroundwaterInflow(groundwaterInflow.Select(x
=> x.HighWaterLevel),
        groundwaterInflow.Select(x => x.MiddleWaterLevel),
        groundwaterInflow.Select(x => x.LowWaterLevel));

    var surfaceRunoff = _db.IntraAnnualDistribution.Where(x =>
x.IADType == IntraAnnualDistributionType.SurfaceRunoff);

    _calculator.SetIADOfSurfaceRunoff(surfaceRunoff.Select(x =>
x.HighWaterLevel),
        surfaceRunoff.Select(x => x.MiddleWaterLevel),
        surfaceRunoff.Select(x => x.LowWaterLevel));
}
```

В першу чергу створюється новий екземпляр контексту БД, після чого створюється новий екземпляр калькулятора. Наступним кроком є послідовне встановлення допоміжних даних, що беруть участь у розрахуванні, через спеціальні методи-встановлювачі. Після встановлення допоміжних даних калькулятор готовий до роботи.

Операції розрахування і збереження даних РВСБ виконуються у методі CalculateWithSaveOutputData, до якого, через вхідні параметри, передаються рік розрахунку, структури вхідних даних розрахунку і даних ДГРШ, а також три булевих прапора для налаштування перебігу операції розрахування.

```
internal void CalculateWithSaveOutputData(Year year, InputData
inputData, WSBC.Structs.GatewaySchedule gatewaySchedule, bool
isCalculateE, bool isTakenGSValuesToAcount, bool
isDistributeGSValues)
```

В першу чергу встановлюються всі допоміжні дані розрахунку, а саме масив коефіцієнтів і масиви за назвавами К1, К2, К3, участь і вплив у розрахунку яких вказано у таблиці 8.

```
_year = year;

var coefficients = _db.Coefficients.Select(x =>
x.Value).ToList();

inputData.K1 = coefficients.GetRange(19, 6);
inputData.K2 = coefficients.GetRange(25, 6);
inputData.K3 = coefficients.GetRange(31, 6);
```

Наступним кроком є встановлення публічних булевих прапорів до об’єкту калькулятора РВСБ.

```
_calculator.IsCalculateE = isCalculateE;
_calculator.IsTakenGSValuesToAcount = isTakenGSValuesToAcount;
_calculator.IsDistributeGSValues = isDistributeGSValues;
```

Після чого викликається метод розрахування РВСБ до якого передаються всі параметри розрахунку.

```
_calculator.Calculate(inputData, gatewaySchedule,
coefficients.GetRange(0, 19).ToArray());
```

Як тільки метод розрахування РВСБ завершить своє виконання, наступним кроком стане збереження і оновлення даних у БД. Якщо з терміном "збереження" питань немає, то чому треба "оновлювати" дані і де саме? Діло в тому, що при створенні розрахунки, або перерахунку, у БД вже міститься чотирнадцять записів у таблицях CalculationInputData і GatewwaySchedule. Перші дванадцять рядків таблиці заповнюються тими даним, що ввів користувач у відповідному представленні, а останні два, наче зарезервовані і заповнюються тільки при операції розрахування РВСБ. Тому саме ці два останніх рядки і оновлюються.

Для цих цілей було створено два приватних метода UpdateCalculationInputData і UpdateGatewaySchedule, що оновлюють необхідні дані.

```
UpdateCalculationInputData(_calculator.Parts.Tables[WaterSaltBal
anceCalculator.WBCPTableName], inputData.P.Sum(),
inputData.SumOfP);

UpdateGatewaySchedule(_calculator.Parts.Tables[WaterSaltBalanceC
alculator.WBPPTableName],
_calculator.Parts.Tables[WaterSaltBalanceCalculator.WBCPTableNam
e]);
```

Оновивши вхідні дані розрахунку і дані ДГРШ, перейдемо до формування списків моделей чотирьох таблиць розрахунку і збереження їх до БД.

```
var wbppModels =
GetCalculationDataModels<WaterBalanceProfitPart>(_calculator.Par
ts.Tables[WaterSaltBalanceCalculator.WBPPTableName]);
var wbcpModels =
GetCalculationDataModels<WaterBalanceConsumablePart>(_calculator
.Parts.Tables[WaterSaltBalanceCalculator.WBCPTableName]);
var sbppModels =
GetCalculationDataModels<SaltBalanceProfitPart>(_calculator.Part
s.Tables[WaterSaltBalanceCalculator.SBPPTableName]);
var sbcpModels =
GetCalculationDataModels<SaltBalanceConsumablePart>(_calculator.
Parts.Tables[WaterSaltBalanceCalculator.SBCPTableName]);
```

Метод GetCalculationDataModels є узагальненим за інтерфейсом ICalculationDataModel і приймає однин вхідний параметр типу DataTable. Це зроблено, щоб не копіювати одне й те саме чотири рази. Але обравши шлях створення узагальненого методу, що створить і заповнить даними список моделей завданого типу, встає питання: яким чином можна звернутися до однієї з моделей, якщо поля для запису розрізняються? Вирішення цієї проблеми полягає у використанні рефлексії тому, що точно відомо що назви стовпців у вихідних таблицях набору даних WaterSaltBalanceCalculator.Parts з іменами публічних полів класів моделей, що відображають таблиці вихідних даних розрахунку у БД.

У світі .NET рефлексією (reflection) називається процес виявлення типів під час виконання.

Із застосуванням служб рефлексії ті ж самі метадані, які відображає утиліта ildasm.exe, можна отримувати програмно у вигляді зручної об'єктної моделі. Наприклад, рефлексія дозволяє витягти список всіх типів, які містяться всередині певної збірки *.dll або *.exe (або навіть всередині файлу *.netmodule якщо мова йде про багато файлову збірку), в тому числі методів, полів, властивостей і подій, визначених у кожному з них. Можна також динамічно виявляти набір інтерфейсів, які підтримуються даним типом, параметрів, які приймає цей метод, і інших деталей подібного роду (таких як імена базових класів, інформація про просторах імен, дані маніфесту і т.д.)[15][1)].

```
private IEnumerable<TModel>
GetCalculationDataModels<TModel>(DataTable dataTable) where
TModel : class, ICalculationDataModel
{
    var dataRows = dataTable.AsEnumerable().ToList();
    var models = new TModel[14];

    var modelProps = typeof(TModel).GetProperties();

    for (int i = 0; i < models.Length; i++)
    {
        for (int j = 0; j < modelProps.Length; j++)
        {
            modelProps[j].SetValue(models[i],
dataRows[i].Field<double>(modelProps[j].Name));
        }

        models[i].Year = _year;
        models[i].MonthId = i;
    }

    return models;
}
```

За допомоги рефлексії у подвійному циклі перебираються моделі і їх публічні поля. Так як заздалегідь відомо, що назви полів і стовпців збігаються, а також збігається їх порядок, то не повинно виникнути жадних проблем у встановлені значень з таблиць.

[1)] [15] Сборки .NET | Рефлексия. URL: https://professorweb.ru/my/csharp/assembly/level2/2_3.php. (дата звернення 08.10.2020).

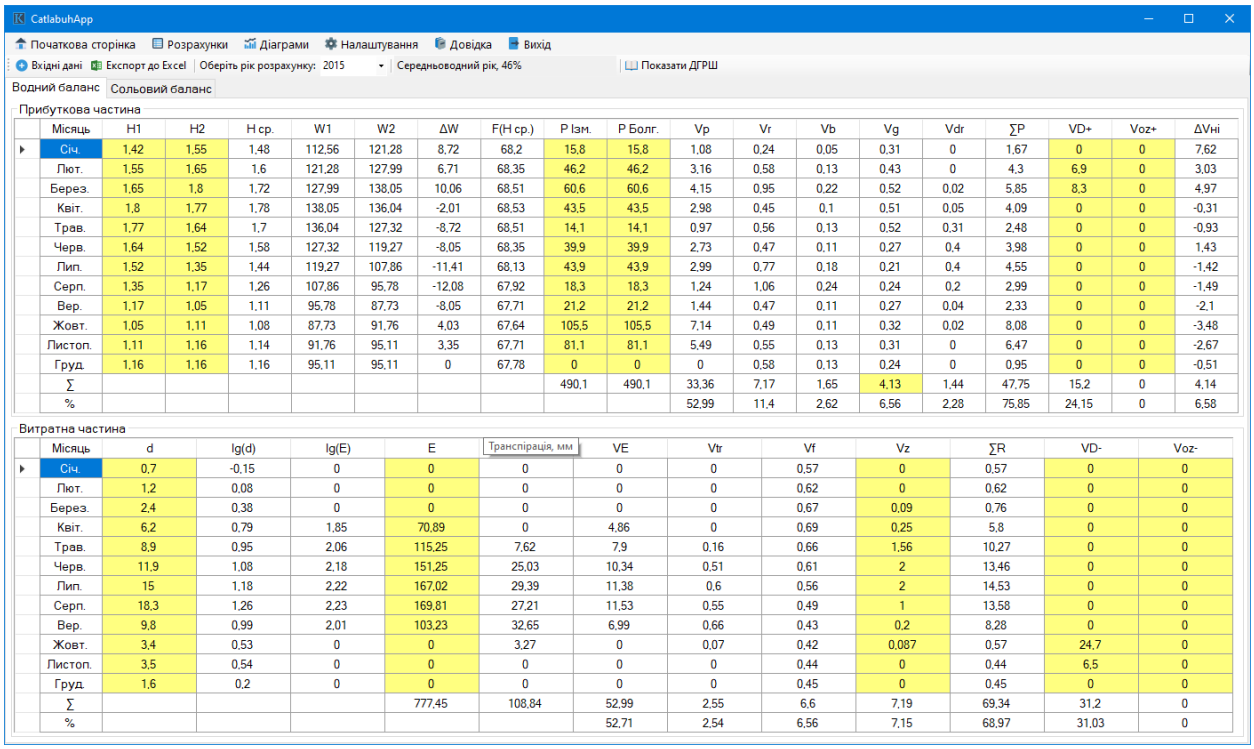
Після формування списків моделей вихідних даних розрахунку, залишається лише додати їх до відповідних таблиць контексту БД і викликати метод збереження змін, який перенесе всі зміни до БД.

```
_db.WaterBalanceProfitParts.AddRange(wbppModels);
_db.WaterBalanceConsumableParts.AddRange(wbcpModels);
_db.SaltBalanceProfitParts.AddRange(sbppModels);
_db.SaltBalanceConsumableParts.AddRange(sbcpModels);
_db.SaveChangesAsync();
```

4 ОПИС РОБОТИ ПРОГРАМНОЇ СИСТЕМИ

При запуску ПП користувача зустрічає головне вікно програми. Через це вікно здійснюється доступ до всього функціоналу ПП. Скориставшись головним меню, можна перемикатися між сторінками, що реалізують інтерфейси представлень, які зображені на діаграмі варіантів використання представлень ПП(див. рис. 13).

На зображені нижче(див. рис. 16) відображено сторінку перегляду розрахунку.



Місяць	H1	H2	H ср.	W1	W2	ΔW	F(H ср.)	P Ізм.	P Болг.	Vp	Vr	Vb	Vg	Vdr	∑P	VD+	Voz+	ΔVні
Січ.	1,42	1,55	1,48	112,56	121,28	8,72	68,2	15,8	15,8	1,08	0,24	0,05	0,31	0	1,67	0	0	7,62
Лют.	1,55	1,65	1,6	121,28	127,99	6,71	68,35	46,2	46,2	3,16	0,58	0,13	0,43	0	4,3	6,9	0	3,03
Берез.	1,65	1,8	1,72	127,99	138,05	10,06	68,51	60,6	60,6	4,15	0,95	0,22	0,52	0,02	5,85	8,3	0	4,97
Квіт.	1,8	1,77	1,78	138,05	136,04	-2,01	68,53	43,5	43,5	2,98	0,45	0,1	0,51	0,05	4,09	0	0	-0,31
Трав.	1,77	1,64	1,7	136,04	127,32	-8,72	68,51	14,1	14,1	0,97	0,56	0,13	0,52	0,31	2,48	0	0	-0,93
Черв.	1,64	1,52	1,58	127,32	119,27	-8,05	68,35	39,9	39,9	2,73	0,47	0,11	0,27	0,4	3,98	0	0	1,43
Лип.	1,52	1,35	1,44	119,27	107,86	-11,41	68,13	43,9	43,9	2,99	0,77	0,18	0,21	0,4	4,55	0	0	-1,42
Серп.	1,35	1,17	1,26	107,86	95,78	-12,08	67,92	18,3	18,3	1,24	1,06	0,24	0,24	0,2	2,99	0	0	-1,49
Вер.	1,17	1,05	1,11	95,78	87,73	-8,05	67,71	21,2	21,2	1,44	0,47	0,11	0,27	0,04	2,33	0	0	-2,1
Жовт.	1,05	1,11	1,08	87,73	91,76	4,03	67,64	105,5	105,5	7,14	0,49	0,11	0,32	0,02	8,08	0	0	-3,48
Листоп.	1,11	1,16	1,14	91,76	95,11	3,35	67,71	81,1	81,1	5,49	0,55	0,13	0,31	0	6,47	0	0	-2,67
Груд.	1,16	1,16	1,16	95,11	95,11	0	67,78	0	0	0	0,58	0,13	0,24	0	0,95	0	0	-0,51
∑								490,1	490,1	33,36	7,17	1,65	4,13	1,44	47,75	15,2	0	4,14
%										52,99	11,4	2,62	6,56	2,28	75,85	24,15	0	6,58

Місяць	d	lg(d)	lg(E)	E	Транспірація, мм	VE	Vtr	Vf	Vz	∑R	VD-	Voz-
Січ.	0,7	-0,15	0	0	0	0	0	0,57	0	0,57	0	0
Лют.	1,2	0,08	0	0	0	0	0	0,62	0	0,62	0	0
Берез.	2,4	0,38	0	0	0	0	0	0,67	0,09	0,76	0	0
Квіт.	6,2	0,79	1,85	70,89	0	4,86	0	0,69	0,25	5,8	0	0
Трав.	8,9	0,95	2,06	115,25	7,62	7,9	0,16	0,66	1,56	10,27	0	0
Черв.	11,9	1,08	2,18	151,25	25,03	10,34	0,51	0,61	2	13,46	0	0
Лип.	15	1,18	2,22	167,02	29,39	11,38	0,6	0,56	2	14,53	0	0
Серп.	18,3	1,26	2,23	169,81	27,21	11,53	0,55	0,49	1	13,58	0	0
Вер.	9,8	0,99	2,01	103,23	32,65	6,99	0,66	0,43	0,2	8,28	0	0
Жовт.	3,4	0,53	0	0	3,27	0	0,07	0,42	0,087	0,57	24,7	0
Листоп.	3,5	0,54	0	0	0	0	0	0,44	0	0,44	6,5	0
Груд.	1,6	0,2	0	0	0	0	0	0,45	0	0,45	0	0
∑				777,45	108,84	52,99	2,55	6,6	7,19	69,34	31,2	0
%						52,71	2,54	6,56	7,15	68,97	31,03	0

Рисунок 16 – Сторінка перегляду розрахунку

Для того щоб вести вхідні дані і виконати операцію розрахування РВСБ, використовується вікно "Налаштування вхідних даних", переглянути яке можна на рисунку 17.

А для того, щоб налаштувати дані ДГРШ, було створено відповідне вікно програми. Його зображення присутнє на рисунку 18.

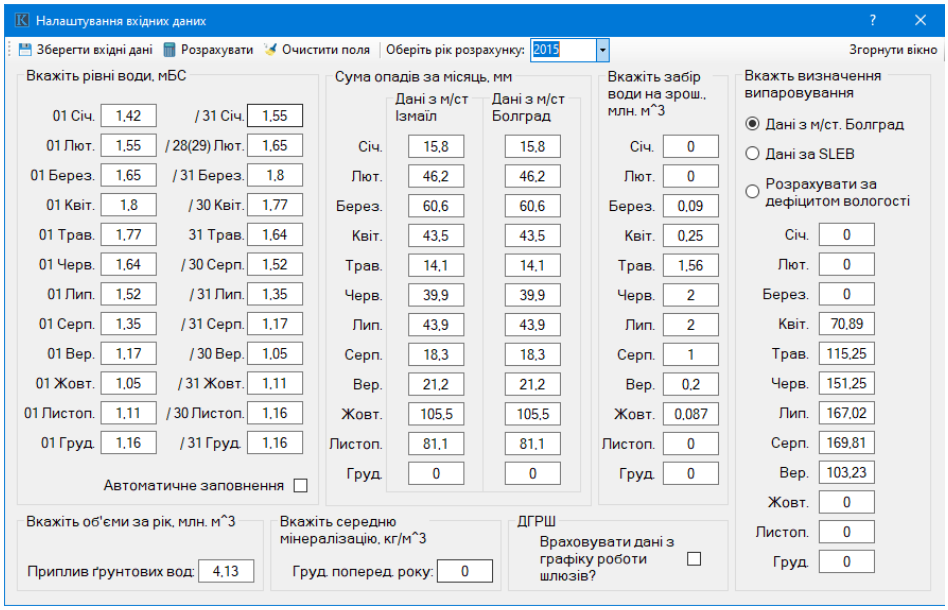


Рисунок 17 – Вікно налаштування вхідних даних

Рисунок 18 – Вікно налаштування даних ДГРШ

З цих двох вікон виконати, незалежно, операцію збереження даних, що були введені користувачем у поля форми.

Запуск процесу розрахування РВСБ здійснюється з вікна налаштувань вхідних даних, при натиску відповідної кнопки. А для більш тонкого налаштування вхідних даних, тобто встановлення допоміжних даних, що беруть участь у розрахунку, використовується сторінка налаштувань програми. Переглянути яку можна у головному вікні програми(див. рис. 19).

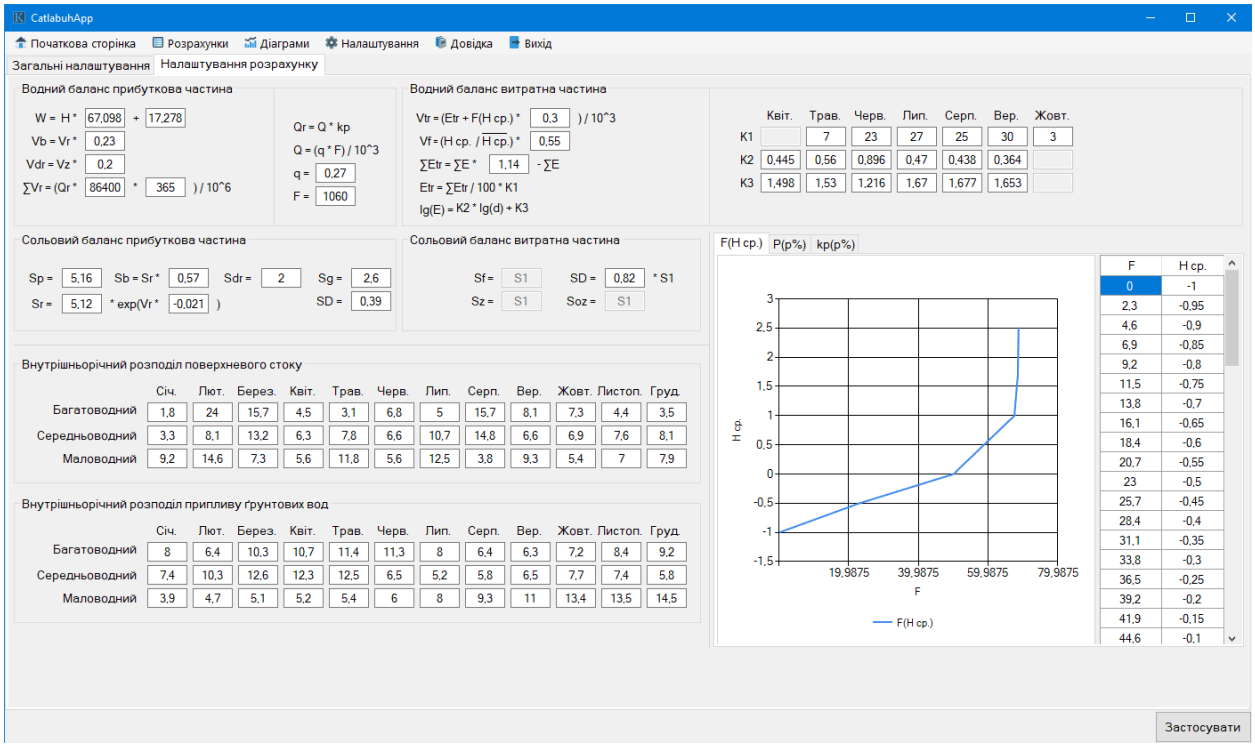


Рисунок 19 – Сторінка налаштувань програми

Для візуалізації даних РВСБ, користувачу у програмну систему було додано сторінку відображення та налаштування діаграми, що дозволяє будувати декілька різновидів діаграм(див. рис. 20). Налаштування компонентів діаграми і її зовнішнього вигляду здійснюється за допомоги відповідних форм.

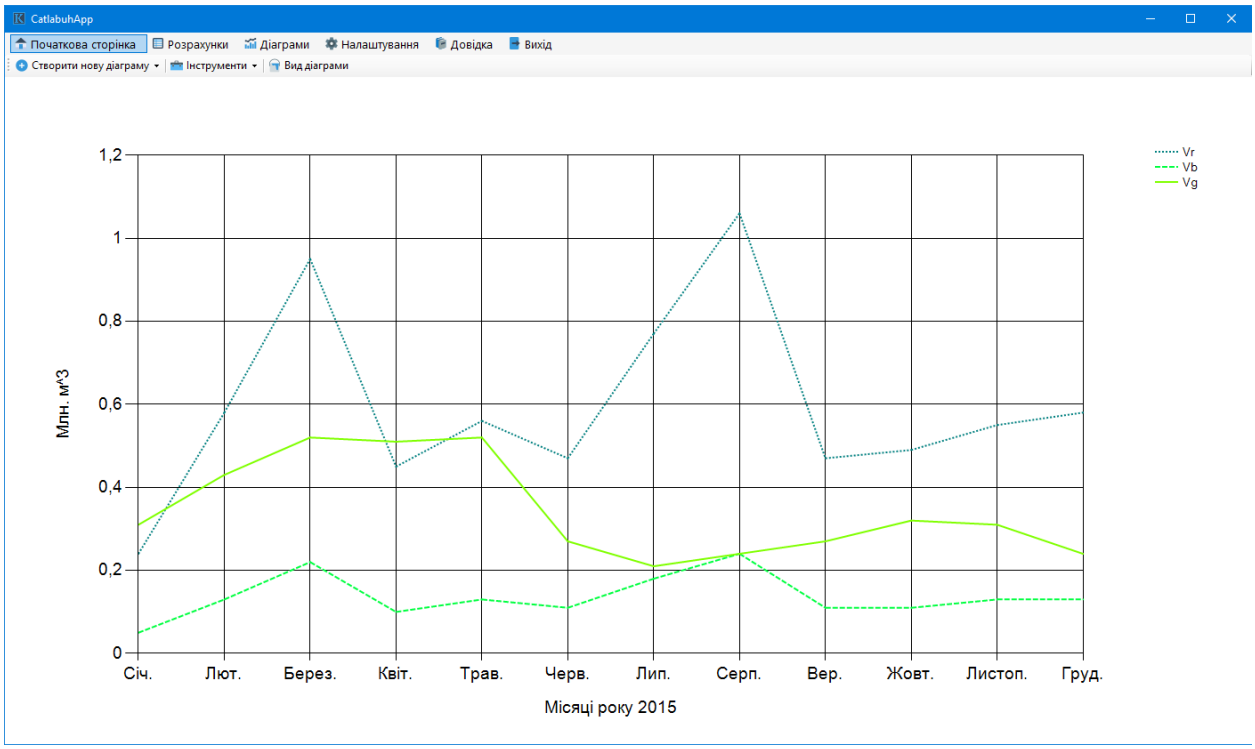


Рисунок 20 - Сторінка перегляду і налаштування діаграм

ВИСНОВКИ

При виконанні проекту дипломної роботи було оглянуто та проаналізовано:

- розрахунок водного і сольового балансів озера Катлабух;
- попередня система збереження результатів розрахунків;
- платформи розробки ПП;
- системи збереження даних розрахунків.

Розроблені:

- бібліотека розрахування РВСБ;
- БД для збереження результатів розрахунків;
- розподілений на незалежні модулі ПП.

Розроблений програмний комплекс дозволяє швидко, зручно та точно розрахувати водно-сольовий режим озера Катлабух, графічно представити одержані результати, змоделювати у часі водний баланс водойми за різних умов його водогосподарської експлуатації з метою надання обґрунтованих техніко-економічних рекомендацій стосовно приведення озера до доброго гідроекологічного стану.

Розроблений програмний продукт в повній мірі відповідає всім вимогам, що були пред’явлені у ТЗ(див. п.1.2).

ПП має велику схильність до розширення функціоналу завдяки обраній структурі, яка дозволяє зібрати у один продукт можливості різного ПЗ, що наслідують схожу із розробленим забезпеченням структуру.

Подальші перспективи розвитку комплексу пов’язані з оптимізацією і прискоренням обчислень розрахунку, просторовим по акваторії озера моделюванням розподілу мінералізації та якості води.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Романова Є.О., Шакірзанова Ж.Р., Медведєва Ю.С. Водний та сольовий баланси озера Катлабух за різних умов експлуатації водойми.
2. Выбор платформы для приложения. URL: https://docs.microsoft.com/ru-ru/windows/apps/desktop/choose-your-platform. (дата звернення 03.09.2020).
3. Microsoft Visual Studio. URL: https://ru.wikipedia.org/wiki/Microsoft_Visual _Studio. (дата звернення 03.09.2020).
4. About SQLite. URL: https://www.sqlite.org/about.html. (дата звернення 03.09.2020).
5. MySQL. URL: https://uk.wikipedia.org/wiki/MySQL. (дата звернення 05.09.2020).
6. XML. URL: https://uk.wikipedia.org/wiki/XML(дата звернення 05.09.2020).
7. Введение в JSON. URL: https://www.json.org/json-ru.html(дата звернення 05.09.2020).
8. Git за полчаса: руководство для начинающих. URL: https://proglib.io/p/git-for-half-an-hour. (дата звернення 05.09.2020).
9. Модель-вид-контролер – Вікіпедія. URL: https://uk.wikipedia.org/wiki/Модель-вид-контролер. (дата звернення 05.09.2020).
10. Model-View-Presenter – Вікіпедія. URL: https://uk.wikipedia.org/wiki/Model-View-Presenter. (дата звернення 05.09.2020).
11. Model-View-ViewModel – Вікіпедія. URL: https://uk.wikipedia.org/wiki/Model-View-ViewModel. (дата звернення 05.09.2020).

12. Библиотеки классов .NET | Microsoft Docs. URL: https://docs.microsoft.com/ru-ru/dotnet/standard/class-libraries. (дата звернення 08.10.2020).
13. .NET Standard | Microsoft Docs. URL: https://docs.microsoft.com/ru-ru/dotnet/standard/net-standard. (дата звернення 08.10.2020).
14. Введение в Entity Framework 6. URL: https://metanit.com/sharp/entityframework/1.1.php. (дата звернення 08.10.2020).
15. Сборки .NET | Рефлексия. URL: https://professorweb.ru/my/csharp/assembly/level2/2_3.php. (дата звернення 08.10.2020)
16. C# docs - get started, tutorials, reference. | Microsoft Docs. URL: https://docs.microsoft.com/en-us/dotnet/csharp/. (дата звернення 08.10.2020)
17. Язык программирования C# и платформа .NET. URL: https://metanit.com/sharp/. (дата звернення 08.10.2020)
18. Overview – Microsoft.Data.Sqlite | Microsoft Docs. URL: https://docs.microsoft.com/en-us/dotnet/standard/data/sqlite/?tabs=netcore-cli. (дата звернення 08.10.2020)
19..NET Framework – Википедия. URL: https://ru.wikipedia.org/wiki/.NET_Framework. (дата звернення 08.10.2020)
20. Нотация UML | Глоссарий ПитерСофт. URL: https://piter-soft.ru/knowledge/glossary/process/notatsiya-UML.htm(дата звернення 08.10.2020)
21. Ліхачов К. Д. Розробка програмного комплексу для розрахунку водного і сольового балансів озера Катлабух. Матеріали XIX наукової конференції молодих вчених ОДЕКУ. 25-29 травня 2020 р. Одеса: ОДЕКУ. 2020. С.216-218.