МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОДЕСЬКОГО НАЦІОНАЛЬНОГО ТЕХНОЛОГІЧНОГО УНІВЕРСИТЕТУ»

Спеціальність: 121 «Інженерія програмного забезпечення»

Освітня програма: «Розробка програмного забезпечення»

Група: 4РП-05

Дипломний проект

здобувача освіти денної форми навчання
РП.05.01.000.ДП

ВЄЛКОВ
ІВАН
ВАСИЛЬОВИЧ

м. Одеса
2022 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОДЕСЬКОГО НАЦІОНАЛЬНОГО ТЕХНОЛОГІЧНОГО УНІВЕРСИТЕТУ»

Спеціальність: **121 «Інженерія програмного забезпечення»**
Освітня програма: «**Розробка програмного забезпечення»**
Група: **4РП-05**

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломного проекту (роботи) на тему:

Побудова веб-орієнтованої системи з використанням алгоритмів оптимізації запитів до бази даних з використанням мови C# і платформи ASP.NET Core MVC

Проектний матеріал складається з пояснювальної записки на 46 сторінках та графічного (презентаційного) матеріалу на 10 аркушах (слайдах).

Дипломник _____ (Вєлков І.В.)
Керівник _____ (Медведєв А.О.)

Консультанти:
з економічної частини _____ (Копайгородська Т.Г.)
з охорони праці_____ (Чорновол Н.І.)
з дотримання вимог ЄСКД _____ (Петрашова В.І.)
старший консультант _____ (Скорнякова О.В.)

До захисту допущений
Голова циклової комісії _____ (Скорнякова О.В.)
Завідувач відділення _____ (Суліма Ю.Ю.)

Захист «____»_____ 2022 р. Протокол ДКК №_____
Оцінка ДКК _____

Секретар ДКК _____

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОДЕСЬКОГО НАЦІОНАЛЬНОГО ТЕХНОЛОГІЧНОГО УНІВЕРСИТЕТУ»

Відділення **комп'ютерних систем** Комісія **КТ та ПІ**

Спеціальність **121 «Інженерія програмного забезпечення»**

Освітня програма **«Розробка програмного забезпечення»**

ЗАТВЕРДЖУЮ:

Заст. дир. з НВР_____

_____

"_____"_____2022 р.

З А В Д А Н Н Я

на дипломний проект (роботу)

Здобувачеві (здобувачці) освіти Вєлков Іван Васильович

(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) Побудова веб-орієнтованої системи з використанням алгоритмів оптимізації запитів до бази даних з використанням мови C# і платформи ASP.NET Core

затверджена наказом по коледжу від "30" грудня 2021 р. № 306-А2-ОД

2. Термін здачі закінченого проекту (роботи) _____

3. Вихідні данні до проекту (роботи) Flutter додаток з елементамі REST-API запитів

4. Зміст розрахунково-пояснювальної записки (перелік питань, які необхідно розробити)

1. Аналітичний розділ дипломного проекту. Розробка Flutter додатка. 2. Економічний розрахунок. 3. Охорона праці.

5. Перелік графічного (презентаційного) матеріалу (з точним зазначенням обов'язкових креслень, кількості слайдів)

Презентація (10 слайдів)

6. Консультанти по проекту (роботі), із зазначенням розділів проекту, що їх стосується

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Основний	Медведєв А.О.		
Економічний	Копайгородська Т.Г.		
Охорона праці	Чорновол Н.І.		
Нормоконтроль	Петрашова В.І.		
Старший консультант	Скорнякова О.В.		

7. Дата видачі завдання _____

Керівник _____

(підпис)

Завдання прийняв до виконання _____

(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/р	Назва етапів дипломного проекту (роботи)	Термін виконання етапів дипломного проекту (роботи)	Відмітка про виконання
1	Розділ 1. Аналітичний розділ дипломного проекту.	17.05.2022	Вик.
2	Розділ 2. Економічний розділ	20.05.2022	Вик.
3	Розділ 3. Охорона праці	23.05.2022	Вик.
4	Розробка презентації до дипломної роботи	1.06.2022	Вик.
5	Чистове оформлення пояснювальної записки	3.06.2022	Вик.
6	Підготовка доповіді до захисту	5.06.2022	Вик.
7	Отримання рецензії, відповіді на зауваження рецензента	5.06.2022	Вик.
8	Захист роботи	16.06.2022	Вик.

Дипломник _____

(підпис)

Керівник _____

(підпис)

Формат	Зона	Поз.	Позначення	Назва	Кіл.	Примітка
				Документація		
			РП 05.01.000 ДП ПЗ	Дипломний проект		
А4			РП 05.01.000 ДП ПЗ	Пояснювальна	1	
				записка		
				Презентація		

					РП 05.01.000	
		№ докум.				
Розробив		Вєдков І.В.			Побудова веб-орієнтованої системи з використанням алгоритмів оптимізації запитів до бази даних з використанням мови C# і платформи ASP.NET Core MVC	ВСП«ОТФК ОНТУ» гр.РП-05
Перевірив		Медведєв А.О				
Н. Контр.		Петрашова В.І				
Затверд.		Скорнякова О.В				

АНОТАЦІЯ

Об'єкт дослідження: об'єктом дослідження є оптимізація запитів до бази даних

Предмет дослідження: предметом дослідження стала розробка веб система у вигляді Flutter додатка.

Мета роботи: Створення алгоритмів для оптимізації запитів до базі даних.

Досягнуті результати:

- Вибрані необхідні засоби розробки;
- Розроблена архітектура взаємодії з мобільною складовою
- Розроблена архітектура пакетів системи;
- Створені необхідні таблиці в базі даних для перевірки алгоритмів оптимізації;
- Виявлено проблеми непрацюючого і незатребуваного функціоналу і враховані при створенні алгоритмів оптимізації запитів;

Ключові слова: .NET, Flutter, SQL, TCP.

Обсяг: 46 стор., 24 рисунки, 7 табл. , 8 джерел.

ЗМІСТ

ВСТУП

У сучасному світі головною цінністю є інформація. Вона всюди оточує нас. Вона міститься на телефонах, комп'ютерах і навіть в світлофорах, але найголовніше що вона міститься в базах даних.

База даних-це упорядкований набір структурованої інформації або даних, які зазвичай зберігаються в електронному вигляді в комп'ютерній системі. База даних зазвичай управляється системою управління базами даних. Дані разом з СУБД, а також додатки, які з ними пов'язані, називаються системою баз даних, або, для стислості, просто базою даних.

Інформація, яка надається користувачеві, проходить шлях від його пристрою через сервер до бази даних.

Тобто як мінімум 2 запити передачі даних. Будь-який запит займає певний час, воно може залежати від швидкості передачі даних, обсягу масиву інформації, способу отримання інформації. Для того щоб прискорити дану процедуру існують деякі алгоритми оптимізації запиту даних, як між клієнтом і сервером, так і між сервером і базами даних.

У моїй дипломній роботі будуть розглянуті обидва шляхи передачі інформації.

Мета проекту – дослідити роботу алгоритмів оптімізації запитів до базі данніх з серверу, та з клієнта до сервера. Створити мобільний додаток для тестування працездатності алгоритмів.

Для виконання завдання мені необхідно виконати наступні цілі:

1. Розібрати архітектуру запиту
2. Створити базу даних
3. Створити серверне рішення
4. Прив'язати серверну частину мобільним додатком

1 АНАЛІЗ ПРОБЛЕМИ ОПТИМІЗАЦІЇ ЗАПИТІВ ДО БАЗИ НА ПРИКЛАДІ ПОБУДОВИ ВЕБ-СИСТЕМИ ASP.NET

1.1 Введення в технологію побудови веб-додатків

Web-додаток – програма з певним набором функціоналу, що використовує в якості клієнта браузер. Іншими словами, якщо додатком для здійснення бізнес-логіки потрібно мережеве з'єднання і наявність на стороні браузера користувача, то його відносять до веб-додатком.

До сьогоднішнього дня історично склалося 3 типи програм:

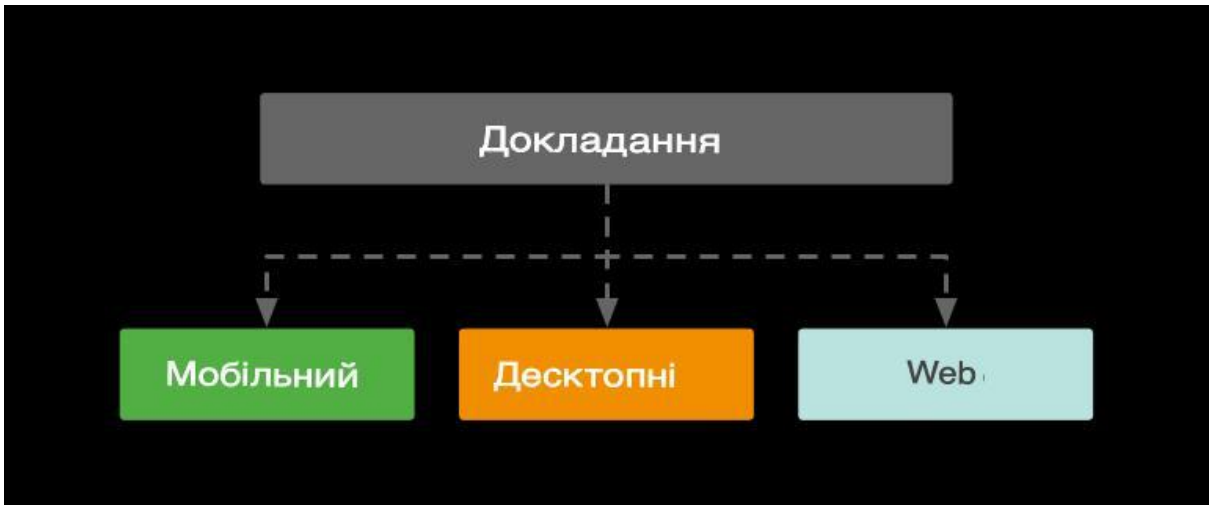

Рисунок 1.1 - Типи докладань

Десктопні програми припускають установку клієнта на стороні користувача. В залежності від типу операційної системи, процесора, відеокарти і інших параметрів можуть знадобитися різні версії програми. Це створює певні незручності як розробникам (їм потрібно постійно вишукувати баги в різних середовищах, розширювати обсяг коду для обліку всіх можливих комбінацій заліза клієнта), так і користувачам (необхідно скачування постійних оновлень, нове залізо з тією операційною системою, яку підтримує додаток).

Мобільні додатки заточені винятково для смартфонів і планшетів з урахуванням встановленої там системи (Android, iOS та ін). Це також додає

складності розробників софту. Важливо відзначити, що багато мобільні додатки фактично є web-додатками (про що не підозрюють всі користувачі), так як можливості «движків» браузерів це дозволяють.

Найбільш динамічно на сьогодні розвиваються web-додатки, так як вони для своєї роботи вимагають тільки встановлений браузер на клієнтській стороні. Вони:

- Можуть працювати як на смартфоні, так і персональному комп'ютері,
- Практично незалежні від заліза
- По функціоналу скоро перестануть поступатися десктопним аналогам.

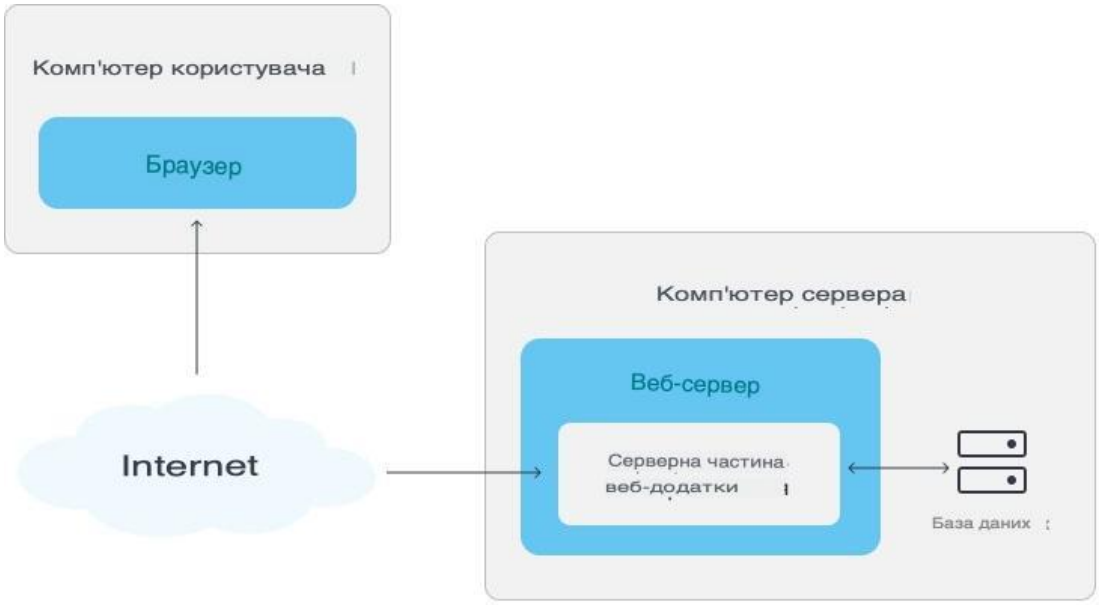

Рисунок 1.2 - Схема зв`язку

Для виконання закладеної логіки веб-додатком потрібен доступ до мережі. Саме тому кожному програмісту слід розуміти структуру Інтернету, способи передачі даних між пристроями.

Щоб повідомлення від вашого комп'ютера дійшло до користувача на іншому кінці планети, використовується складна структура передачі мережевих даних на декількох рівнях.

Розглянемо найбільш просту модель мережевих шарів, що складається з п'яти рівнів. В літературі її ще називають моделлю **TCP/IP**.

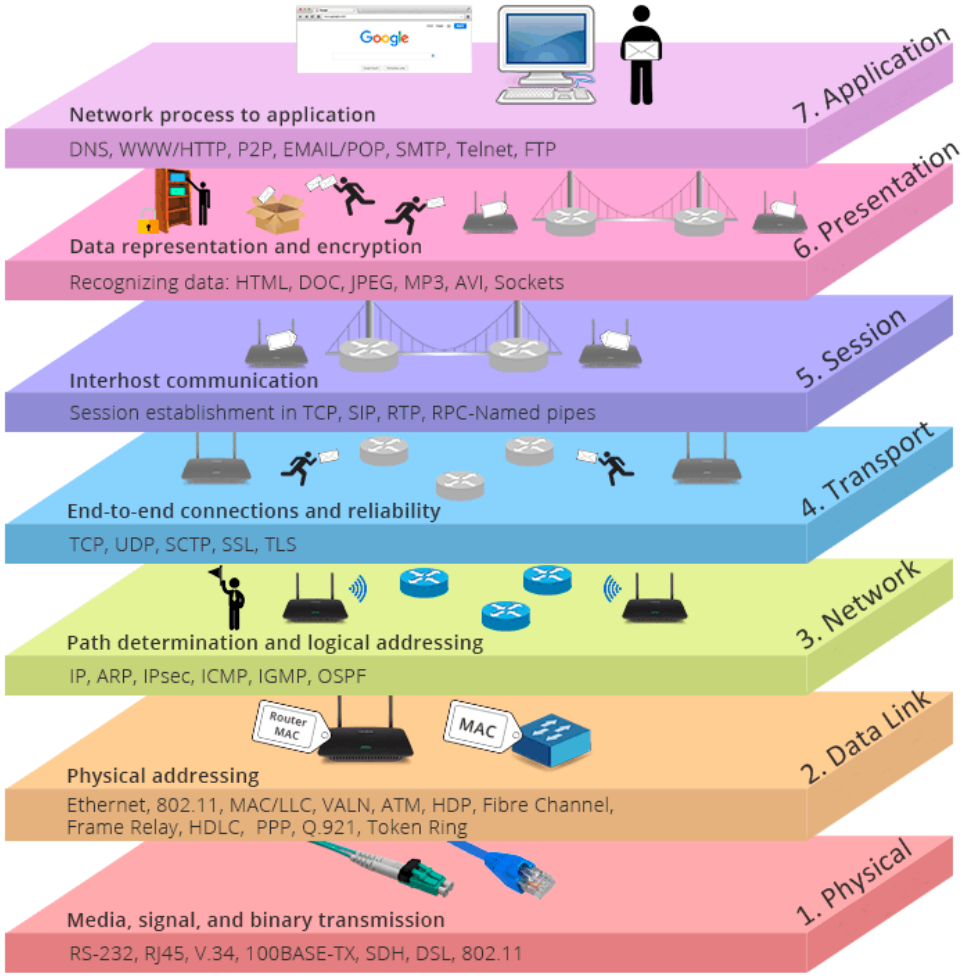

Рисунок 1.3 - Модель OSI

Фізичний рівень

Фізичний рівень є найнижчим. Він пов'язаний безпосередньо з законами фізики. Наприклад, через оптоволоконний кабель передаються пучки світла, які інтерпретуються машиною як нулі або одиниці. Будь-яка ваша дія в Інтернеті в кінці кінців стає набором сигналів, які через кабель або радіохвилі транслюються між пристроями.

Канальний рівень

Канальний рівень відповідає за з'єднання пристроїв одній мережі (локальної). Тут використовується два популярних протоколу:

- Ethernet (коли комп'ютери з'єднуються кабелем)
- WiFi (бездротове з'єднання).

Щоб різні пристрої могли різнитися, вони мають MAC-адреси, які являють собою унікальні ідентифікатори. У світі практично немає двох гаджетів з однаковими MAC-адресами. Приклад MAC-адреси: 12:FA:A9:11:BC:DE (шість 16-ричных однобайтных чисел, розділених двокрапкою).

Мережевий рівень

Мережний рівень – необхідність функціонування Інтернету, так як дає можливість спілкування пристроїв з різних мереж. Використання IP-протоколу дозволяє присвоювати мережевим пристроям певні адреси і визначати їх приналежність до конкретної мережі. IP-адреса складається з чотирьох однобайтных чисел, які записуються в десятковій системі. Типовий приклад: 192.16.0.1.

На підставі адрес пристроїв мережі ми можемо підключатися до сайтів, серверів, поштових клієнтів і отримувати необхідну інформацію. Існує 2 протоколу IP: 4-ої і 6-ої версії. У найближчому майбутньому відбудеться повний перехід на останню версію, так як версія IPv4 вже не справляється із зростаючою кількістю комп'ютерів і гаджетів з доступом до мережі (більш того, у 2020 році в світі закінчилися вільні IP-адреси).

Транспортний рівень

Транспортний рівень дозволяє ідентифікувати адресатів і не змішувати потоки даних. На будь-якому ПК або мобільному пристрої може одночасно працювати кілька додатків, яким потрібно мережеве підключення. Погодьтеся, email і MMORPG ніколи не міняються місцями при отриманні повідомлень. Ви завжди впевнені в тому, що на пошту приходить повідомлення, а в грі інформація характеризується іншими параметрами.

Щоб трафіки не змішувалися, кожна програма використовує для мережевого спілкування певний порт. Наприклад, браузер для відкриття сайту використовує порт 80, а для з'єднання з базою даних може відкриватися порт 8080.

Найбільш популярні на транспортному рівні 2 протоколи: TCP і UDP.

TCP гарантує отримання всієї переданої інформації (інакше виникне помилка), а UDP такої гарантії не дає. Так, коли ви завантажуєте книгу з сайту, застосовується протокол TCP (хіба комусь потрібна книга шматками?), а коли дивитеся стрім на ютубі, то досить протоколу UDP (якщо загубиться кілька кадрів, ви цього не помітите, зате якість трансляції буде високою).

Рівень додатків

Рівень додатків – це, по факту, то, з чим безпосередньо працює програміст більшу частину часу. Головне завдання даного шару – надати клієнту зручний інтерфейс для взаємодії з мережею і пристроями.

Тут використовується величезна кількість різноманітних протоколів, серед яких:

1. HTTP (застосовується браузером для отримання даних з Інтернету),
2. SMTP (для поштових сервісів),
3. FTP (завантаження файлів),
4. BitTorrent (пиринговый протокол),
5. SSH (захищене з'єднання).

1.2 Платформа ASP.NET і її роль

ASP.NET — технологія створення веб-додатків і веб-сервісів від компанії Microsoft. Вона є складовою частиною платформи Microsoft .NET і розвитком більш старої технології Microsoft ASP.

Після випуску сервера Internet Information Services 4.0 в 1997 році, компанія Microsoft почала досліджувати можливість нової моделі веб-додатки, яка задовольнить скарги на ASP, особливо пов'язані з відділенням оформлення від змісту, і яка дозволить писати «чистий» код. Робота по розробці такої моделі була доручена Марку Андерсу, менеджеру команди IIS, і Скотту Гатрі, який поступив на роботу в Microsoft у 1997. Андерс і Гатрі розробили

початковий проект протягом двох місяців, і Гатрі написав код початкового прототипу під час різдвяних канікул 1997 року.

Початковий проект називався «XSP»; Гатрі пояснив в інтерв'ю 2007 року, «завжди запитують, що означає буква X. В той час вона нічого не значила. XML починається з неї; XSLT починається з неї. Все кльове починається з X, тому ми його так і назвали.» Прототип XSP був написаний на Java, так як на той момент в Microsoft не було Java-подібної технології. В той час вже передбачалося (як далі з'ясувалося грунтовно), що ліцензування Java для Microsoft не буде продовжено в 2003 році (у 2003 закінчувався термін виданої Sun Microsystems ліцензії). У 1999 було вирішено побудувати нову платформу на основі Common Language Runtime (CLR), так як в ньому як і в Java висіло програмування за принципами ООП, Збірка сміття та інші можливості. Скотту Гатрі описав це рішення як «величезний ризик», так як успіх нової розробки був пов'язаний з успіхом CLR, яка, як і XSP, перебувала на ранній стадії розробки.

Розібравшись з історією ASP.NET стає зрозуміло, що це один з найбільш передових і зручних інструментів для створення REST API.

1.3 Основні відомості до побудови запиту до бази даних

Перед побудовою архітектури бази даних треба розібратися з її типом. Існують наступні види:

- **Ієрархічна база даних** – кожен об'єкт при такому зберігання інформації представляється у вигляді певної сутності, тобто, у цієї сутності можуть бути дочірні елементи, батьківські елементи, а у тих дочірніх можуть бути ще дочірні елементи, але є один об'єкт, з якого все починається. Виходить своєрідне дерево. Прикладом ієрархічної бази даних може бути, документ у форматі XML або файлова система комп'ютера.

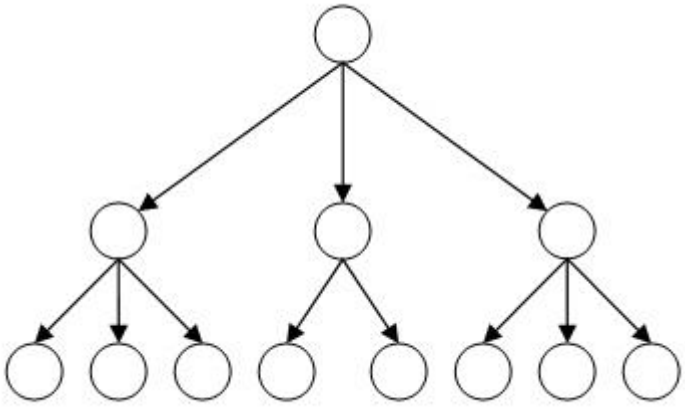

Рисунок 1.4 - План ієрархічной бази даних

- **Мережеві бази даних**, є своєрідною модифікацією ієрархічних баз даних. У ієрархічних баз даних у кожного дочірнього елемента може бути тільки один нащадок. Мережеві бази даних відрізняються від ієрархічних тим, що у дочірнього елемента може бути кілька предків, тобто елементів, що стоять вище нього. Для більшої наочності і розуміння структури мережевих баз даних зверніть увагу на малюнок:

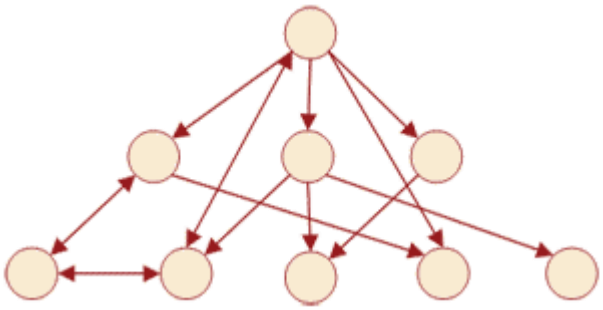

Рисунок 1.5 - План мережевой бази даних

- **Реляційні бази даних**, зберігають дані у вигляді набору двовимірних таблиць. Тобто, таблиця складається з набору стовпців, в якому може зазначатися: назва, тип даних(дата, число, рядок, текст тощо). Ще однією важливою особливість реляційних БД є те, що число стовпців фіксовано, тобто, структура бази даних відома заздалегідь, а ось число рядків або рядів у

реляційних базах даних нічим не обмежена, якщо говорити грубо, то рядки в реляційних базах даних є об'єкти, які зберігаються в базі даних.

Вибравши потрібний тип бази даних можна переходити до побудови запиту. Побудова запиту будується на мові SQL і являє собою цілу дисципліну, але в загальних рисах все можна звести до 6 командам.

- **SELECT, FROM** — обов'язкові елементи запиту, які визначають вибрані стовпці, їх порядок та джерело даних.

```
SELECT * FROM Customers
```

Рисунок 1.6 - Пример запиту Select

- **WHERE** — необов'язковий елемент запиту, який використовується, коли потрібно відфільтрувати дані за потрібною умовою. Дуже часто всередині елемента where використовуються IN / NOT IN для фільтрації стовпця по декількох значеннях, AND / OR для фільтрації таблиці за кількома стовпцями.

```
select * from Customers
where City IN ('London', 'Berlin')
```

Рисунок 1.7 - Пример запиту Where

- **GROUP BY** — необов'язковий елемент запиту, за допомогою якого можна задати агрегацію за потрібного стовпця (наприклад, якщо потрібно довідатися яка кількість клієнтів живе в кожному з міст).

```
select City, count(CustomerID) from Customers
GROUP BY City
```

Рисунок 1.8 - Пример запиту Group By

- **HAVING** — необов'язковий елемент запиту, який відповідає за фільтрацію на рівні згрупованих даних (по суті, WHERE, але тільки на рівень вище).

```
select City, count(CustomerID) from Customers
group by City
HAVING count(CustomerID) >= 5
```

Рисунок 1.9 - Пример запиту Having

- **ORDER BY** — необов'язковий елемент запиту, який відповідає за сортування таблиці.

```
select * from Customers
ORDER BY Country, City
```

Рисунок 1.10 - Пример запиту Order by

- **JOIN** — необов'язковий елемент, використовується для об'єднання таблиць по ключу, який присутній в обох таблицях. Перед ключем ставиться оператор ON.

```
select * from Orders
JOIN Customers ON Orders.CustomerID = Customers.CustomerID
```

Рисунок 1.11 - Пример запиту Join

Зрозумівши основні команди мови SQL можна перейти до підключення бази даних до проекту. Для цього нам знадобиться пакет Entity Framework.

Entity Framework являє ORM-технологію (object-relational mapping - відображення даних на реальні об'єкти) від компанії Microsoft для доступу до даних. Entity Framework Core дозволяє абстрагуватися від самої бази даних і таблиць та працювати з даними об'єктами класом незалежно від типу сховища. Якщо на фізичному рівні ми оперуємо таблицями, індексів, первинними і зовнішніми ключами, але на концептуальному рівні, який нам пропонує Entity Framework, ми вже працюємо з об'єктами.

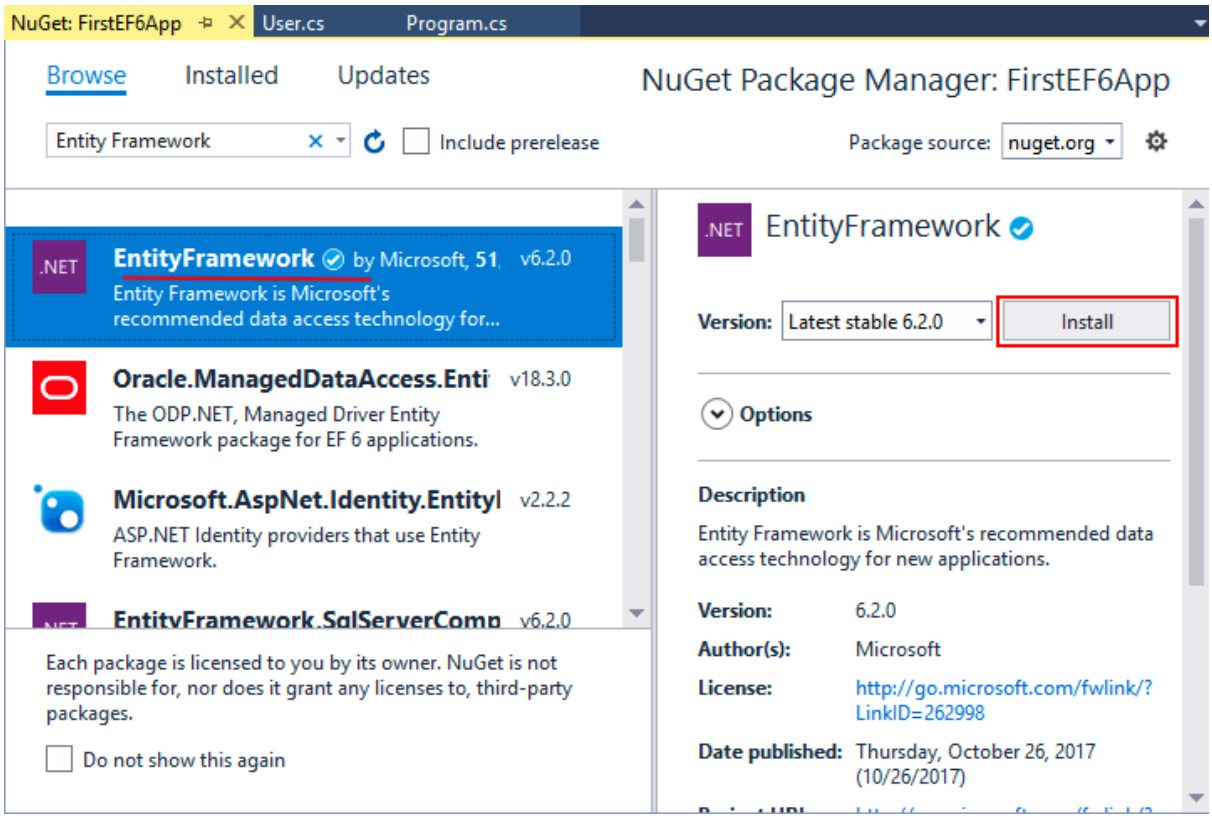

Рисунок 1.11 - Додавання EntityFramework

Після підключення пакету необхідно описати сутності. У наш випадку ми будемо створювати об'єкт User з трьома властивостями: Id, Name і Age.

```
public class User
{
    public int Id { get; set; }
    public string? Name { get; set; }
    public int Age { get; set; }
}
```

Рисунок 1.12 - Створення сутности User

Далі ми ініціалізуємо таблицю на основі нашої сутності.

Рисунок 1.13 - ініціалізація таблиці

Коли база даних створена - можна переходити до побудови запитів. Більшість запитів зводяться до CRUD операціями:

- Create
- Read
- Update
- Delete

Коли основні запити виконані можна підходити до питання їх оптимізації. Один з головних способів заощадити час виконання запиту це звести до мінімуму використання підзапитів. Приклад на рис. 1.15 набагато краще

```
Select Column_A
From Table_1
Where Column_B = (Select max (Column_B From Table_2)
And Column_C = (Select max (Column_C From Table_2)
And Column_D = ‘position_2’
```

Рисунок 1.14 - приклад поганого запиту

написаний, ніж на прикладі рис. 1.15.

```
Select Column_A
From Table_1
Where (Column_B, Column_C) = (Select max (Column_B), max (Column_C)
From Table_2)
```

Рисунок 1.15 - приклад оптимизованого запиту

Також для оптимізації радиться використовувати конкретні імена стовпців після оператора select, замість «*» – це дозволить збільшити швидкість відпрацювання запиту і зменшення мережевого трафіку.

З'єднання таблиць у запиті також є критичним: у випадку, коли з'єднання таблиць відбувається в правильному порядку, то загальне число рядків, необхідних до обробки, що значно скоротиться.

При з'єднанні основний і уточнюючої таблиць переконуємося, що першою буде основна таблиця, в іншому випадку ми ризикуємо отримати обробку набагато більшого числа рядків, ніж необхідно.

Враховуємо, що при з'єднанні таблиць EXIST краще distinct (таблиці відношення «один-до-багатьох»).

1.4 Розробка мобільного додатка

Мобільний додаток-це програмне забезпечення, спеціально розроблене під конкретну мобільну платформу (iOS, Android, Windows Phone і т.д.). Призначено для використання на смартфонах, фаблетах, планшетах, розумних годинниках і інших мобільних пристроях.

Мобільні додатки пишуться на мовах програмування високого рівня, а потім компілюються в машинний код операційної системи для отримання максимальної продуктивності. Розробка додатків має свої особливості: мобільні пристрої працюють від батареї і комплектуються менш продуктивними процесорами, ніж персональні комп'ютери. Крім того, сучасні смартфони і Планшети повсюдно оснащені додатковими пристроями, такими як гіроскопи, акселерометри і фотокамери, що дають унікальні можливості для розширення функціоналу Програми.

Зазвичай мобільні пристрої продаються вже з деякими попередньо встановленими додатками. Решта за бажанням користувача можна завантажити (як платно, так і безкоштовно) на спеціалізованих сервісах: Apple AppStore, Google Play, Windows Phone Store та інших. Перші магазини додатків, такі як Apple AppStore і Android Market, що став згодом Google Play, з'явилися в 2008 році. Через два роки Американське діалектичне ТОВАРИСТВО назвало термін» додаток " словом року.

За підсумками 2015 року в каталогах двох найбільших маркетів було близько 3 млн додатків. Кількість завантажень за один рік перевищила 300 мільйонів.

1.5 Використовувані технології

Для розробки мобільного та браузерної версії системи я використовував технологію Flutter.

Flutter — безкоштовний і відкритий набір засобів розробки мобільного користувальницького інтерфейсу, створений компанією Google і випущений у травні 2017 року. Простіше кажучи, з допомогою Flutter можливо створити власне мобільний додаток з одним масивом коду. Це означає, що для створення двох додатків (IOS і Android) можна використовувати єдину мову програмування і одну базу коду.

Flutter націлений на дві важливі речі:

SDK (Software Development Kit): набір інструментів, який допоможе вам в розробці додатків. Він включає інструменти для компіляції коду в нативному машинному коді (код для IOS і Android).

Framework (Бібліотека інтерфейсу користувача на основі віджетів): Колекція функціональних елементів користувальницького інтерфейсу (кнопок, текстових вводів, повзунків і т. д.), які можна персоналізувати під особисті переваги.

Для розробки з Flutter використовується мова програмування під назвою Dart. Це також мова Google, створений у жовтні 2011 року, але значно збільшився в останні роки.

Flutter фокусується на кроссплатформному програмуванні. Його можна з легкістю використовувати для створення мобільних і веб-додатків.

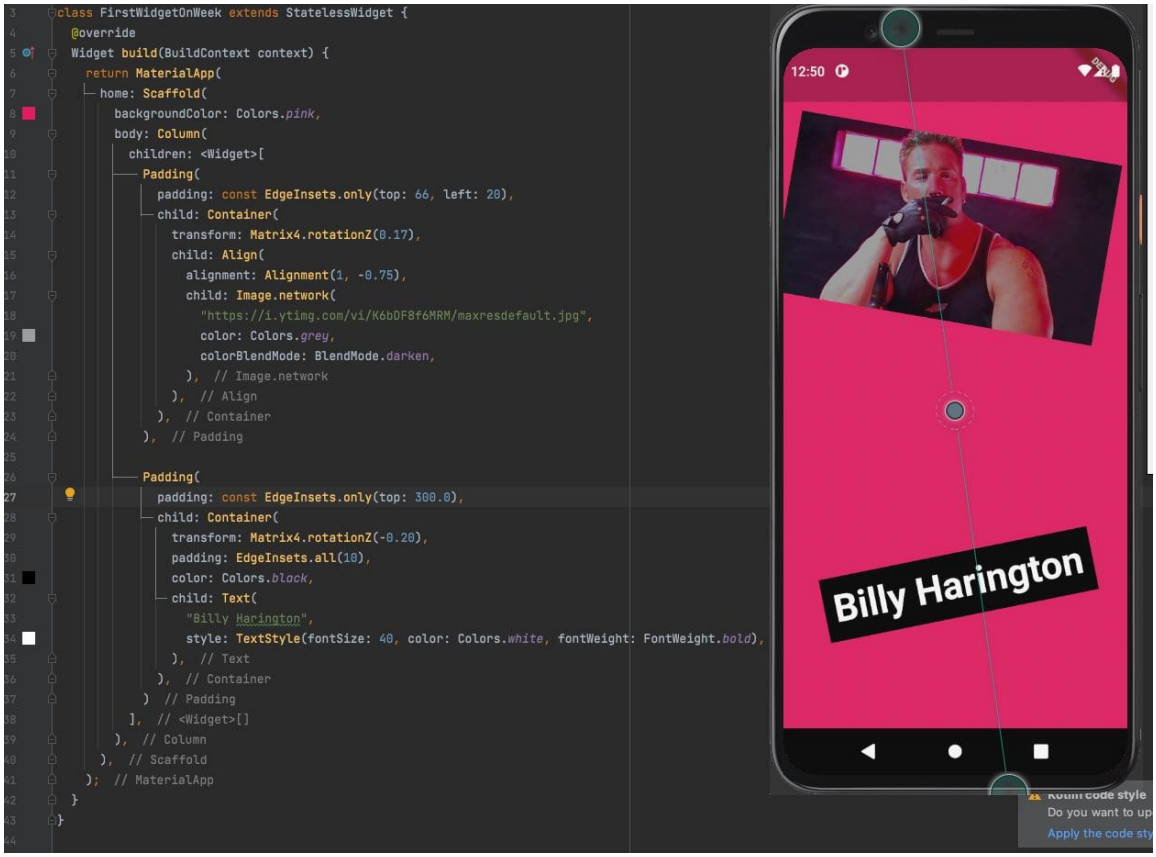

Рисунок 1.16 - приклад верстки на рекламному лендінгу

1.6 Архітектура мобільного додатку

Перед проектування додатки розберемося в загальних рисах з чого воно складається. Dart є об'єктно орієнтованою мовою, а відповідно його framework Flutter перейняв цю рису. У Flutter всі об'єкти називаються віджетами. Знаючи це можна можна розділити основні шари на: графічну і логічну.

До логічної можна віднести репозиторій, сервіси та state management.

State management це спосіб управління станом віджета. Всього існують 4 підходи за даною технологією:

Native State

Для створення Stateful Widget вам потрібно створити 2 класу. Перший клас повинен успадковуватися від Stateful Widget, який в свою чергу успадковується від Widget і є незмінним. Екземпляр цього класу не створюється при кожному відтворенні і використовується для зберігання переданих параметрів і ініціалізації стану. Другий-клас стану, який має доступ до Stateful Widget через внутрішню властивість і займається безпосередньо отрисовкой стану, реагуючи на його зміну.

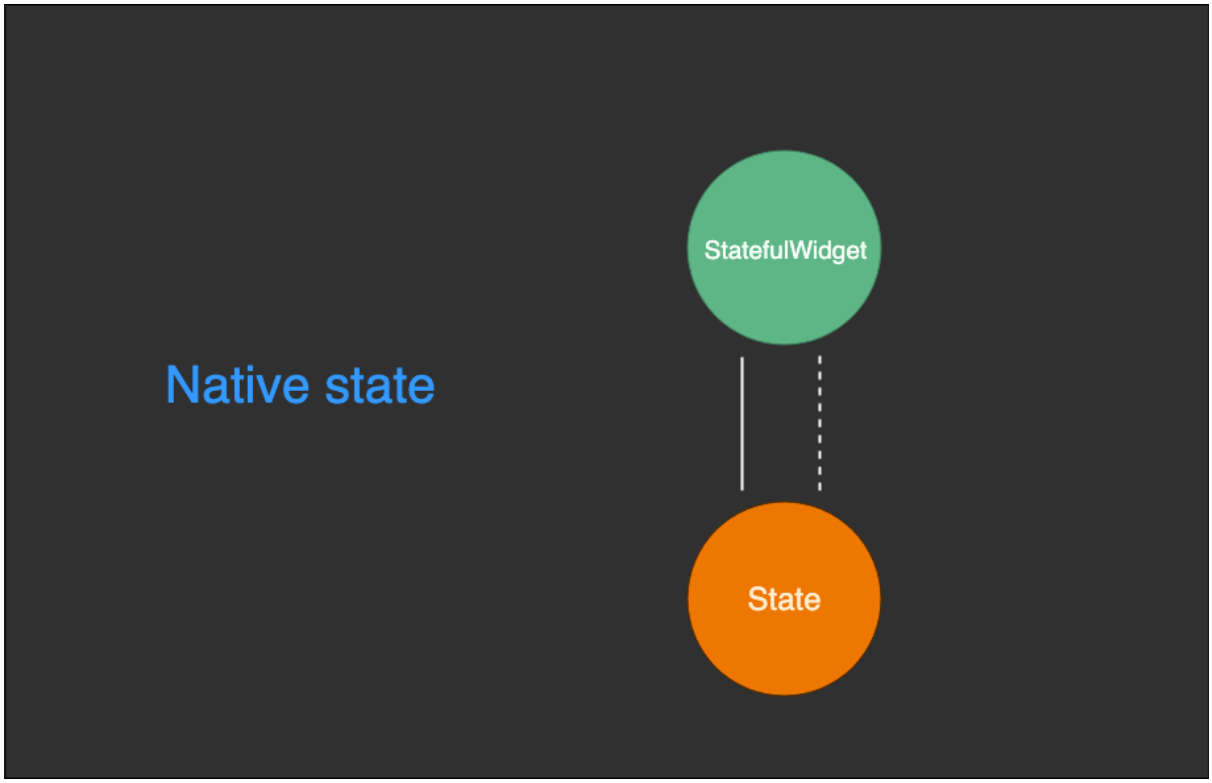

Рисунок 1.17 - приклад-схема Native state

Provider

Такий тип архітектури дозволяє винести бізнес-логіку з подання і дає можливість перевикористовувати цю логіку в різних модулях системи. Результат досягається за допомогою створення моделі і реагування підписаних віджетів на її зміну. Спочатку Brian Egan і Andrew Wilson розробили пакет scoped_model, який витягли з кодової бази Fuchsia і піддали значним поліпшенням. Однак після Google I / O 2019 був представлений новий пакет, provider, який замінює і покращує Scoped Model, дозволяючи передавати моделі вниз по дереву віджетів без ручного використання InheritedWidget. Для локального стану віджета насамперед необхідно створити модель з усіма полями, які будуть використовуватися у вашому віджеті. Після зміни кожного поля (або полів) вам потрібно повідомити передплатникам, що модель змінилася, і виконати отрисовку підписаних віджетів. Щоб підписати віджет на модель, використовується клас ChangeNotifierProvider, який є частиною

бібліотеки provider. Підписка відбувається безпосередньо до того віджету, який буде залежати від даних зі створеної моделі.

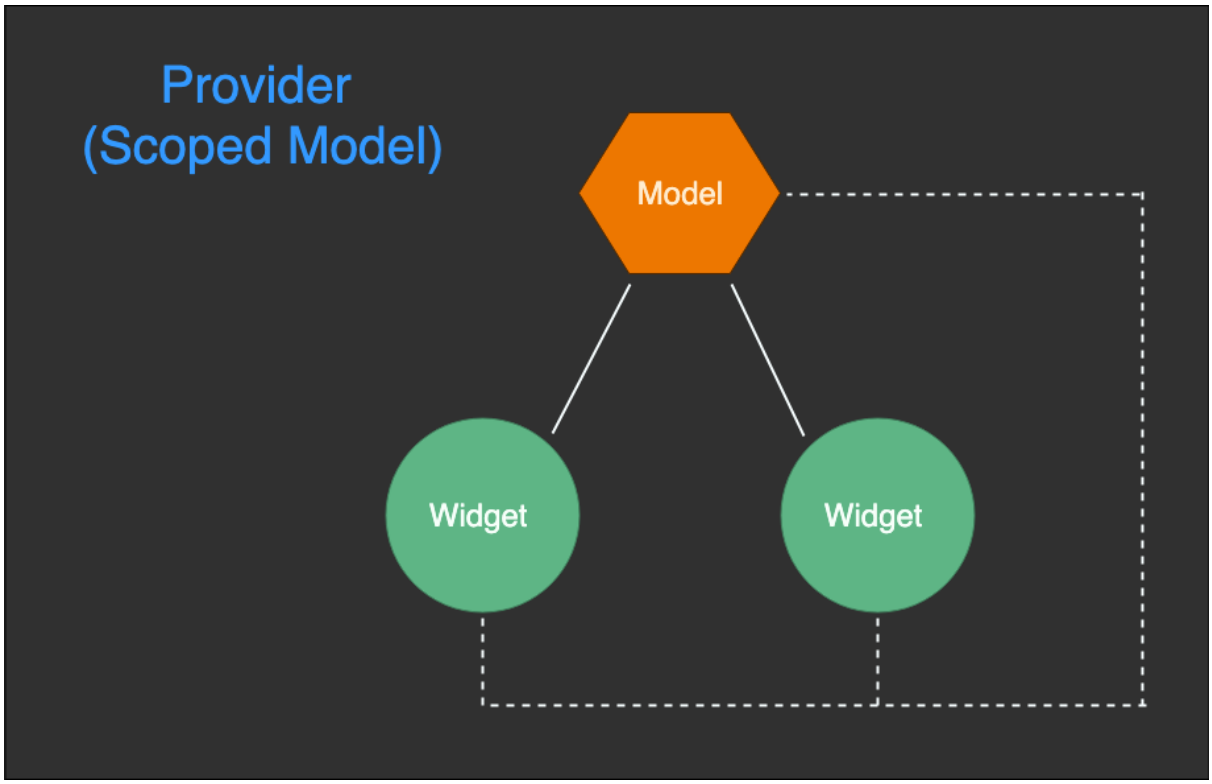

Рисунок 1.18 - приклад-схема Provider

BLoC

BLoC (business Logic Component) - шаблон, створений Google для управління складним станом програми, грунтуючись на реактивній парадигмі.

Основна ідея полягає в тому, що наш додаток розбито на модулі, що реалізують бізнес-логіку. Кожен модуль має одну або кілька Sink (труб), які є деяким вхідним потоком для агрегування подій ззовні. В якості вихідних даних виступає Stream (потік), який визначає асинхронний формат даних для наших віджетів. Щоб скористатися модулем на рівні віджета, застосовують StreamBuilder, який управляє потоком даних і автоматично вирішує проблеми підписки і перемальовування дочірнього дерева віджетів.

Незважаючи на це, використовувати BLoC в чистому вигляді — досить складна робота, оскільки треба застосовувати бібліотеку RxDart для маніпуляції з потоками, вручну відписуватися від потоків, інакше можна отримати серйозний витік пам'яті на великих додатках. З метою вирішення цих проблем була винайдена бібліотека bloc від Фелікса Ангелова, одного з розробників BMW Tech, який по максимуму спростив використання цього шаблону і надав зручне API для управління станом з можливістю легкого тестування модулів. Важливою перевагою цього пакету є можливість автогенерації коду за допомогою плагінів для найбільш популярних IDE (IntelliJ, VS Code). Таким чином, ми не витрачаємо часу на написання зайвого коду і маємо гнучкість в зміні без зайвої магії всередині.

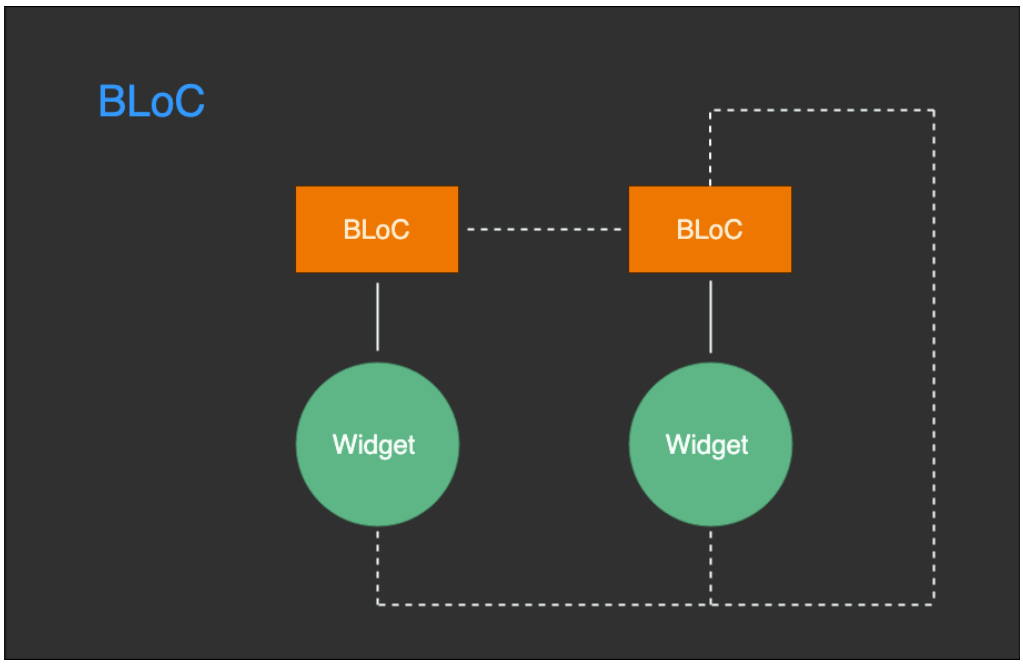

Рисунок 1.19 - приклад-схема BLoC

Redux

Майже всі, хто прийшов у Flutter зі світу фронтенда (зокрема, з React), знають про Flux-архітектурі і найпопулярнішою її реалізації — Redux. Причини популярності прості:

Централізованість-стан всього додатка знаходиться в одному місці, що дозволяє зберігати його в будь-якому зручному для вас сховищі.

Передбачуваність - не потрібно імперативно міняти залежать один від одного моделі, ми просто реагуємо на дії, які посилає нам система. Простота налагодження-завжди є можливість подивитися повне дерево стану, а також можливість time-travel-налагодження, коли ви можете послідовно пройтися по всіх змін в стейті і своєчасно знайти і виправити помилки.

Гнучкість-існує велика кількість middleware-розширень на всі потреби програміста в управлінні станом. Деякі Redux-реалізації портовані на інші платформи, і Flutter не став винятком. У dartpub є пакет redux, який може бути використаний як в Інтернеті, так і на мобільних платформах з впровадженням додаткового пакету flutter_redux. Як вже говорив вище, в Redux немає поділу на локальне і глобальне. Стан завжди глобальне-доступно будь-якому віджету і доступно до зміни через екшени в будь-якому місці системи.

Формально у нас існують такі поняття:

- State-модель стану, яка може бути як скалярним, так і будь-яким іншим складовим типом.
- Action-клас-ідентифікатор події, який зберігає в собі payload для передачі потрібних параметрів.
- Reducer-обробник екшенів, має доступ до поточного стану і екшену, який чекає обробки.
- Dispatch-метод виклику екшену, який обробляється одним з редьюсерів.
- Store-дерево стану програми, комбінує в собі всі редьюсери, які ми визначили в додатку. StoreConnector-дає можливість дочірньому віджету отримати доступ до store.

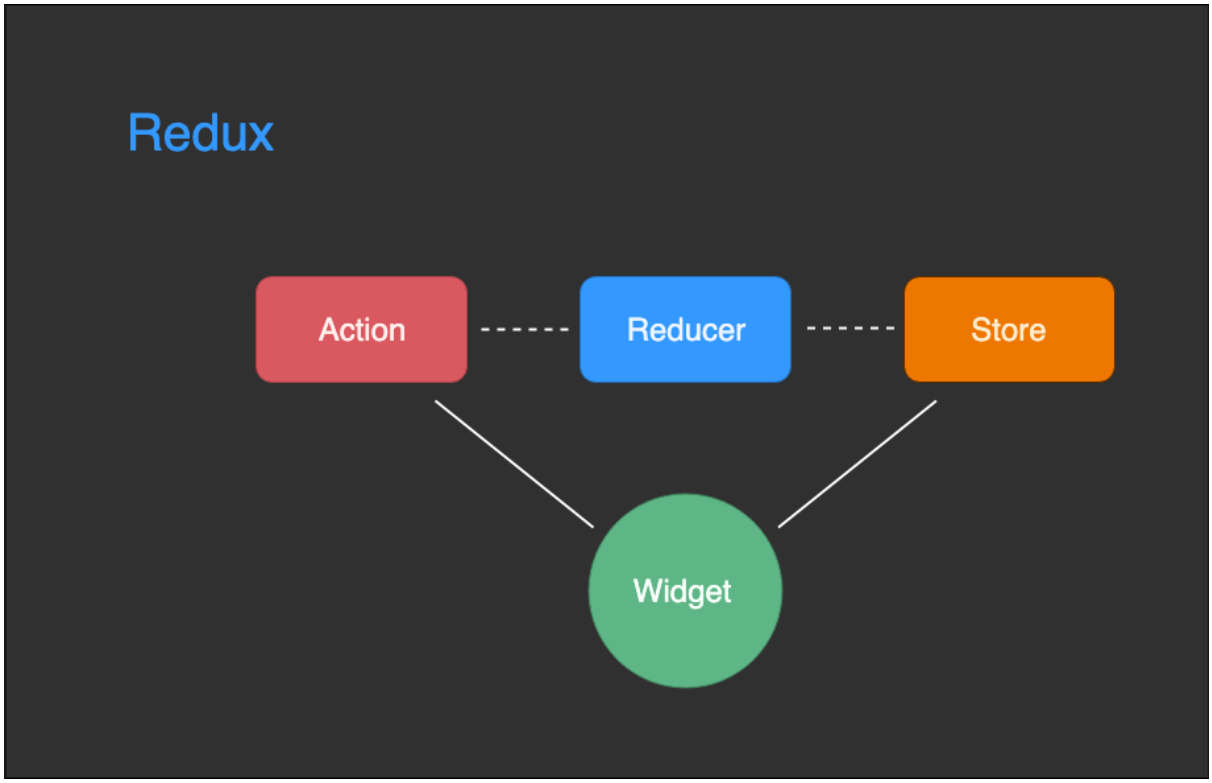

Рисунок 1.20 - приклад-схема Redux

Для свого проекту я використовував store manager Redux. З причин того що він поєднує в собі простоту розширюваності і гнучкість. Redux дозволяє до станів звернутися з будь-якої точки програми, що вкрай зручно для великих проектів.

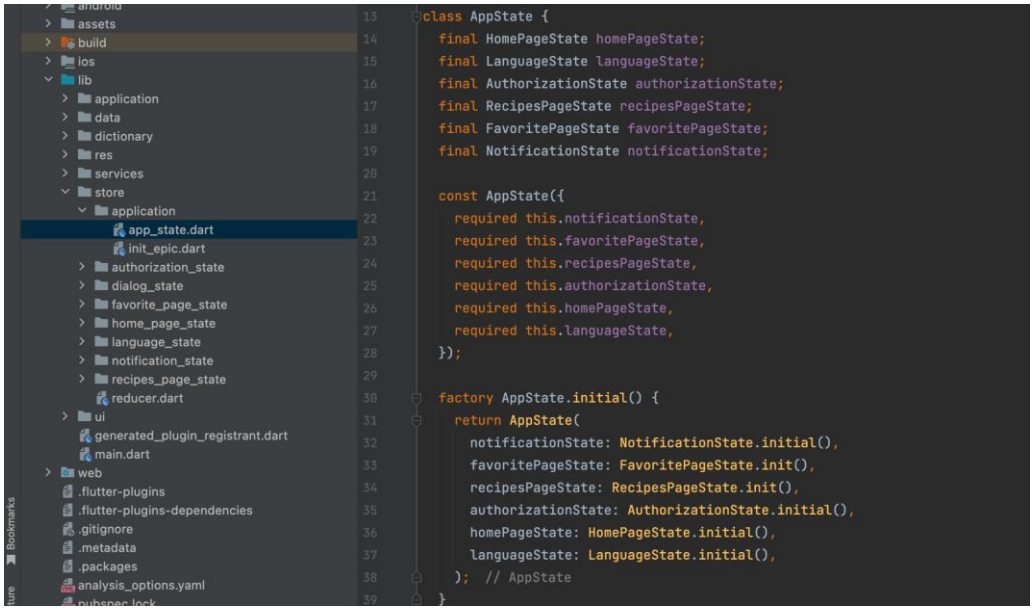

Рисунок 1.21 - шість створенних стейтів

Також одним з плюсів цього архітектурного патерну є можливість шифрування епіків, чим я скористався.

Епік - це потік, що викликається користувачем, який являє собою асинхронний конвеєр, який може не перериваючи повертати різні стани.

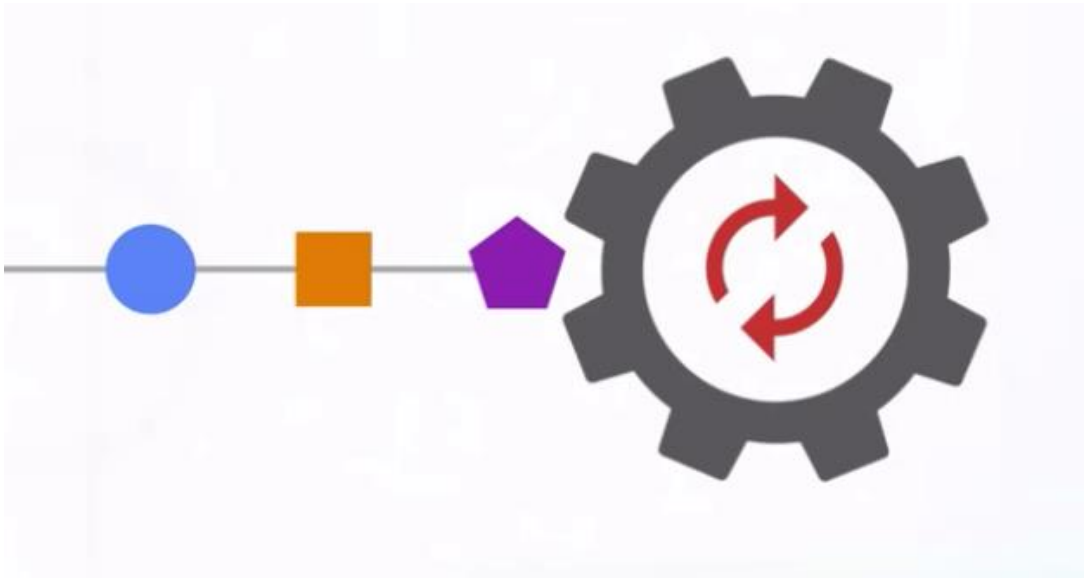

Рисунок 1.22 - “конвеєрний” приклад праці stream

Шифрування досягається за рахунок кастомного елемента патерну-Reducer. Даний компонент є розробкою компанії AppVesto і не має аналогів, що дозволяє гарантувати безпеку потоку.

```
LanguageState reducer(dynamic action) {
  return Reducer<LanguageState>(
    actions: HashMap.from({ChangeLanguageAction: (dynamic action) => _changeLanguage(action)}),
  ).updateState(action, this);  // Reducer
}

LanguageState _changeLanguage(ChangeLanguageAction languageAction) {
  FlutterDictionary.instance.setNewLanguage(languageAction.newLanguage!);
  language = FlutterDictionary.instance.language.toString();
  return copyWith(
    language: languageAction.newLanguage,
    langDict: FlutterDictionary.instance.language,
  );
}
}
```

Рисунок 1.23 - пріклад хешування екшона на стейті локалізації

За рахунок того що Епік це асинхронний потік - я в ній відправляю запити на сервер.

```
class HomePageEpics {
  static final indexEpic = combineEpics<AppState>([
    _getIngredient,
  ]);

  static Stream<dynamic> _getIngredient(Stream<dynamic> actions, EpicStore<AppState> store) {
    return actions.whereType<GetIngredientsAction>().switchMap((action) async* {
      DialogService.instance.show(LoaderDialog());
      ResponseIngredients response = await Repository.client.getIngredients(
        store.state.authorizationState.userDTO.token!,
        FlutterDictionaryDelegate.getCurrentLocale,
        emptyString,
      );
      yield SaveIngredients(ingredients: response.lists);
      DialogService.instance.close();
    });
  }
}
```

```
package:yellow_team_empty_fridge_11_21/services/dialog_service/dialog_service.dart
class DialogService

DialogService it is service for control dialogs. This class it - Singleton, for function using use
```

Рисунок 1.24 - звернення до репозірою та виклик потрібного запиту

1.7 Оптимізація запитів на сервер

Для створення запитів я використовую бібліотеку DIO Brando. Це пакет від китайський розробників, який спрощує роботу з Rest API запитами.

```
import 'package:dio/dio.dart';
void getHttp() async {
  try {
    var response = await Dio().get('http://www.google.com');
    print(response);
  } catch (e) {
    print(e);
  }
}
```

Рисунок 1.21 - приклад створення запиту за домогою Dio Brando

Великий плюс цього пакета що він працює з розширенням Freezed, яке дозволяє генерувати код. Для цього необхідно додати ці пакети в додаток. Це робиться через файл pubspec.yaml

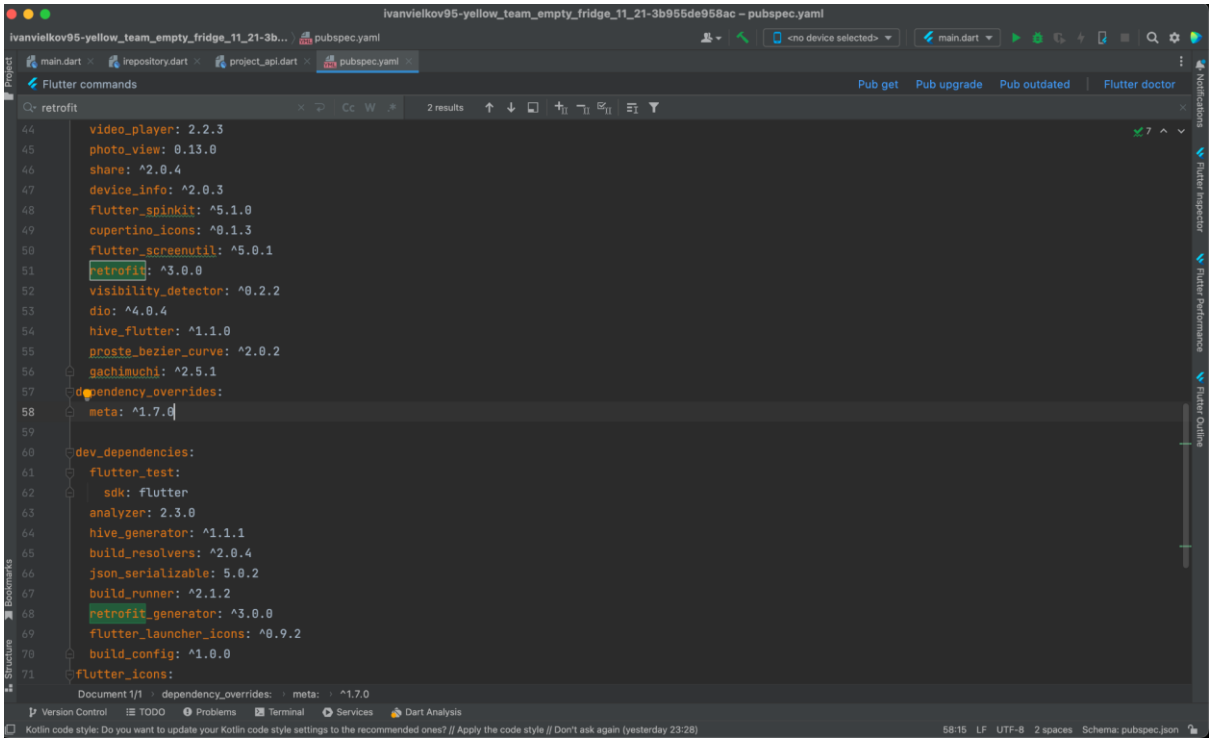

Рисунок 1.22 - файл pubspec.yaml

Коли пакети додані в проект ми їх ініціалізуємо з допомогою команди "pub get" і після чого можемо приступати до побудови запитів. Завдяки використанню Freezed не потрібно писати вручну методи для кожного запиту. Пакет це зробить за автоматично. Що б пакет зрозумів, що йому треба згенерувати - я описую запити в абстрактному класі і позначаю його як потребує генерації.

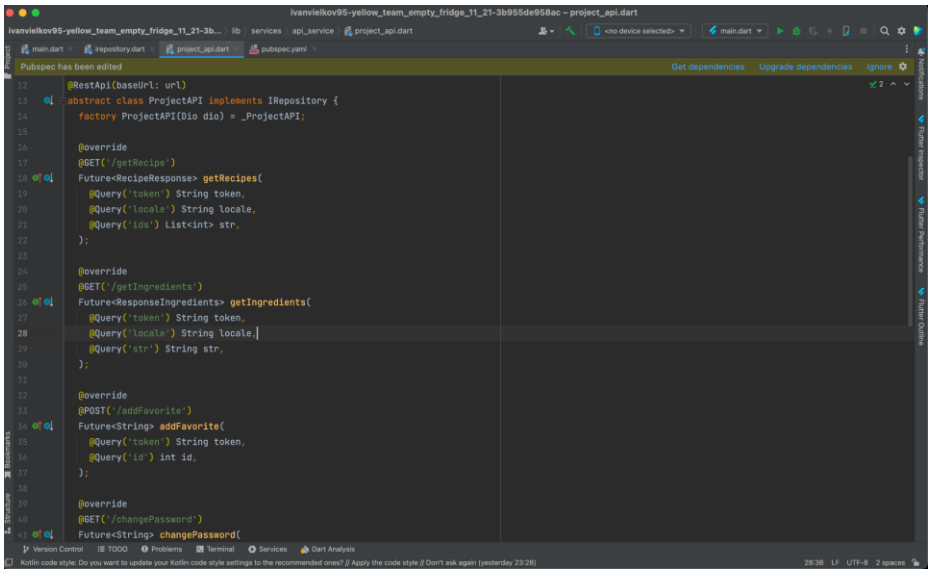

Рисунок 1.23 - необхідний файл для генерации

Після виконаних дій ми отримуємо згенеровані методи, завдяки яким можна комунікувати з сервером.

Оптимізація полягає в частковому кешування даних. Наприклад: у додатку необхідно організувати пошук за елементами, відомо що їх кількість не планує збільшуватися і воно дорівнює 500. Для цього при запуску програми програма відправляє запит на отримання елементів, після чого зберігає їх локально і використовує в подальшому. Даний спосіб також збільшує швидкість роботи програми, так як не доведеться чекати відповіді з сервера після кожного введеного символу в рядок введення.

```
child: ListView.builder(
  itemBuilder: (ctx, index) {
    return Column(
      children: [
        InkWell(
          onTap: () {
            vm.addIngredientInPage(vm.inSearch[index].i);
            _inputText.value = const TextEditingValue(text: emptyString);
          },
          child: Padding(
            padding: const EdgeInsets.symmetric(horizontal: 8.0),
            child: _ingredientBuilder(
              vm.inSearch[index].name,
              vm.inSearch[index].image,
            ),
          ),  // Padding
        ),  // InkWell
        const Divider(
          color: AppColors.kBlack,
          thickness: 0.5,
          height: 5.0,
        ),  // Divider
      ],
    );  // Column
  },
```

Рисунок 1.24 - верстка зкешованих елементів

2 ЕКОНОМІЧНИЙ РОЗДІЛ

2.1 Резюме

В даному дипломному проекті побудована веб-орієнтована система з використанням алгоритмів оптимізації запитів до бази даних з використанням мови C # і платворми ASP.NET Core MVC

Ефективність кожного програмного продукту визначається його якістю та ефективністю процесу розробки. Якість ПП визначається наступними складовими: з точки зору користувача; з позиції використання ресурсів; виконання вимог до програмного забезпечення.Оцінка якості програмного продукту з точки зору користувача визначається необхідним на стадії функціонування розміром оперативної пам'яті ЕОТ, витратами машинного часу, пропускною спроможністю каналів передачі даних. Оцінка якості програмного продукту включає визначення трудомісткості і вартості його створення.

2.2. Визначення трудомісткості розробки програмного забезпечення.

Тривалість розробки програмного продукту залежить від його обсягу, трудомісткості розробки, кваліфікації виконавців, а також планових термінів, визначених умовами ринку. Методом структурної аналогії по відповідних каталогах аналогів програмного забезпечення визначається обсяг програмних засобів, у тисячах умовних машинних команд програми аналога

Каталог аналогів

Таблиця 2.1

Найменування ПП	Обсяг функції ПП – V_o, усл. машинних командах.
1. ПП автоматизованих розрахунків	1300 – 8600
2. Комплексні системи ведення БД	950 – 7430
3. ПП введення інформації	1060 – 5750

У таблиці 2.1 представлені аналоги програмного забезпечення, функції яких, у більшому або меншому ступені, виконує розроблений програмний продукт. Для нашого варіанта виділено сірим кольором.

Вибравши аналог ПП, що містить Vo в умовних машинних командах, трудомісткості визначати на основі табл.2.2

Таблиця.2.2

Обсяг ПП, тис.умов.машинних команд	Норма часу, люд/год
1.00	229
2.00	244
3.00	262

На підставі отриманого значення, по довіднику, визначається укрупнена норма часу на розробку аналога програмного забезпечення (коректується поправочним коефіцієнтом враховуючої умови розробки ПП, тобто в умовах комп'ютера, Кк=0,7÷0,8): T^a = = 244 х 0,7 = 170,08 (люд/годин).

Трудомісткість програмного продукту визначається по кожному етапу розробки окремо на підставі трудомісткості аналога з урахуванням складності

розробки, ступеня новизни і ступеня використання в розробці стандартних модулів на підставі формул:

$$T_{ТЗ} = T^a p \times L_1 \times K_H \qquad (2.1)$$

$$T_{ТП} = T^a p \times L_2 \times K_H \qquad (2.2)$$

$$T_{РП} = T^a p \times L_3 \times K_H \times K_T \qquad (2.3)$$

Для розрахунку необхідні наступні коефіцієнти:

Li – питома вага і-го етапу розробки (див. табл. 2.2.);

Кн – поправочний коефіцієнт, що враховує ступінь новизни (див. табл. 4.3.);

Кт – поправочний коефіцієнт, що враховує ступінь використання в розробці типових програм (див. табл. 4.4.).

Таблиця 2.2. Значення питомих коефіцієнтів трудомісткості стадії в загальній трудомісткості розробки ПП.

Код стадії	Ступінь новизни		
	А	Б	В
ТЗ (L_1)	0,15	0,12	0,12

ТП (L_2)	0,16	0,15	0,11
РП (L_3)	0,55	0,58	0,61

Таблиця 2.3. Значення поправочного коефіцієнта, що враховує ступінь новизни

Код ступеня новизни	Ступінь новизни	Значення $К_н$
А	Принципово нові ПП	1,75 – 1,2
Б	ПП – розвиток визначеного параметричного ряду	1,0 – 0,8
В	ПП маючий аналог	0,7

Таблиця 2.4. Значення коефіцієнта ступеня використання в розробці типових програм

Ступінь охоплення реалізованих функцій розроблювального ПП типовими програмами, %	Значення $К_т$
60 і вище	0,6
40-60	0,7
20-40	0,8
До 20	0,9

Для нашого варіанта виділено сірим кольором.

Тепер розраховуємо трудомісткість по кожному етапу окремо:

Трудомісткість технічного завдання

$$Т_{тз}=Т^а *L_1*К_н= 170,08*0,12*0,7= 14,35 \quad (люд/годин) \qquad (2.1)$$

Трудомісткість розробки технічного проекту

$$Т_{тп}=Т^а *L_2*К_н =170,08*0,11*0,7 = 13,10 \quad (люд/годин) \qquad (2.2)$$

Трудомісткість розробки робочого проекту

$$Т_{рп}=Т^а *L_3*К_н*К_т=170,08*0,61*0,7*0,7= 50,81 \quad (люд/годин) \qquad (2.3)$$

Для подальших розрахунків визначили кількість папера, витраченого на кожен етап: технічне завдання $N_{тз}$= 1 (стр), розробка ТП $N_{тп}$=10 (стр), розробка робочого проекту $N_{рп}$=20(стр), пояснювальна записка відповідно $N_{пз}$=20 (стр)Розрахунок зведений у таблицю 2.5

Таблиця 2.5. Розрахунок трудомісткості ПП

Найменування етапів	Розрахунок, годин.		
1	2	3	4
1.ТЗ	$T_{Pтз}$=14,35	$T_{кк}=0,7*N_{тз}$= 0,7*1=0,7	$T_{нк}=0,15*N_{тз}=0,15*1$ = 0,15
2.Розробка ТП	$T_{Pтп}$=13,10	$T_{кк}=0,7*N_{тп}=0,7*10$ =7,0	$T_{нк}=0,15*N_{тп}=0,15*1$ 0=1,5
3.Розробка РП	$T_{Pрп}$=50,81	$T_{кк}=0,7*N_{рп}=0.7*20$ =14,0	$T_{нк}=0,15*N_{рп}=0,15*$ 20=3,0
4.Розробка ПЗ	$T_{пз}$=1,5 *20 $*N_{пз}$=30	$T_{кк}=0,7*N_{тз}=0,7*20$ =14,0	$T_{нк}=0,15*N_{пз}=0,15*20$ =3,0
Усього, в т.ч.:	©T=151,61		
- на розробку	©$T_р$= 108,26		
- контроль керівника		©$T_{кк}$=35,7	
- нормоконтроль			©$T_{нк}$=7,65

2.3 Розрахунок ціни програмного продукту.

У цьому розділі для визначення ціни розраховуємо основну заробітну плату виконавців, матеріальні витрати, вартість машино – години і витрати на розробку ПО. Розрахунок основної заробітної плати виконавців приведений у таблиці 2.6. Відповідно до статті 8 «Закону про Державний бюджет України на 2022» встановлено мінімальну заробітну плату у місячному розмірі з 1 січня 2022 року - 6500 гривень; мінімальну погодинну тарифну ставку – 39.26 грн.

Таблиця 2.6 Розрахунок основної заробітної плати виконавців.

Найменування робіт	Трудомісткість робіт, години	Погодинна тарифна ставка, грн.	Розрахунок, грн.
1.Розробка ПП	108,26	40,00	4330,40
2.Контроль керівника	35,7	65.00	2320,50
3.Нормоконт-роль	7,65	65.00	497,25
Усього	-	-	©Зо= 7148,15

Зробимо розрахунок матеріальних витрат на розробку ПП. Розрахунок зведемо в таблицю 2.7

Таблиця 2.7 Розрахунок матеріальних витрат на розробку ПО

Найменування матеріальних витрат	Тип, модель	Кількість	Ціна одиниці, грн.	Вартість, грн.
Папір	Лист А4	60	2.50	150,00
				$В_{м\,і}$ =150,00
Транспортно– заготівельні витрати (10%)				$Втр\_з = 0{,}1 \times Вм1$ = 15,00
Усього				$В_м=В_{мі}+В_{тр\_з}$= 150,00+15,00= 165,00

На підставі отриманих даних по окремих статтях витрат складена калькуляція планової собівартості в цілому ПП за формою, приведеною в таблиці 2.8.

Таблиця 2.8. Розрахунок статей витрат планової собівартості

Стаття витрат	Значення, грн.	Формула розрахунку
1. Матеріали	165,00	$В_м$ (див. табл. 2.7)
2. Основна заробітна плата	7148,15	$З_о$ (див. табл. 2.6)
3.Додаткова заробітна плата	1072,22	$Зд = 0,15 \times Зо$ =0,15х7148,15
4.Відрахування до єдиного фонду соціального внеску	1808,48	$Вє.с.в. = 0,22 \times (Зо + Зд)$ =0,22х(7148,15+ 1072,22) =
5. Накладні витрати	2144,45	$Внак. = 0,3 \times Зо$ =0,3 х7148,15
6. Повна собівартість	12338,30	$С_{пов}$= $В_м$ +$З_о$ + $З_д$ +Вє.с.в. + Внак.=

Розмір прибутку, що включається в ціну, визначаємо по наступній формулі: $П=(С_{пов} \text{ х } Р)/100=$ 12338,30 х12/100=1480,60 грн (2.4)

Де Р– плановий рівень рентабельності (10-15%).

Оптова ціна (кошторисна вартість) визначається по формулі:

$Ц_о=С_{пов}+П$=12338,30+1480,60=13818,90 грн (2.5)

Податок на додану вартість визначаємо по наступній формулі:

ПДВ = 0.2х $Ц_о$ = 0,2х13818,90=2763,78 грн (2.6)

Виходячи з отриманих даних, ціна реалізації розробленого програмного продукту на основі наступної формули, становитиме:

$Ц_р$= $Ц_о$+ПДВ =13818,90+2763,78=16582,68грн (4.7)

3 ОХОРОНА ПРАЦІ

Вступ

Важливим моментом в державні політиці України є її ставлення до питань захисту працюючого громадянина через прийняття законодавчих і нормативних актів про охорону праці, створення державних виконавчих структур для забезпечення ефективності їх виконання.

Основні принципи державної політики в галузі охорони праці ґрунтуються на забезпеченні координації державних органів та громадських об'єднань, що вирішують різні проблеми охорони праці, гігієни та безпеки праці.

Основною задачею охорони праці на підприємствах є поліпшення умов праці на робочих місцях.

В розділі охорона праці дипломного проекту розглядаються питання умов праці програміста (оператора ПК), які повинні бути забезпечені на підприємстві для безпечної роботи працівника.

3.1 Небезпечні та шкідливі фактори в роботі користувача ПК.

Небезпечним називається фактор, вплив якого на працюючу людину в певних умовах може привести до виробничої травми або іншому раптовому різкому погіршенню здоров'я. Якщо ж виробничий чинник приведе до захворювання або зниження працездатності, то його вважають шкідливим. Залежно від рівня й тривалості впливу, шкідливий чинник може стати небезпечним.

В процесі роботи на користувачів ПК можуть мати вплив наступні небезпечні та шкідливі фактори:

- ✔ Невідповідність параметрів мікроклімату нормам;
- ✔ Недостатній рівень освітленості;
- ✔ Ураження електрострумом;
- ✔ Статична електрика;

✔ Порушення організації робочого місця тощо.

У відповідності з Правилами охорони праці під час експлуатації ЕОМ на робочому місці користувача ПК повинні бути створенні умови для високопродуктивної праці. Розглянемо ці умови.

Користувач персонального комп'ютера має значне навантаження, як фізичне (сидяче положення, навантаження на очі тощо), так і розумове, що приводить до зниження його працездатності.

Розвитку стомлюваності сприяють наступні фактори – неправильна ергономічна організація робочого місця, нераціональні зони розміщення устаткування по висоті від підлоги, характер протікання праці – чергування праці й відпочинку, зміна одних форм роботи на інші.

3.2 Розробка заходів з охорони праці

3.2.1 Виробничі будівлі та приміщення.

Розміщення робочих місць з ВДТ заборонено у підвальних приміщеннях та на цокольних поверхах заборонено. Для приміщень, які призначені для роботи з ВДТ, доцільно обрати орієнтацію вікон на північ або на північний схід. На вікнах повинні бути жалюзі, що регулюються, або штори, що дають можливість їх повністю закривати. Приміщення відповідно до ДБН В.2.5-28-2006 «Природне і штучне освітлення» повинні мати природне та штучне освітлення. При приміщеннях з ВДТ мають бути обладнані побутові приміщення для відпочинку, психологічного розвантаження тощо.

Площа на одне робоче місце для користувачів повинна складати не менше 6 кв.м, а об'єм – не менше 20,0 куб.м. Стіни пофарбовані матовою фарбою, у відповідності з санітарними вимогами.

3.2.2 Гігієнічне нормування параметрів мікроклімату.

Найбільш значним фактором продуктивності й безпеки праці є виробничий мікроклімат. Він характеризується параметрами температури, вологості і швидкістю руху повітря. Порушення відповідності цих параметрів впливають на працездатність працівників, їх реакцій, збільшення кількості помилок.

Тому в приміщенні повинні бути установлені оптимальні параметри мікроклімату: температура повітря 22-25 ^{0}C, вологість повітря – 40-60%, швидкість пуху повітря – 0,1-0,2 м/с. Для цього приміщення має бути оснащене системами опалення й кондиціювання, що забезпечують постійне й рівномірне нагрівання, циркуляцію й очищення повітря від пилу й шкідливих речовин.

3.2.3 Освітлення виробничих приміщень.

Одним із основних питань охорони праці є організація раціонального освітлення виробничих приміщень і робочих місць.

Для освітлення приміщення, у якому працює користувач ПК, використовується змішане освітлення, тобто сполучення природного й штучного освітлення.

Природне освітлення здійснюється через вікна в зовнішніх стінах будинку.

Штучне освітлення використовують при недостатньому природному освітленні й здійснюють за допомогою двох систем: загального та місцевого освітлення.

Для загального освітлення приміщення використовуються газорозрядні лампи типу ЛД. Норма для необхідної освітленості робочого місця становить 300-500 лк.

3.2.4 Шум і вібрація.

При розумовій праці, яка вимагає зосередженості припустимий рівень шуму становить 50дБ. Для зменшення шуму й вібрації в приміщенні устаткування, апарати й прилади встановлюють на спеціальні прокладки, що амортизують. Якщо стіни в приміщенні є джерелами шумоутворення, вони повинні бути облицьовані звуковбирним матеріалом.

3.2.5 Електробезпека.

На відмінну від інших джерел небезпеки електричний струм не можна виявити без спеціального устаткування й приладів, тому вплив його на людину найчастіше зненацький.

Проходячи через організм людини електричний струм робить термічну, електролітичну і біологічну дію. У результаті термічного впливу викликається розігрів організму й виникають опіки ділянок тіла, у результаті електролітичного впливу розкладається кров і інші органічні рідини в організмі. Біологічний вплив проявляється в порушенні й роздратуванні тканин і мимовільному судорожному скороченні м'язів.

Для попередження поразок електричним струмом необхідно:

- У повному обсязі виконувати правила провадження робіт і правил технічної експлуатації;
- Виключати можливість доступу працівника до частин устаткування, що працює під небезпечною напругою, неізольованим частинам, призначеним для роботи при малій напрузі й не підключеним до захисного заземлення;
- Застосовувати ізоляцію, що служить для захисту від поразки електричним струмом.

Заземлені конструкції, що знаходяться в приміщеннях, де розміщені робочі місця операторів (батареї опалення, водопровідні труби, кабелі із заземленим відкритим екраном) мають бути надійно захищені діелектричними щитками або сітками з метою недопущення потрапляння працівника під напругу.

3.3 Пожежна безпека

Пожежна безпека приміщень, що мають електричні мережі, регламентується ГОСТ 12.1.033-81, ГОСТ 12.1.004-85. Робота оператора ЕОМ повинна вестися в приміщенні, що відповідає категорії Д пожежної безпеки (негорючі речовини й матеріали в холодному стані).

Пожежна безпека забезпечується:

-системою запобігання пожежі;

-системою протипожежного захисту;

- організаційно-технічними заходами.

Протипожежний захист приміщення забезпечується застосуванням установки автоматичної пожежної сигналізації, наявністю засобів пожежогасіння, організацією своєчасної евакуації людей.

Для ліквідації невеликих осередків пожеж, а також для гасіння пожеж у початковій стадії їх розвитку силами персоналу об’єктів, застосовуються первинні засоби пожежогасіння. Це вогнегасники (вуглекислотні та порошкові), пожежний інвентар (покривала з негорючого полотна, ящики з піском, бочки з водою), пожежний інвентар.

ВИСНОВКИ

У наш час самий важливий ресурс – це час. Велику частину свого цифрового життя ми витрачаємо на загрузку певних даних. Значну долю дії можна було бу пропустити, якщо запроси до сервера з клієнту та з сервера до бази даних були оптимізовані. Саме ця задача є наважливу при сучасному обсягу інформації. Але ця відповідальність тримається не тільки на серверних програмістів, але й на frontend розробниках також. Підчас виконання дипломного проекту я працював за типи, та створив запити клієнтського, серверного, СУБД рівнів. Завдяки цім знанням були створені алгоритми з оптимізації запитів, та підключені на створених мобільному додатку та серверу. В результаті виконання проекту було проведено такі роботи:

- Вибрані необхідні засоби розробки;
- Розроблено архітектуру оптимизацаціі запитів
- Розроблено архітектуру пакетів системи;
- Створено необхідні таблиці у базі даних для модуля оптимизацаціі запитів;

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Web Call Server 5: [Электронный ресурс]. URL: https://flashphoner.com/

2. 40 open source, free and top UML tools: [Электронный ресурс]. URL: https://www.predictiveanalyticstoday.com/open-source-free-unified-modeling-language-uml-tools/

3. Что такое ASP.NET: [Электронный ресурс]. URL: http://www.internet-technologies.ru/articles/lekciya-1-chto-takoe-asp-net-installyaciya-i-testovyy-proekt.html

4. Web Call Server - Руководство Разработчика: [Электронный ресурс]. URL: https://flashphoner.com/docs/wcs5/wcs_docs/html/ru/wcs-developer-guide-2/

5. Which is Best for Web Application Development—Dot Net, PHP, Python, Ruby, or Java: [Электронный ресурс]. URL: https://www.addonsolutions.com/blog/which-is-best-for-web-application-development-dot-net-php-python-ruby-or-java.html/ (Дата обращения 15.04.2018);

6. Сравнение современных СУБД: [Электронный ресурс]. URL: http://drach.pro/blog/hi-tech/item/145-db-comparison/

7. Версии лицензий в линейке продуктов Oracle Database : [Электронный ресурс]. URL: https://oracle-patches.com

8. Выпуски и компоненты SQL Server 2014 [Электронный ресурс]. URL: https://msdn.microsoft.com/ru-ru/library/ms144275%28v=sql.120%29 (Дата обращения 17.04.2018)