Форма № 24

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій
(повне найменування інституту, факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)
бакалавра
(ступінь вищої освіти)

на тему РОЗРОБКА ПРОГРАМИ ВЕБЗАСТОСУНКУ «ОБЛІК МАЙНА КОМПАНІЇ»
DEVELOPMENT OF THE KEEPING TRACK OF THE COMPANY PROPERTY WEB APPLICATION

Виконав: студент(ка)4курсу, групи КНТ-129сп
Спеціальності 121 Інженерія програмного забезпечення
(код і найменування спеціальності)

Освітня програма (спеціалізація)
Інженерія програмного забезпечення

Жук В.В.
(прізвище та ініціали)
Керівник Сердюк С.М.
(прізвище та ініціали)
Рецензент Казурова А.Є.
(прізвище та ініціали)

2022

Форма № 25

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування закладу вищої освіти)

Інститут, факультет ФКНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)

Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації)

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
_____С.О. Субботін
“____” _____2022року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Жука Віктора Владиславовича
(прізвище, ім’я, по батькові)

1. Тема проекту (роботи) Розробка програми веб застосунку «Облік майна компанії» Development of the Keeping Track of the Company Property Web Application
керівник проекту (роботи) Сердюк Сергій Микитович, к.т.н., доцент ,
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом закладу вищої освіти від « 27 » квітня 2022 року № 102
2. Строк подання студентом проекту (роботи) 07 червня2022 року
3. Вихідні дані до проекту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)1. Аналіз предметної області. 2. Розробка архітектури програми. 3. Розробка програми. 4. Керівництво програміста. 5. Керівництво оператора

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)
Слайди презентації

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	Сердюк С.М., доцент		
Нормоконтролер	Бєлова А.В., асистент		

7. Дата видачі завдання « 04 » квітня 2022 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Постановка завдання роботи	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області	2 тиждень	Розділ 1
3	Розробка архітектури програми	3 тиждень	Розділ 2
4	Розробка програми	4тиждень	Розділ 3
5	Тестування та експериментальне дослідження програми	5 тиждень	Розділ 4
6	Створеннякерівництва оператора програми	6 тиждень	Розділ 5
7	Оформлення пояснювальної записки та документів до неї. Нормоконтроль та рецензування	7 тиждень	Додатки
8	Захист роботи	8 тиждень	

Студент(ка) Жук В.В.
(підпис) (прізвище та ініціали)

Керівник проекту (роботи) Сердюк С.М.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 78 с., 23 рис., 3 табл., 3 дод., 15 джерел.

МАЙНО, ОБЛІК, ВТРАТА ВАРТОСТІ, МАТЕРІАЛЬНІ РЕСУРСИ, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, БАЗА ДАНИХ.

Об’єкт дослідження – процес обліку матеріального майна компанії. Предмет дослідження – програмне забезпечення обліку майна компанії. Мета дослідження – розробити програмне забезпечення обліку матерільних ресурсів компанії з урахуванням втрати вартості з часом.

Матеріали, методи та технічні засоби: мова C#, мова TypeScript, вебфреймворкAngular, EntityFramework, технологія ASP.Net.coreWeb API персональний комп’ютер під керування Windows 10.

Результати. У результаті виконано аналіз аналогів і їх порівняння з розробкою, представлено результати порівняння фреймворків для розробки вебзастосунківмовамиTypeScript та C#, обрано фреймворк Angulary цій якості, представлено діаграму варіантів , схему бази даних, схему функціонування програми, описано створені одиниці програми та загальну логіку роботи з програмою.

Висновки. Програма дозволила створити якісний інструмент обліку майна компанії з урахуванням втрати вартості, зручною категорізацєю, імпортом даних з інших файлів, створення звітів на основі даних, що маються.

Галузь використання.Облік майна компанії та бізнесу.

ABSTRACT

Explanatorynotetothediplomaqualifyingworkofthebachelor: 78pages,23 figures, 3tables, 3 appendixes, 15sources.

PROPERTY, ACCOUNTING, LOSS OF VALUE, MATERIAL RESOURCES, SOFTWARE, DATABASE.

Theobjectofresearchistheprocessofaccountingfortangibleassetsofthecompany . Thesubjectoftheresearchisthecompany'spropertyaccountingsoftware. Thepurposeofthestudyistodevelopsoftwareforaccountingforthecompany'smaterialresources, takingintoaccountthelossofvalueovertime.

Materials, methodsandtechnicalmeans: C # language, TypeScriptlanguage, Angularwebframework, EntityFramework, ASP.Net.coreWeb API technologypersonalcomputerrunning Windows 10 .

Results. As a result, theanalysisofanaloguesandtheircomparisonwithdevelopmentisperformed, theresultsofcomparisonofframeworksforwebapplicationdevelopmentinTypeScript and C # languagesarepresented, Angularframeworkinthiscapacityisselected, variantdiagram, databasescheme, programoperationschemearepresented. workwiththeprogram.

Conclusions. Theprogramallowedtocreate a high-qualitytoolforaccountingforcompanyassets, takingintoaccountthelossofvalue, convenientcategorization, importingdatafromotherfiles, creatingreportsbasedonavailabledata.

Fieldofuse. Accountingforcompanyandbusinessassets.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

MVC – Model-View-Controller;

EF – EntityFramework;

СУБД – Система управління базами данних;

БД –База даних.

ВСТУП

Майно організації дозволяє їй вести господарську діяльність та отримувати дохід. Тому ведення бухгалтерського обліку та облік майна підприємства нерозривно пов'язані. Організація обліку майна на підприємстві передбачає створення системи зі збору інформації про майнове становище підприємства з її подальшим аналізом.

Майно організації[1] можна поділити на такі позиції:

– гроші;

–матеріальні активи;

–нематеріальні активи.

Одним з головних показників матеріальних активів є втрата вартості з часом. Відстеження чього показника дає компанії можливість бачити загальну капіталізацію, прибутковість того чи іншого матеріального активу (як от станок для виробництва паперу), та розуміти доцільність його використання про виробництві.

Облік майна також необхідний для того, щоб розуміти яка кількість того чи іншого матеріального активу є в наявності, чи потрібно замовити його додатково, або доцільніше його продати.

У великих компаніях гостро стоїть питання спрощення ведення обліку матеріальних активів. Велика кількість майна потребує багато спеціалістів з ведення обліку. Більшість продуктів з обліку майна мають застарілий функціионал, часто не мають зв`язку з інтернетом. В епоху глобалізації суумнівів у необхідності можливості віддалено взаємодіяти з іншими місцями збереження майна немає.

Тому в результаті в дипломній роботі було прийняте рішення про розробку веб-застосунку з обліку матеріальних активів компанії.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Облік майна

Облік майна компанії або бізнесу є одним з найпопулярніших напрямків реалізації програмного забезпечення. Однак облік майна це дуже скалдний та важливий процес адже компанія може мати дуже багато матеріальних активів у своемурозпоряджені. Це питання гостро стоїть як для малого так і для великого бізнесу.

Загалом облік матеріальних ресурсів компанії є важливим через наступні причини:

– точне розуміння фінансового та матеріального стану компанії, що є основним показником капіталізації більшості виробництв;

– більшість компаній володіють великою кількістю майна через що процес обліку цього майна стає великою проблемою для бізнесу адже доводиться наймати більше персоналу. Обілк майна компанії програмними засобами полегшує роботу обліковця а одже немає необхідность наймати більше персоналу для цієї роботи;

– відстежування втрати вартості матеріальних активів дозволяє компанії чітко розуміти доцільність придбання, продажу або ж оренди будь-яких матеріальних ресурсів;

– дуже часто необхідно робити звіти по ноявності матеріальних ресурсі, їх ціні, кількості, технічному становищу;

– поділ матеріальних ресурсів на категорії та типи скорочує час для компанії між отриманням заяви на перевірку наявності якогось ресурсу та отримання результатів;

– можливість швидко продати якийсь матеріальний ресурс є важливою частиною роботи будь-якого бухгалтера та обліковця, спрощення цих механізмів скорочує час на ці дії, а отже компанія отримує більше прибутку через швидшу преакцію на зміни;

– можливість швидко та з будьякого місця отримати інформацію про матеріальне становище компанії, побачити технічний стан того чи іншого матеріального ресурсу, створити звіт на основі даних, що маються – це величезнию плюс при наявності веб-застосунку, що веде облік майна;

– спрощенне додавання матеріального ресурсу до БД зменшує необхудну кваліфікацію для бухгалтерів через що компанія може економити гроші на наймі менш кваліфакованої робочої сили.

1.2 Огляд програмних аналогів

Прогама 1С: ERP (рис. 1.1) [2] є одним з розповсюджених засобів обліку майна.

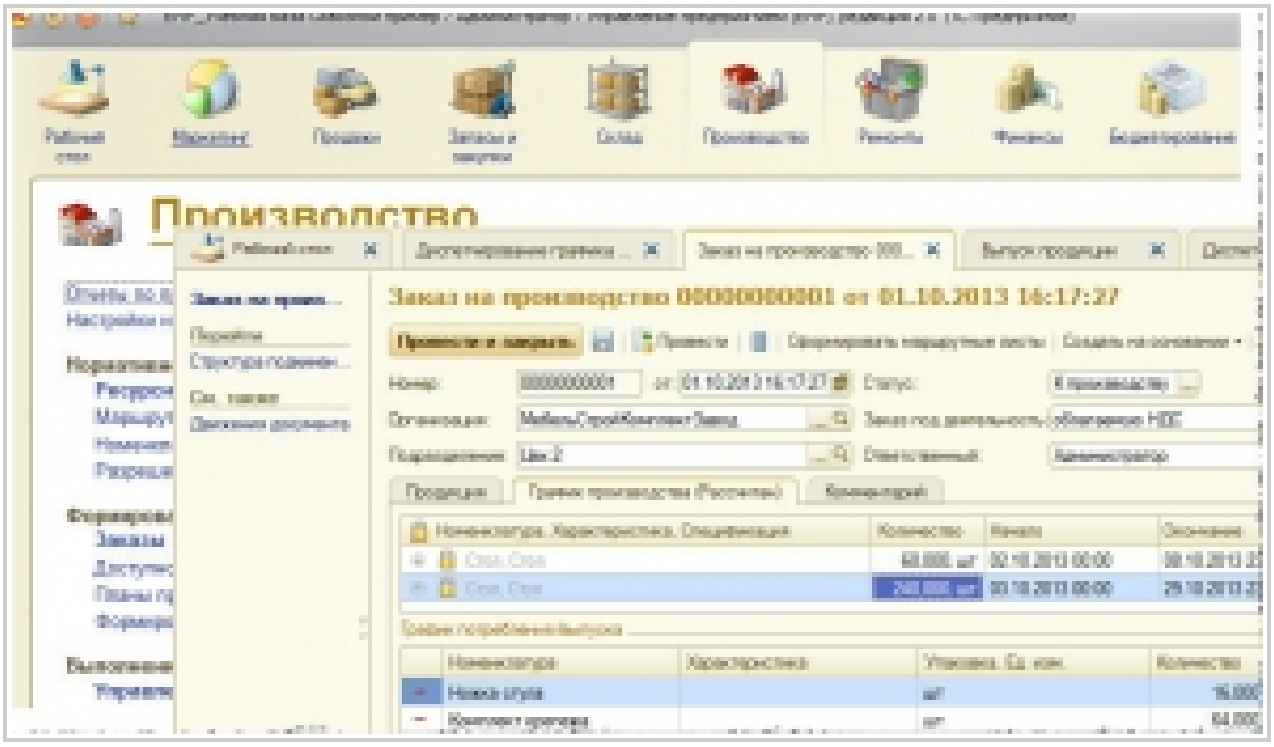

Рисунок 1.1 – Програма 1С: ERP

1С дозволяє вести облік майна. Вона надає можливість вести облік майна компанії, але не є вебзастосунком через що користувач не має можливость отримати достуа до неї з будь-якого приладу (телефон, веб

браузер). Має застарілий інтерфейс і дуже складний функціонал. Не має можливості отримувати інтеграції з іншими відомими веб-сервісами,якіможуть знадобитись при використанні програми обліку майна. Також має орієнтацію на великий та середній бізнес.

Другим засобом є програма Microsoft Dynamics 365 (рис. 1.2) [3].На мою думку еталон програми аналітики бізнесу, але вона все ж має орієнтацію на великий та середній бізнес, через що основними функціями програми є облік персоналу, нематеріальних ресурсів та маркетинг. Програма має багато інтеграцій з сервісами Microsoft, що є плюсом. Але через те, що ця програма створена під американський ринок, вона не підходить більш простій системі оподаткування України. Також має багато функцій, які не потрібні малому бізнесу, але за які доведеться платити.

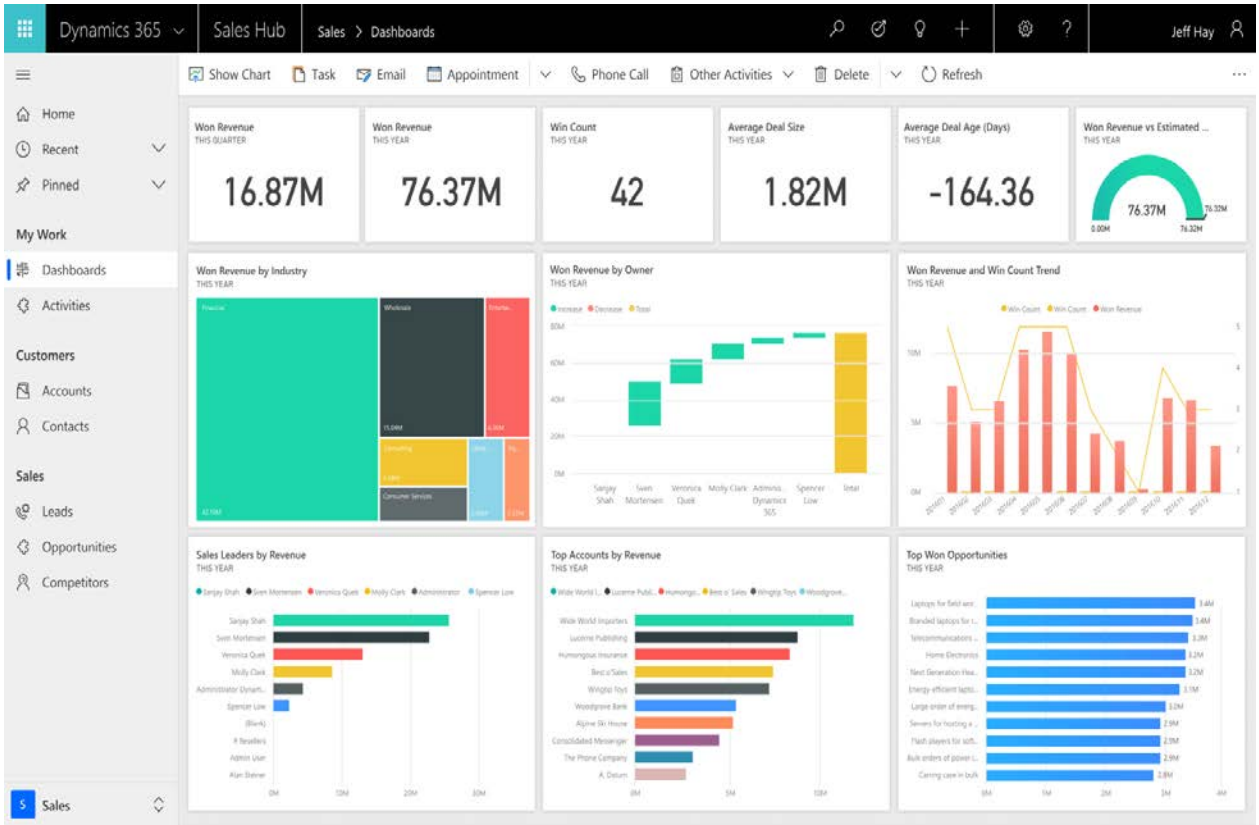

Рисунок 1.2 – Програма Microsoft Dynamics 365

Третьою програмою, що було окремо виділено, є програма Парус-Підприємство(рис. 1.3) [4]. Програмний продукт класу ERP, що відповідає

найвищим вимогам до подібних рішень і дозволяє вирішувати такі облікові та управлінські завдання: управління фінансами, персоналом, бухгалтерський та податковий облік, розрахунок заробітної плати, управління конкурсними закупівлями, виробництвом, автотранспортом, складською логістикою, взаємовідносинами з клієнтами (CRM), діловими процесами, контакт-центром (contact-centre), страховою діяльністю та інші завдання. Має орієнтацію на російський ринок, не має можливості інтеграцій з іншими сервісами, не є вебзастосунком.

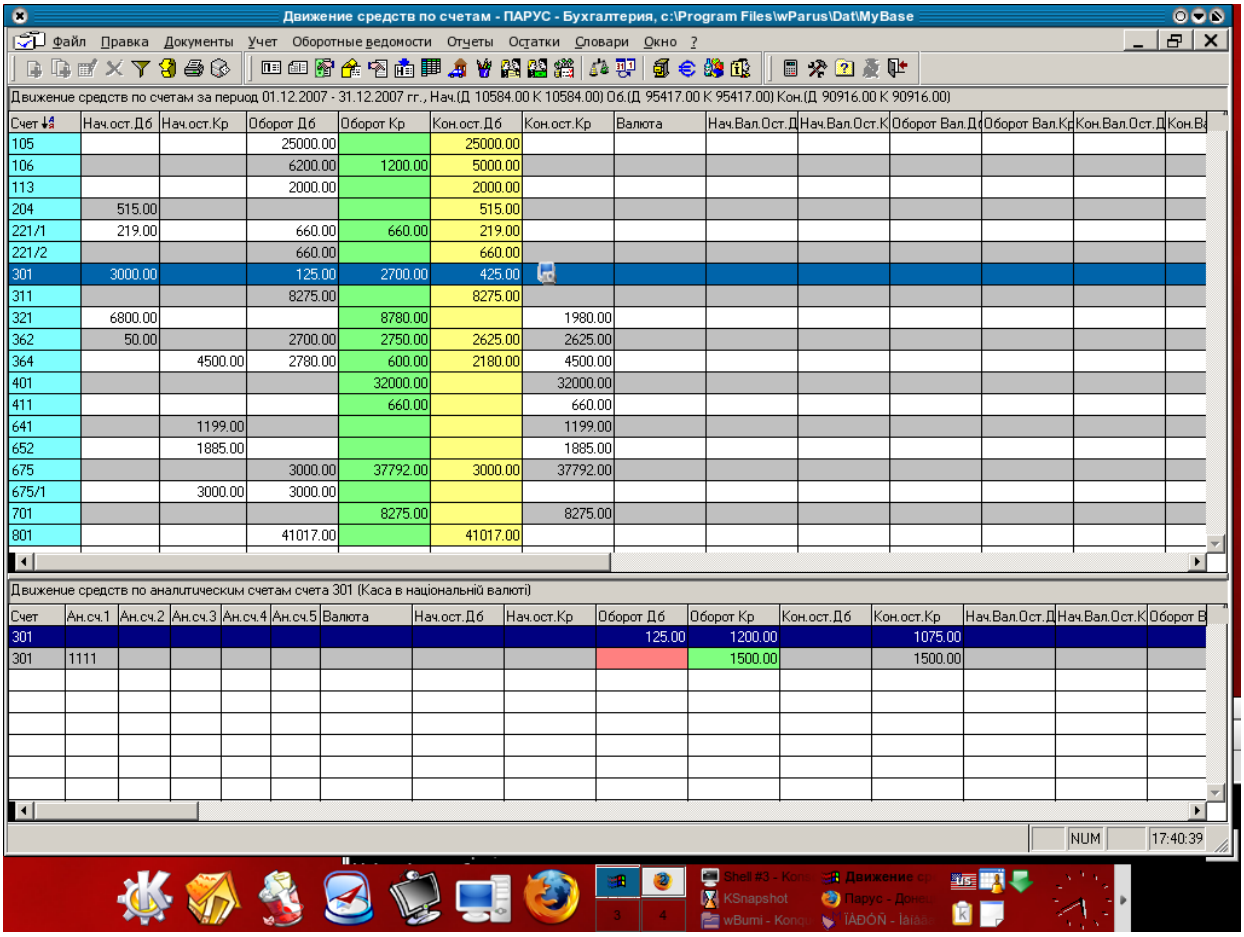

Рисунок 1.3 – Програма Парус-Підприємство

Ці аналоги виконуваної розробки було порівняно між собою і з цією розробкою. Результати представлені в табл. 1.1.

Виконувана розробка повинна мати можливості, що стосуються не тільки обліку майна компанії, але й можливість саме в системі продати

частину майна, відстежити втрату вартості майна різними методами, створити звіт на основі даних, що маються. Також більшість таких програм зараз мають застарілий інтерфейс. Основною метою створення програми є орієнтація саме на потреби малого бізнесу.

Таблиця 1.1 – Порівняння аналогів з виконуваною розробкою

Показник	1С: ERP	Microsoft Dynamics 365	Парус-Підприємство	Розробка
Орієнтація на малий бізнес	Ні	Ні	Ні	Так
Веб-застосунок	Ні	Так	Ні	Так
Інтеграція зі сторонніми сервісами	Ні	Так	Ні	Так
Хмарне сховище	Ні	Так	Ні	Так
Орієнтація на матеріальні ресурси	Так	Ні	Ні	Так
Оновлений інтерфейс	Ні	Так	Ні	Так
Відстеження втрати вартості	Ні	Так	Ні	Так

1.3 Висновки за розділом 1 та завдання роботи

Завдання роботи:

– дослідження проблемиведення обліку майна;

– визначення засобів розробки;

– визначеннявимог до програми;

– проєктування програми ведення обліку майна з відстежуванням втрати вартості;

– програмна реалізація.

Розглянуто актуальність і важливість ведення обліку майна з відстеженням втрати вартості з визначенням причин цієї важливості, що показують необхідність мати зручний інструмент обліку майна. Виконано огляд програмних засобів, що включають програми 1С: ERP, Microsoft Dynamics 365, Парус-Підприємство, визначено особливості розроблюваної програми порівняно з цими аналогами.

2 РОЗРОБКА АРХІТЕКТУРИ ПРОГРАМИ

2.1 Визначення функціональнихвимог до програми

Згідно з технічним завданням сновною направленістю програми є робота з обліком майна, підрахунком втраченної вартості майна а також формування звітів. Зважаючи на це, було визначено такі функціональні вимоги до програми:

–перегляд таблиці ассетів;

–додавання нових ассетів;

–редагування існуючих ассетів;

–перегляд, додавання, редагування та видалення користувацьких типів;

–перегляд, додавання, редагування та видалення користувацьких категорій;

–імпорт ассетів з файлу;

–експорт ассетів у файл;

–продаж ассетів;

–пошук ассетів у таблиці;

–створення звіту на основі інформації в таблиці ассетів;

–експорт звіту у формати xlsx та pdf.

Усі ці вимоги було перетворено на набір сценаріїв, за якими бухгалтер, тобто користувач, який взаємодіє з обліком ассетів і створює їх, працює з програмою.

До сценаріїв роботи рецензента належать:

– перегляд таблиці ассетів;

– додавання нового ассету;

– редагування ассету;

– продаж ассету;

– пошук ассетів у таблиці;

– перегляд користувацьких типів, що включає в собі додавання, редагування та видалення користувацьких типів;

– перперегляд користувацьких категорій, що включає в собі додавання, редагування та видалення користувацьких категорій;

– імпорт ассетів з файлу, та експорт у файл;

– перехід в модуль звітів, що в ключає в собі можливість переглянути, відсортувати звіт або ж єкспортувати його у форматі pdf чи xlsx.

2.2 Вибір засобів розробки

Розробку вебзастосунку заплановано на основі мови TypeScript[5] для фронтенду через її придатність для вирішення широкого кола задач і зокрема створення вебзастосунків, які за рахунок використання вебфреймворку придатні для використання у бізнесах різного рівня, хоча на рівень впливає і вибір фреймворка. Для бекенду було обрано мову C#[6] через її швидкодійність, можливість працювати в декілька потоків, гнучкість при використанні шаблонів проектування, та орієнатцію на високонавантажені застосунки. Тому тепер потрібно порівняти (табл. 2.1) два найпопулярніші фреймовркивеброзробки для TypeScript: Angular[7]aReact[8]. Також необхідно порівняти (табл. 2.2) дві найпопулярніші технології мови C#: ASP.Net.Core[9], а самеWebApiта Blazor[10].

Таблиця 2.1 – Вибір вебфреймворку розробки мовою TypeScript

Показник	Angular	React
Високорівневий API	+	–
Висока оптимізація	+	–
Кросплатформеність	–	+
Формування динамічних вебсторінок	+	+
Використовувана мова програмування	TypeScript	TypeScript

Таблиця 2.2 – Вибір технології розробки мовою C#

Показник	ASP.Net.CoreWeb API	ASP.Net.CoreBlazor
Висока оптимізація	+	+
Кросплатформеність	+	+
Використання стороннього фреймворку для фронтенду	+	–
Використовувана мова програмування	C#	C#

Вибір у підсумку фреймворку Angularобґрунтований його основними перевагами, що включаютьвисокорівневий API, завдяки чому спрощено початок розробки вебзастосунку через відсутність необхідності самостійного встановлення великої кількості бібліотек. Високий рівень оптимізації також є важливим показником для розробки високонаванntаженної системи. Враховуючи вибір цього фреймворку, за основу було взято компонентну архітектуру [11] для фронтенду.

Вибір у підсумку “чистого” ASP.Net.Coreобгурнтований в основному можливістю використовувати сторонній фреймворк для розробки фронтенду.

Враховуючи вибір цих технологій, за основу було взято шаблон проектовуннняModel-View-Controller (MVC)[12].

2.3Проєктування бази даних

Проєктування БД є важливою частиною проєктування застосунку. Для створення БД було обрано EntityFramework[13] (EF) на базі технології .Net. EF має декілька підходів[14] для проектування БД:

– Code-first[15] – вперше з'явився в EntityFramework 4.1, зазвичай використовується, коли у вас вже є існуюча програма, що містить модель даних. Ця модель зазвичай описується за допомогою декількох класів і коду взаємодії між цими класами;

– Model-First – вперше з'явився у версії EntityFramework 4, застосовується розробниками, які не хочуть використовувати інструменти СУБД для створення та управління базами даних, а також не хочуть вручну налаштовувати класи моделі EDM. Фактично це найпростіший підхід під час роботи з EntityFramework. Проектування моделі відбувається у графічному дизайнері EDM середовища VisualStudio;

– Database-First – робочий процес створення моделі починається зі створення та проектування бази даних. Після створення сутнісних класів моделі з існуючої бази даних, робота з EntityFramework аналогічна підходам Code-First і Model-First. Це означає створення об'єкта класу контексту та використання цього об'єкта для виконання необхідних завдань.

Для розробки було обрано підхід Code-first через відсутність необхідності роботи зі сторонніми засобами окрім середи розробки та кращу сумісність створенної БД з кодом, що було написано через відсутність конфліктів типів данних.

Збереження майна(далі – ассет) виконується таблицею CompanyAssetsзанаступними полями:

– id– ідентифікатор;

– Type– користувацький тип ассету;

– PurchaseDate– дата покупки ассету;

– AssetName– назва ассету;

– PurchaseAmount– ціна покупки ассету;

– DepreciationStartDate – дата початку процессу втрати вартості ассету;

– DepreciationEndDate – дата закінчення процессу втрати вартості ассету;

– BalanceAccountId – ідентифікатор аккаунту баланса;

– DepreciationAccountId – ідентифікатор аккаунту втрати вартості;

– PlAccountId – ідентифікатор аккаунту прибуток\збиток;

– Kind – бізнес тип ассету;

– Rate – процент втрати вартості при використанні методу процента;

– SalvageValue – ліквідаційна вартість ассету;

– StoppedAt – дата та час припинення процесу втрати вартості;

– StoppedById - ідентифікатор користувача, що зупинив процес втрати вартості;

– IdWithinCompany – внутрушнійвдентифікаторассету;

– AccountingTypeId – ідентифікатор типу обліку;

– BookedBeforeDate – користвуцька дата закінчення процесу втрати вартості;

– Comment – коментар до ассету;

– CategoryId – ідентифікатор категорії ассету;

– PeriodKind – тип періода;

– PostingType – тип постінгу;

– BookedBeforeAmount – користувацька вартість на початок втрати вартості;

– ImportId – ідентифікатор імортуассета;

– CompanyAssetTypeId – користувацький тип ассету компанії;

– SerialNumber -серійний номер ассету;

– OwningWay – спосіб володіння ассетом;

– VendorId – ідентифікатор клієнта;

– StampNo – номер штампу;

– AddedAt – дата зміни ассету;

– AddedById – ідентифікатор користувача, що змінив ассет;

– CreatedAt – дата та час створення ассету;

– CreatedById – ідентифікатор користувача, що створив ассет;

– LastUpdatedAt – дата та час останної зміни ассета;

– LastUpdatedById – ідентифікатор користувача, що останнім змінював ассет;

– DeletedAt – дата та час видалення ассету;

– DeletedById – ідентифікатор користувача, що видалив ассет;

– поле VendorId є зовнішнім ключем з таблиці CompanyCustomers. Таким чином реалізується зв’язок між таблицями CompanyAssets і CompanyCustomers, що визначений як один до багатьох: один клієнт може користуватись декількомаассетами але не може бути такого, що одним і тим самим ассетом користуються декілька клієнтів;

– поле CompanyAssetTypeId є зовнішнім ключем з таблиці CompanyAssetTypes. Таким чином реалізується зв’язок між таблицями CompanyAssets і CompanyAssetTypes, що визначений як один до багатьох: один ассет може мати один тип, але один тип може мати багато ассетів;

– поле CategoryId є зовнішнім ключем з таблиці CompanyCatefories. Таким чином реалізується зв’язок між таблицями CompanyAssets і CompanyCatefories, що визначений як один до багатьох: один ассет може мати одну категорію, але одна категорія може мати багато ассетів;

– поле BalanceAccountId є зовнішнім ключем з таблиці CompanyAccounts. Таким чином реалізується зв’язок між таблицями CompanyAssets і CompanyAccounts, що визначений як один до багатьох: один

ассет може мати одинрахунокбалансу, але одинрахунокбалансу може використовуватись в великій кількості ассетів;

– поле DepreciationAccountId є зовнішнім ключем з таблиці CompanyAccounts. Таким чином реалізується зв’язок між таблицями CompanyAssets і CompanyAccounts, що визначений як один до багатьох: один ассет може мати один рахунок втрати вартості, але один рахунок втрати вартості може використовуватись в великій кількості ассетів;

– поле PlAccountId є зовнішнім ключем з таблиці CompanyAccounts. Таким чином реалізується зв’язок між таблицями CompanyAssets і CompanyAccounts, що визначений як один до багатьох: один ассет може мати один рахунок балансу, але один рахунок прибуток\збиток може використовуватись в великій кількості ассетів;

– поле ImportId є зовнішнім ключем з таблиці CompanyAssetImports. Таким чином реалізується зв’язок між таблицями CompanyAssets і CompanyAssetImports, що визначений як один до багатьох: один ассетможе бути імпортований тільки один раз, але за один імортможно імпортувати багато ассетів.

Збереження бугалтерських рахунків виконується таблицею CompanyAccounts за наступними полями:

– Id – ідентифікатор бугалтерського рахунку;

– CompanyId – ідетифікатор компанії яка створила бухгалтерський рахунок;

– AccountNo – номер бухгалтерського рахунку;

– AcoountName – ім`я бухгалтерського рахунку;

– IsEnabled – чи ввімкнуто використання бухгалтерського рахунку;

– CreatedAt – дата та час створення бухгалтерського рахунку;

– CreatedById – ідентифікатор користувача, що створив бухгалтерський рахунок;

– LastUpdatedAt – дата та час останної зміни бухгалтерського рахунку;

– LastUpdatedById–ідентифікатор користувача, що останнім змінював бухгалтерський рахунок;

– поле CompanyId є зовнішнім ключем з таблиці Companies. Таким чином реалізується зв’язок між таблицями CompanyAccountsiCompanies, що визначений як один до багатьох: одна компанія може мати багато бухгалтерських рахунків, але один рахунок може мати лише одну компанію, що його створила.

Збереження данних про клієнтів компанії виконується таблицею CompanyCustomers за наступними полями:

– Id – ідентифікатор клієнта;

– CompanyId – ідентифікатор компанії, клієнтом якої є клієнт;

– CustomerId – глабільний ідентифікатор компанії для використання між різними модулями;

– CompanyCustomer – назва компанії;

– IsEnabled – чи ввімкнуто використання клієнта;

– CreatedAt – дата та час створення клієнта;

– CreatedById – ідентифікатор користувача, що додав клієта;

– LastUpdatedAt – дата та час останної зміни данних про клієнта;

– LastUpdatedById – ідентифікатор користувача, що останнім змінював дані про клієнта;

– поле CompanyId є зовнішнім ключем з таблиці Companies. Таким чином реалізується зв’язок між таблицями CompanyCustomers і Companies, що визначений як один до багатьох: одна компанія може матибагатоклієнтів, але один клієнт може відноситись лише до однієї компанії.

Збереження данних про користувацькі типи ассетів компанії виконується таблицею CompanyAssetTypes за наступними полями:

– Id – ідентифікатор користувацького типу ассетів;

– Name – назва користувацького типу ассетів;

– CompanyId – ідентифікатор компанії, що додала користувацький тип ассетів;

– IdWithinCompany – локальний ідентифікатор компанії;

– CreatedAt – дата та час створення користувацького типу;

– CreatedById – ідентифікатор користувача, що додав користувацький тип;

– LastUpdatedAt – дата та час останної зміни користувацького типу;

– LastUpdatedById – ідентифікатор користувача, що останнім змінював дані користувацького типу;

– DeletedAt – дата та час видалення користувацького типу;

– DeletedById – ідентифікатор користувача, що видалив користувацький тип;

– поле CompanyId є зовнішнім ключем з таблиці Companies. Таким чином реалізується зв’язок між таблицями CompanyAssetTypes і Companies, що визначений як один до багатьох: одна компанія може мати багато користувацьких типів ассетів, але один користувацький тип ассетів може відноситись лише до однієї компанії.

Збереження данних про користувацькі категорії ассетів компанії виконується таблицеюCompanyCategories за наступними полями:

– Id – ідентифікатор користувацької категорії;

– Name – назва користувацько категорії;

– CompanyId – ідентифікатор компанії, що додала користувацьку категорію;

– IdWithinCompany – локальний ідентифікатор користувацької категорії;

– HierarchyId – ідентифікатор головної категорії, до якої належить користувацька категорія;

– CreatedAt – дата та час створення користувацької категорії;

– CreatedById – ідентифікатор користувача, що додав користувацькукатегорію;

– LastUpdatedAt – дата та час останної зміни користувацької категорії;

– LastUpdatedById – ідентифікатор користувача, що останнім змінював дані користувацької категорії;

– DeletedAt – дата та час видалення користувацької категорії;

– DeletedById – ідентифікатор користувача, що видалив користувацьку категорію;

– поле CompanyId є зовнішнім ключем з таблиці Companies. Таким чином реалізується зв’язок між таблицями CompanyCategories і Companies, що визначений як один до багатьох: одна компанія може мати багато користувацьких категорій ассетів, але одна користувацька категорія ассетів може відноситись лише до однієї компанії.

Збереження данних про пари властивість-ассет та значення властивості-ассетвиконується таблицею CompanyAssetDimensions за наступними полями:

– AssetId – ідентифікатор ассету, до якого відноситься властивість;

– CompanyId – ідентифікатор компанії, що додала пару властивість-ассет та значення властивості-ассет;

– CompanyDimensionId – ідентифікатор властивості;

– CompanyDimesionElementId – ідентифікатор конткертного значення властивості;

– CreatedAt – дата та час створення пари властивість-ассет та значення властивості-ассет;

– CreatedById – ідентифікатор користувача, що додав пару властивість-ассет та значення властивості-ассет;

– LastUpdatedAt – дата та час останної зміни пари властивість-ассет та значення властивості-ассет;

– LastUpdatedById – ідентифікатор користувача, що останнім змінював дані пари властивість-ассет та значення властивості-ассет;

– поле AssetId є зовнішнім ключем з таблиці CompanyAssets. Таким чином реалізується зв’язок між таблицями CompanyAssetDimensions і CompanyAssets, що визначений як один до багатьох: це зроблено через те, що різні ассети можуть мати різні властивості і для оптимізацію таблиці CompanyAssets доцільніше винести ці відносини в проміжну таблицю;

– поле CompanyId є зовнішнім ключем з таблиці Companies. Таким чином реалізується зв’язок між таблицями CompanyAssetDimensions і Companies, що визначений як один до багатьох: одна компанія може мати багато пар властивість-ассет, але одна пара властивість-ассетможе відноситись лише до однієї компанії;

– поле CompanyDimensionId є зовнішнім ключем з таблиці CompanyDimensions. Таким чином реалізується зв’язок між таблицями CompanyDimensionsiCompanyAssetDimensions, що визначений як один до багатьох: одна властивість може використовуватись в різних парах ассет-власивість, але одна така пара може мати тільки одну властивість;

– поле CompanyDimesionElementId є зовнішнім ключем з таблиці CompanyDimesionElemens. Таким чином реалізується зв’язок між таблицями CompanyDimesionElemensiCompanyAssetDimensions, що визначений як один до багатьох: одне значення властивості може використовуватись в різних пара, але одна така пара може мати тільки одне значення властивості.

Збереження данних про властивості виконується таблицею CompanyDimensions за наступними полями:

– CompanyDimensionId – Ідентифікатор властивості;

– Name – назва властивості;

– CompanyId – ідентифікатор компанії, що додала властивість;

– CreatedAt – дата та час створення властивості;

– CreatedById – ідентифікатор користувача, що додав властивість;

– LastUpdatedAt – дата та час останної зміни властивості;

– LastUpdatedById – ідентифікатор користувача, що останнім змінював дані властивості;

– DeletedAt – дата та час видалення властивості;

– DeletedById – ідентифікатор користувача, що видалив властивість;

– поле CompanyId є зовнішнім ключем з таблиці Companies. Таким чином реалізується зв’язок між таблицями CompanyDimensions і Companies, що визначений як один до багатьох: одна компанія може мати багато властивостей для ассетів, але одна властивість може відноситись лише до однієї компанії.

Збереження данних про значення властивості виконується таблицею CompanyDimensionElements за наступними полями:

– CompanyDimensionId – Ідентифікатор властивості до якої відноситься значеня;

– CompanyDimensionElementId – ідентифікатор значення властивості;

– CompanyId – ідентифікатор компанії, що додала значеня властивості;

– Name – конкретне значення властивості;

– CreatedAt – дата та час створення значення властивості;

– CreatedById – ідентифікатор користувача, що додав значення властивості;

– LastUpdatedAt – дата та час останної зміни значення властивості;

– LastUpdatedById – ідентифікатор користувача, що останнім змінював дані значення властивості;

– DeletedAt – дата та час видалення значення властивості;

– DeletedById – ідентифікатор користувача, що видалив значення властивості;

– поле CompanyId є зовнішнім ключем з таблиці Companies. Таким чином реалізується зв’язок між таблицями CompanyDimensionElements і Companies, що визначений як один до багатьох: одна компанія може мати

багато значень властивостей для ассетів, але одне значення властивості може відноситись лише до однієї компанії;

– поле CompanyDimensionId є зовнішнім ключем з таблиці CompanyDimensions. Таким чином реалізується зв’язок між таблицями CompanyDimensionElements і CompanyDimensions, що визначений як один до багатьох: одна властивість може мати багато значень, але одне значення може відноситись тільки до однієї властивості.

Збереження данних про імортиассетів виконується таблицею CompanyAssetImports за наступними полями:

– Id – ідентифікатор імпорту ассетів;

– FileName – назва файлу з якого було виконано іморт;

– BlobName – назва блоб який було збережено;

– CompanyId – ідетифікатор компанії, що виконала імпорт ассетів;

– CreatedAt – дата та час іморту;

– CreatedById – ідентифікатор користувача, що виконав іморт;

– LastUpdatedAt – дата та час останної зміни імпорту;

– LastUpdatedById – ідентифікатор користувача, що останнім змінював дані імпорту;

– DeletedAt – дата та час видалення імпорту;

– DeletedById – ідентифікатор користувача, що видалив іморт;

– поле CompanyId є зовнішнім ключем з таблиці Companies. Таким чином реалізується зв’язок між таблицями CompanyAssetImports і Companies, що визначений як один до багатьох: одна компанія може багато раз імпортувати ассети, але один імпорт може бути виконаний тільки однією компанією.

Збереження данних про компанії виконується таблицею Companies за наступними полями:

– Id – Ідентифікатор компанії;

– CompanyName – назва компанії;

– Country – країна реєстрації компанії;

– CreatedAt – дата та час додавання компанії;

– CreatedById – ідентифікатор користувача, що додав компанію;

– LastUpdatedAt – дата та час останної зміни даних про компанію;

– LastUpdatedById – ідентифікатор користувача, що останнім змінював дані про компанію.

Створена схема структури БДпоказана на рис. 2.3.

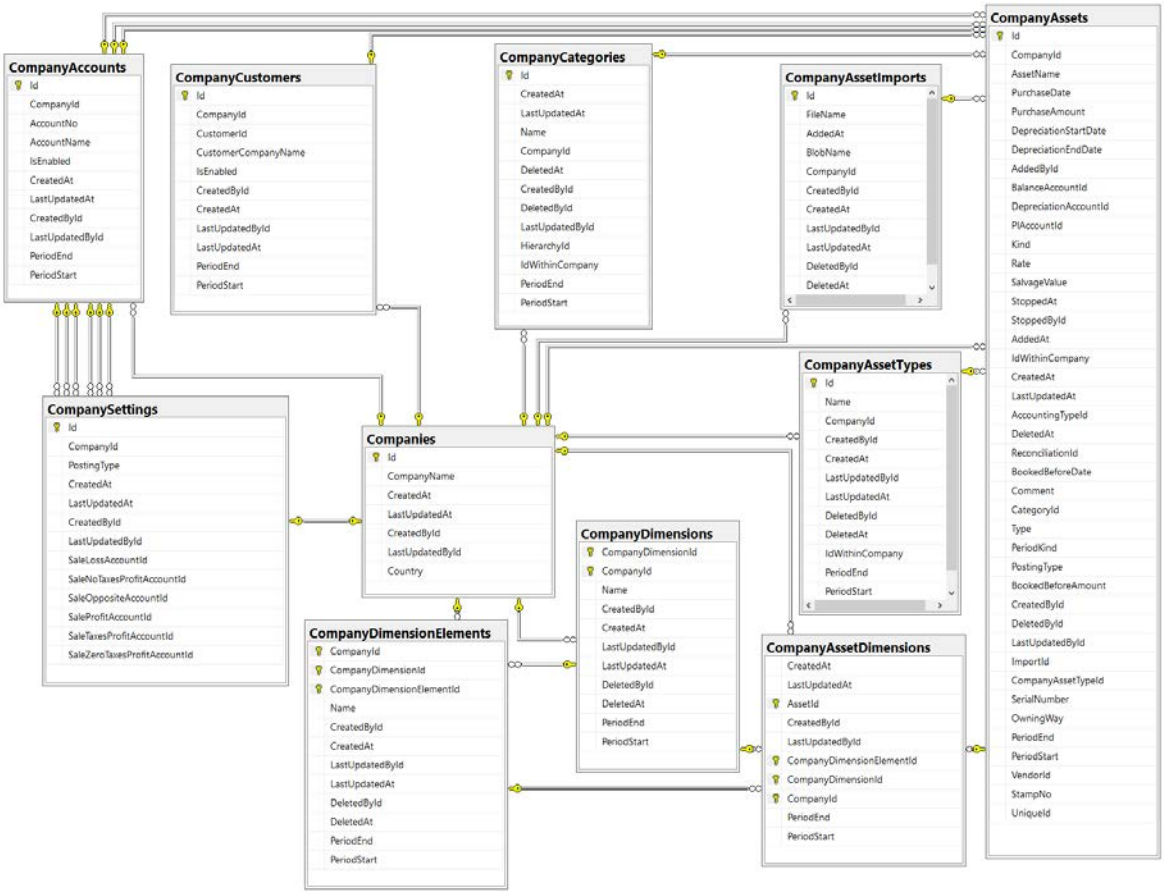

Рисунок 2.3 – Схема структури БД

2.4 Висновки за розділом 2

За результатами проєктування представлено результати порівняння фреймворків для розробки вебзастосунків мовою C# та TypeScript, обрано фреймворк Angular у цій якості, що визначило структуру програми на основі використання шаблону проєктування MVC, представлено функціональні

вимоги до програми, діаграму варіантів використання, схему бази даних і описано її відповідні таблиці і зв’язки між ними.

ЗРОЗРОБКА ПРОГРАМИ

3.1 Алгоритм функціонування програми

Процес роботи бухгалтера з програмою було уніфіковано для покращення орієнтування в програмі, оскільки вона має достатньо багато функціональних можливостей, які прості у використанні, проте потребують забезпечення доступу до них. Головним засобом керування доступом було вирішено визначити головну таблицю та додаткове меню (рис. 3.1).

Будь-який користувач у випадку традиційного звернення до вебсервера, тобто вказуючи загальний шлях до вебсервера, розпочинає свою роботу з процесу відображення таблиці ассетів. На ній можна переглянути данні про ассет, перейти до вікна зміни ассету, додати новий ассет, робити пошук по таблиці або ж перейти в додаткове меню. Далі в будь-який момент користувач може змінювати і вибирати інший напрямок дій.

Окремим розділом є напрямок дій, орієнтований на створення звітів на основі данних в таблиці, там можна обрати діапазон часу, за який створити звіт, відфільтрувати дані, за якими буде створено звіт, а також обрати формат файлу звіту для завантаження.

Ще одним з головних напрямков дії є створення нового ассету. Кнопка для переходу у модуль ствrення нового ассету знаходиться на головному вікні над таблицею. Вікно створення ассету розділене на 3 окремі тематичні вкладки, для полегшення орієнтації.

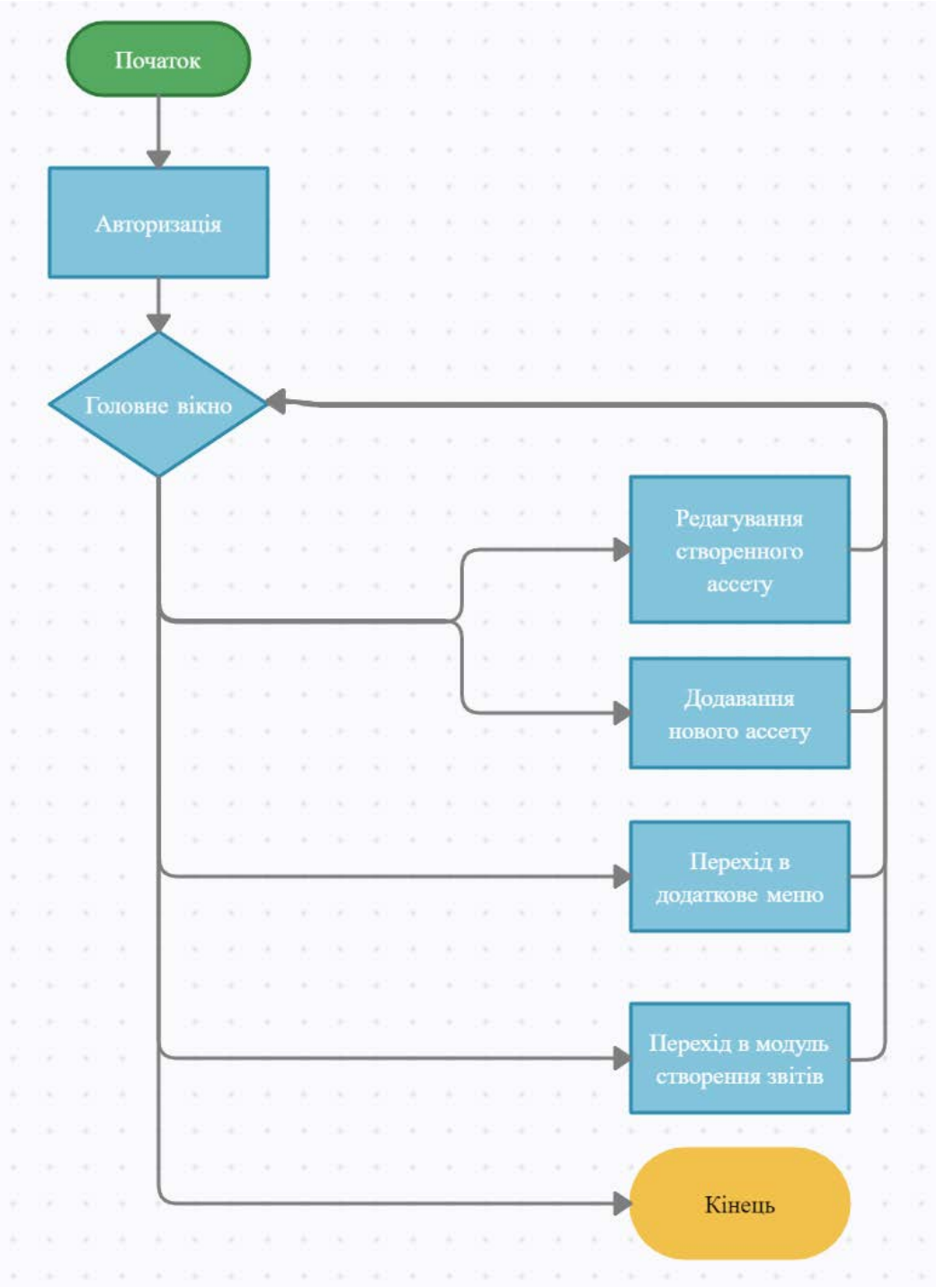

Рисунок 3.1 – Схема функціонування програми

3.2 Опис реалізованих класів програми

При реалізації програми за рахунок того, що було застосованоEntityFramework, вдалось зменшити час реалізації через уникнення подвійної розробки, коли створюється окремо БД і засоби роботи з класами, подібними до таблиць БД, в програмі.Це дозволило уникнути

створення класів моделей іполегшитиподальшу міграцію БД на основі цих класів з автоматичним створенням таблиць БД.

Клас AssetEditDto має таку програмно визначену структуру:

– AssetName - символьне поле, що визначає назву ассету;

– Type–зовнішній ключ з моделі AssetTypes, що визначає тип ассету;

– BalanceAccount – числове поле, що означає номер рахунку балансу;

– DepreciationAccount – числове поле, що означає номер рахунку втрати вартості;

– PLAccount – числове поле, що означає номер рахунку прибутку-збитку;

– SalvageValue – числове поле з плаваючою крапкою, що означає ліквідаційну вартість;

– PurchaseAmount - числове поле з плаваючою крапкою, що означає вартість ассету;

– DepreciationRate – числове поле з плаваючою крапкою, що означає процент втрати вартості при використанні метода проценту;

– DepreciationKind – значення з переліку, що означає тип методу підрахування втрати вартості;

– PeriodKind – значення з переліку, що означає день місяня початку втрати вартості;

– PostingType – значення з переліку, що означає тип постінгу;

– PurchaseDate – дата придбання ассету;

– DepreciationEndDate – дата закінчення втрати вартості;

– DepreciationStartDate – дата початку втрати вартості;

– Dimensions – масив властивостей та значень властивостей ассету;

– BookedBeforeDate – користувацька дата закінчення втрати вартості;

– BookedBeforeAmount – числове поле, що означаєкористувацьку ціну на початок втрати вартості;

– Category – числове поле, що означає категорію ассету;

– CompanyAssetType – числове поле, що означає користувацький тип ассету;

– SerialNumber – символьне поле, що означає серійний номер ассету;

– OwningWay – значення з переліку, що означає тип володіння ассетом;

– VendorId – числове поле, що означає ідентифікатор клієнта, що володіє ассетом;

– StampNo – числове поле, що означає номер штампу.

Клас CompanyTypeDto має таку програмно визначену структуру:

– Id – числове поле, що означає ідентифікатор користувацьго типу;

– Name – символьне поле, що означає назву користувацького типу.

Клас CustomerDto має таку програмно визначену структуру:

– Id – числове поле, що означає ідентифікатор компанії клієнта;

– Name – символьне поле, що означає назву компанії клієнта.

Клас DimensionDto має таку програмно визначену структуру:\

– Id – числове поле, що означає ідентифікатор властивості;

– Name – символьне поле, що означає назву властивості;

– Elements – массив, що зберігає в собі набір значень властивості.

Клас DimensionElementDto має таку програмно визначену структуру:

– Id – символьне поле, що означає ідентифікатор значення властивості;

– Name – символьне поле, що означає значення властивості.

Клас EditCategoryDto має таку програмно визначену структуру:

– Name – символьне поле, що означає назву категорії.

Клас EditCompanyTypeDto має таку програмно визначену структуру:

– Name – символьне поле, що означає назву користувацького типу.

Клас EditDimensionDto має таку програмно визначену структуру:

– DimensionId – числове поле, що означає ідентифікатор властивості;

– ElementId – символьне поле, що означає ідентифікатор значення властивості.

Клас ImportDto має таку програмно визначену структуру:

– Id – числове поле, що означає ідентифікатор імпорту;

– FileName – символьне поле, що означає назву файлу імпорту;

– AddedAt – дата виконная імпорту ассетів.

Клас SaleAccountsDto має таку програмно визначену структуру:

– TaxesProfit – числове поле, що означає рахунок прибуту з урахуванням податків;

– ZeroTaxesProfit – числове поле, що означає рахунок прибутку з відсутністю податків;

– NoTaxesProfit – числове поле, що означає рахунок прибутку без урахування податків;

– Opposite – числове поле, що означає рахунок обороту;

– Profit – числове поле, що означає рахунок прибутку;

– Loss – числове поле, що означає рахунок збитків.

Клас SaleParametersDto має таку програмно визначену структуру:

– Type – значення з переліку, що означає тип продажу ассету;

– Date – дата продажу ассету;

– Amount – числове поле, що означає ціну, за яку було продано ассет;

– CustomerId – числове поле, що означає ідентифікатор клієнта, якому було продано ассет;

– AccountNo – числове поле, що означає номер рахунку на який було зараховано кошти з продажу;

– SalePeriodReversed – булівське поле, що означає чи було відмінено продаж ассету.

КлассBookingFactory має таку програмно визначену структуру:

– метод CreateBookings – на основі параметрів викликає необхідний метод для підрахунку втрати вартості на кожниій період;

– метод SumOfTheYearDigits – один з методів підрахунку втрати вартості. Вираховує втрату вартості на основі значення втрати вартості на 1 рік;

– метод DecliningBalance – один з методів підрахунку втрати вартості. Вираховує втрату вартості на основі проценту втрати, що вказаний в ассеті;

– метод StraightLine – один з методів підрахунку втрати вартості. Вираховує втрату вартості рівномірно розподілену на період часу.

КлассAccountLoader має таку програмно визначену структуру:

– tfsoApiFactory – ін`єкція сервісу побудови API;

– метод LoadAsync – завантаження списку рахунків.

Клас AssetsComponent має таку програмно визначену структуру:

– assets$ – асінхронне поле з типом Observable, що зберігає в собі масив ассетів;

– columns$ – асінхронне поле з типом Observable, що зберігає в собі масив колонок таблиці;

– flatCategories$ – асінхронне поле з типом Observable, що зберігає в собі масив об`єктів категорій з дочерніми категоріями;

– companyTypes$ – асінхронне поле з типом Observable, що зберігає в собі масив користувацьких типів;

– moreFilters$ – асінхронне поле з типом Observable, що зберігає в собі логічне значення, яке відповідає за відображення додаткових полей фільтрації;

– customers$ – асінхронне поле з типом Observable, що зберігає в собі масив клієнтів компанії;

– country$ – асінхронне поле з типом Observable, що зберігає в собі країну компанії.

Клас EditAssetFormComponent має таку програмно визначену структуру:

– flatCategories$ – асінхронне поле з типом Observable, що зберігає в собі масив об`єктів категорій з дочерніми категоріями;

– companyTypes$ – асінхронне поле з типом Observable, що зберігає в собі масив користувацьких типів;

– customers$ – асінхронне поле з типом Observable, що зберігає в собі масив клієнтів компанії;

– country$ – асінхронне поле з типом Observable, що зберігає в собі країну компанії.

Клас EditCompanyTypesFormComponent має таку програмно визначену структуру:

– types$ – асінхронне поле з типом Observable, що зберігає в собі масив користувацьких типів.

Клас ImportAssetsDialogComponent має таку програмно визначену структуру:

– selectedFiles$ – асінхронне поле з типом Observable, що зберігає масив імортованих файлів;

– history$ – асінхронне поле з типом Observable, що зберігає масив ассетів в іпмортованихфайлах.

Клас ReportsComponent має таку програмно визначену структуру:

– flatCategories$ – асінхронне поле з типом Observable, що зберігає в собі масив об`єктів категорій з дочерніми категоріями;

– companyTypes$ – асінхронне поле з типом Observable, що зберігає в собі масив користувацьких типів;

– assets$ – асінхронне поле з типом Observable, що зберігає в собі масив ассетів;

– balanceAccounts$ – асінхронне поле з типом Observable, що зберігає в собі масив бухгалтерських аккаутнів компанії.

Компонентassets.component.html виконує підготовку даних дляперегляду таблиці ассетів та говоного вікна з навігаційним меню, пошуком по таблиці, та додаванням нового ассета.

Компонентedit-asset-form.component виконує підготовку даних для редагування та створенняассету, використовуючи данні зі state. Додавання та редагування зберігається за допомогою відповідних POST-запитів, що включають усі поля ассету.

Компонент edit-categories-form.component.html виконує підготовку даних для перегляду, редагування, створення або видалення категорії використовуючи данні зі state. Додавання, видалення, редагування виконуються за допомогую відповідних POST-запитів, що включають в себе ім`я категорії name та id категорії.

Компонентedit-company-types-form.component.html виконує підготовку даних для перегляду, редагування, створення та видалення користувацього типу використовуючи дані зі state. Додавання, видалення, редагування виконуються за допомогою відповідних POST-запитів, що включать в себе ім`я користувацького типу name та id користувацького типу.

Компонент reports.component.html виконує підготовку данних для побудови звіту на основі даних з таблиці ассетів.

Нижче представлено діаграму класів проекту (рис. 3.2)

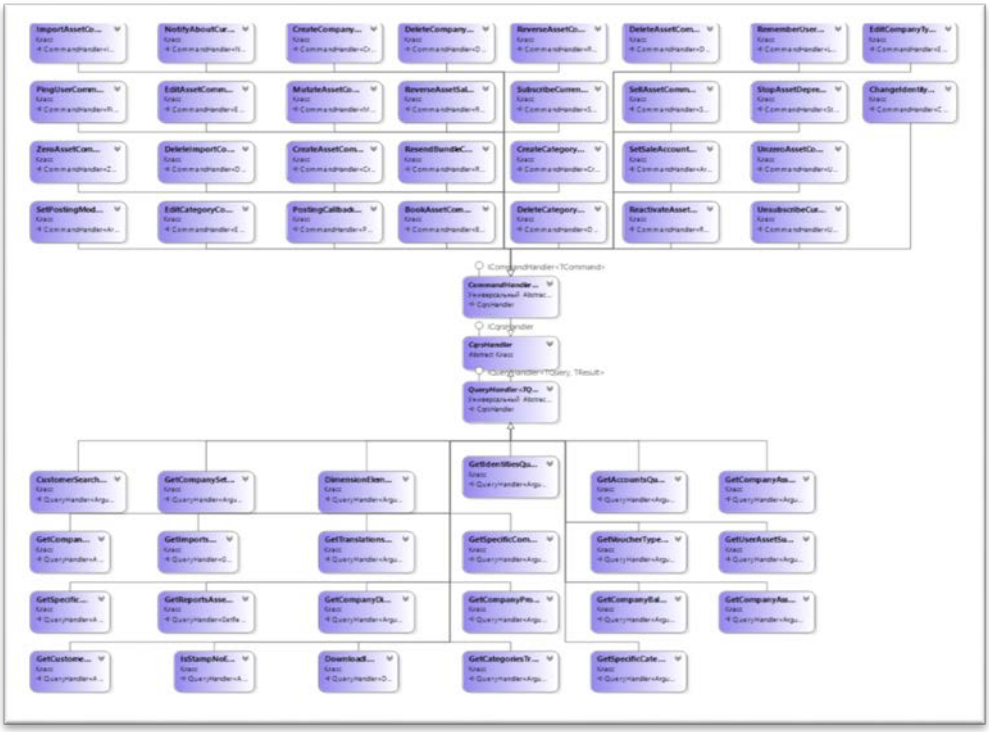

Рисунок 3.2 – Діаграма класів

3.3 Тестування програми

Тестування програми дозволило перевірити роботу програми щодо виконання функціональних можливостей, які передбачались у процесі розробки. Тестування спочатку виконувалось під час реалізації цих можливостей і дозволяло виявляти проблеми і вирішувати їх. У подальшому, після того, як всі такі проблеми було усунуто, було проведено окремий етап тестування з перевіркою виконання всіх функцій, тому успішність реалізації була підтверджена ставить Разделза результатами тестування на персональному комп’ютері під керуванням операційної системи Windows 10. Та за допомогою unit-тестування, інтеграційного тестування та тестування збірки.

3.4 Висновки за розділом 3

За результатами реалізації програми подано схему її функціонування, детально описано створені одиниці програми, включаючи класи моделей і компонентів, принцип їх роботи і організації, та загальну логіку роботи користувача, який виступає фактично бухгалтером, з програмним забезпеченням.

4.КЕРІВНИЦТВО ПРОГРАММІСТА

4.1 Загальні відомості про програму

Пограмапризначенна для спрощення обліку матеріальних активів компанії.

Додаток було створено за допомогою середовищ розробки JetbrainsRider, JetbrainsWebStormтаJetbrainsIntellij IDEA.

Базу даних було створено за допомогою EntityFramework.

Підключення таблиць бази данних на форму проекту будл виконано за допомогою EntytuFrameworkта HTTP.

Підключення до бази даних було організоване за допомогою строки з`єднання.

4.2 Структура програми

Програма складається з трьох проектів:

–Assets.WebAPi – проект бекендуна базі Asp.Net CoreWeb API, що містить в собі основні контроллери та функції програми, створення БД, міграції та допоміжні бібліотеки;

–Assets.WebApi.Tests – проект з тестуванням програми, що містить в собі XUnitтести, інтеграційни тести та тести збірки програми;

–Frontend – проект на базі Angular 14, що містить в собі основні компоненти програми з відображення даних.

Зв`язок між проектами Assets.WebAPIтаFrontendвідбувається за допомогою HTTPпротоколу.

4.3 Налагодження програми

Запуск програми відбувається на будь-якій операційній системі, що має підтримку бібліотеки пакетів dot-net. Перед запуском необхідно

використати файл prelaunch.bat, що автоматично вставить усі необхідні для роботи програми бібліотеки, та налаштує середовище перед запуском. Назви та версії усіх необхідних бібліотек описані у файлі package.jsonв обох проектах.

Щоб запустити програму потрібно окремо запустити проекти Assets.WebApitaFrontend. Проект Assets.WebApізапускається за допомогою консольної команди dot-netstart. Запуск проекту Frontendвідбувається за допомогою команди ngstart.

5КЕРІВНИЦТВО ОПЕРАТОРА

5.1 Призначення програми

Облік матеріальних ресурсів компанії та відстеження втрати їх вартості, продаж майна або ж придбання нового, створення звітів є одними з головних обов`язків бухгалтера компанії.

Загалом призначенням програми є створення інструменту для обліку матеріальних ресурсів компанії з відстеженням втрати їх вартості з можливісту сортування головної таблиці для біль швидкого та зручного пошуку, та побудови звітів для подальшого їх використання в компанії.

5.2Умови виконання програми

Мінімальний рівень технічних засобів для забезпечення роботи програмної системи включає процесор не менш ніж 4 ядра і частотою 2,9 Ггц, оперативну пам'ять у 16 Гб, жорсткий диск у 120Гб.

5.3 Експлуатація програми

Робота користувача з програмою є достатньо простою. Кожен користувач має абсолютно однаковий інтерфейс, проте не всі функції йому можуть бути доступними. Для того щоб отримати доступ до системи , йому потрібно авторизуватися.

Одним із перших і головних вікон програми є головне вікно з навігаційним меню, таблицею ассетів з можливістю фільтрації.

Першою кнопкою, є кнопка натискаючи яку користувач може додати новий ассет (рис. 5.1).

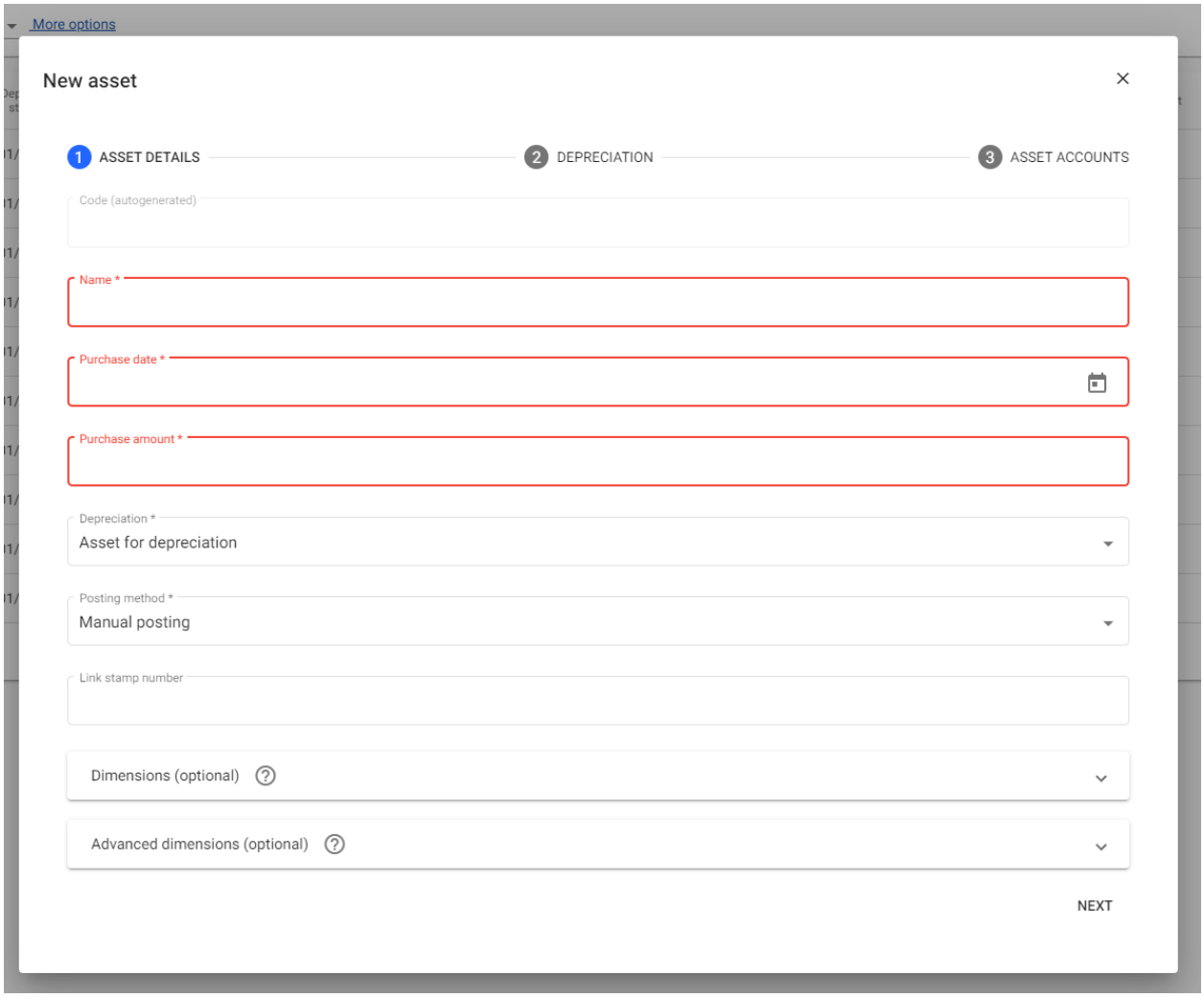

Рисунок 5.1 – Вікно додавання ассету

На будь-який елемент таблиці можна натиснути і перейти до його редагування або ж просто детально передивитись усі поля ассету. Не всі поля асету можна редагувати після його створення, адже деякі поя вираховуються на основі значень з інших. Вікно редагування(рис. 5.2). має майже такий самий вигляд як і вінко додавання ассету, але має деякі додаткові функції, як от продаж ассету або ж перегляд історії змін ассету.

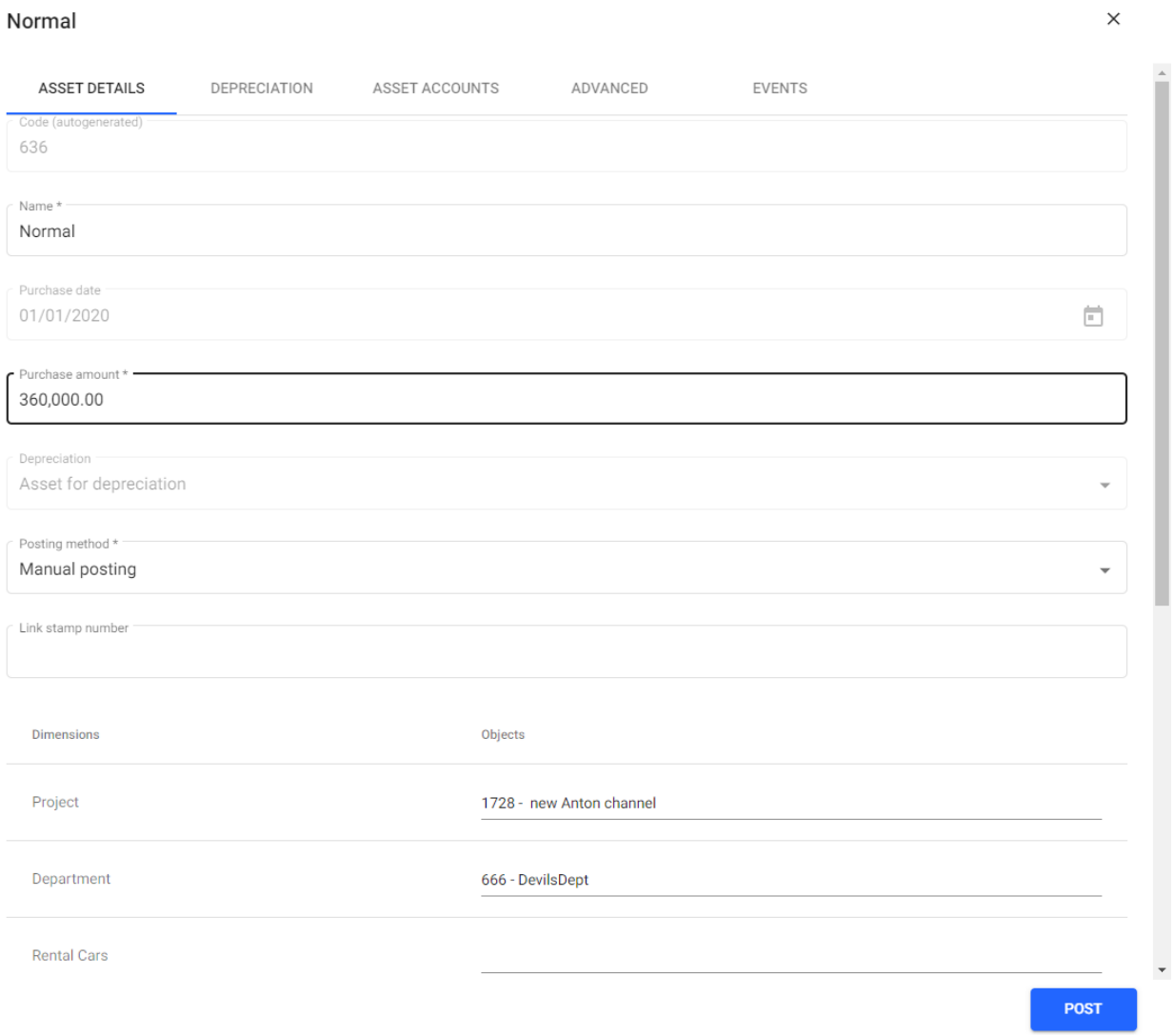

Рисунок 5.2 – Вікно редагування ассету

Натискаючи на кнопку додаткового меню можно побачити декілька нових пунктів:

– експорт даних з таблиці у файл формату csv;

– вікно імпорту ассетів з файлу;

– вікно налаштувань в якому можна редагувати категорії та користувацькі типи.

Вікно імпорту ассетів з файлу представлено у вигляді вікна з трьома вкладками, перша з яких відповідає за додаванян файлу для імпорту з подальшою його валідацією, друга показує історію імпортованих файлів з

моєливістю видалити їх (рис.5.3), трете зберігає в собі приклади файлів для імпорту, та правила для побудови таких файлів.

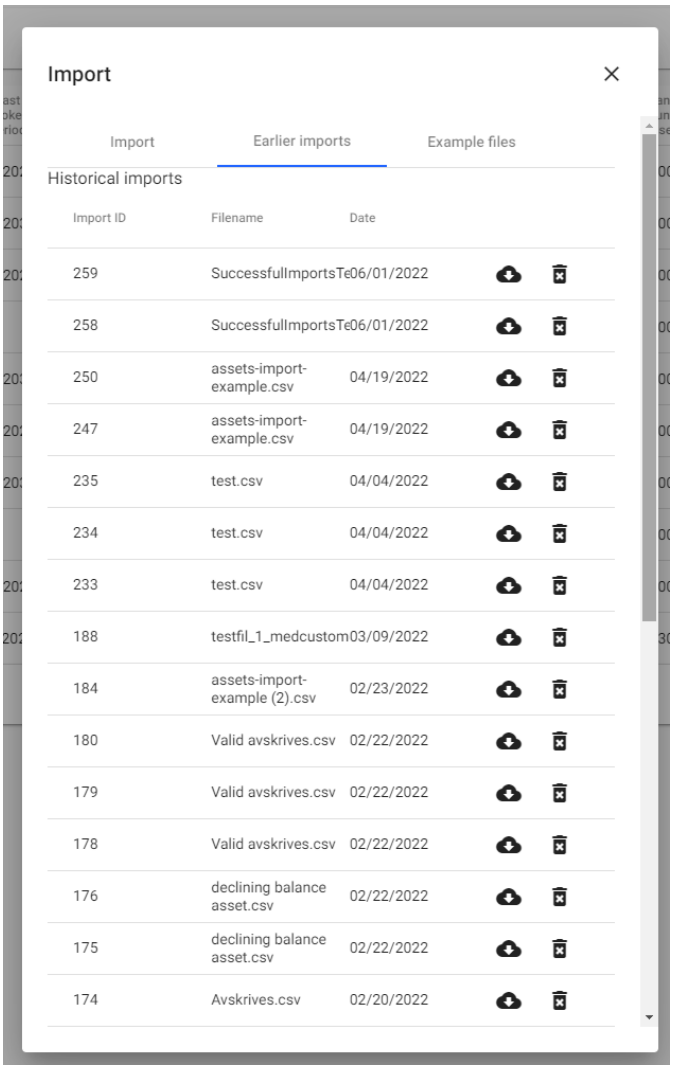

Рисунок 5.3 – Список минулих імпортів

Наступне вікно є вікном додавання, перегляду, редагування та видалення категорій(рис. 5.4). Тут можна переглянути підкатегорії для конкретної категорії, видалити категорію, якщо вона не використовується в будь-якому з ассетів, змінити ім`я категорії, або ж додати нову категорію.

Майже такий самий вигляд та функціонал має вікно редагування користувацьких типів ассетів (рис. 5.5). Тут можна переглянути користувацькі типи, змінити ім`я користувацького типу, видалити його або додати новий.

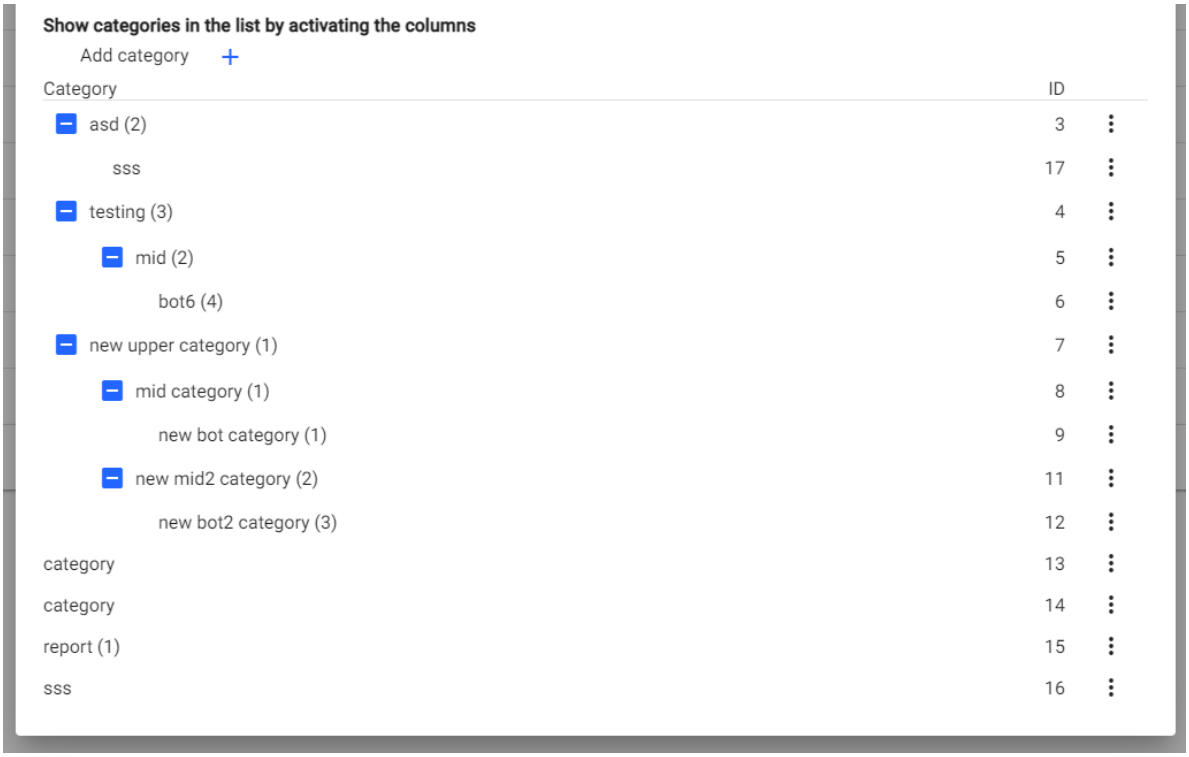

Рисунок 5.4 – Вікно редагування категорій

Рисунок 5.5 – Вікно редагування користувацьких типів

Наступне вікно відкривається за допомогою кнопки “AssetReport” та відкриває модуль звітів (рис. 5.6). В ньому можносворити звіт на основі данних з таблиці ассетів, вибрати параметри за якими створити звіт, та

завантажити його у форматах pdfta xlsx. Звіт показує прибуток та збиток по кожному з рахунків компанії та загальний результат по всім рахункам.

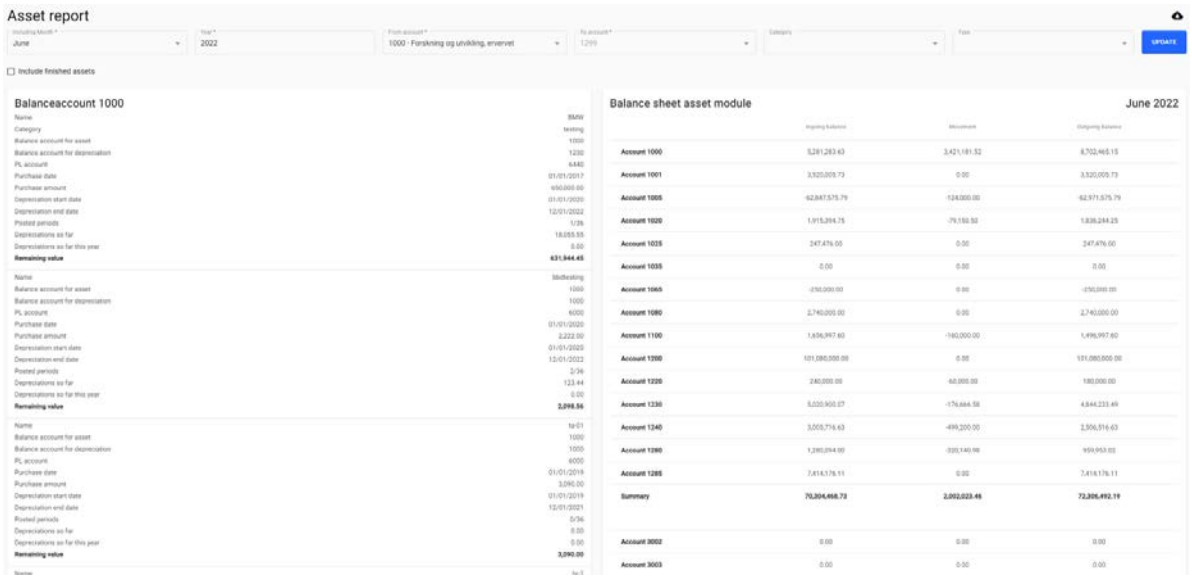

Рисунок 5.6 – Вікно створення звітів

5.4 Висновки за розділом 5

Описано процес експлуатації програми користувачем, надано порядок і засоби доступу до його основних функцій. Реалізовано тестування, що підтвердило успішність реалізації програми загалом.

ВИСНОВКИ

Метою роботи було розробити програмне забезпечення обліку матеріальних ресурсів компанії з відстежуванням втрати їх вартості та побудови звітів на основі цих даних, що було досягнуто в результаті. Для цього було досягнуто наступні результати за відповідними, встановленими спочатку, завданнями.

За результатами аналізу проблеми виконано аналіз аналогів і їх порівняння з розробкою, визначено її особливості, що стосуються загалом обліку матеріальних активів компанії з відстежування втрати їх вартості та побудови звітів на основі цих даних.

За результатами проєктування представлено результати порівняння фреймворків для розробки вебзастосунків мовою TypeScriptта C#, обрано фреймворк Angular у цій якості, що визначило структуру програми на основі використання шаблону проєктування MVC, представлено функціональні вимоги до програми, діаграму варіантів використання, схему бази даних і описано її відповідні таблиці і зв’язки між ними.

За результатами реалізації програми подано схему її функціонування, детально описано створені одиниці програми, включаючи класи моделей і уявлень, та загальну логіку роботи користувача, який виступає бухгалтером компанії, з програмою. Під час виконання розробки та за результатами її виконання реалізовано тестування програми, що дозволило отримати результат, який відповідає запланованим для реалізації у технічному завданні вимогам.

Розроблена програмадозволила в результаті створити інструмент для зручного та швидкого обліку матеріальних ресурсів компанії бухгалтером зі зручною категорізацією, пошуком за різними параметрами, відстеженням втрати вартості активів різними методами, відстеження залишків, прибутку та збитку по ріхних бухгалтерських рахунках. Також програма може використовуватись для побудови звітів по бухгалтерських рахунках.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Майновий стан, облік майна суб`єктів господарювання [Електронний ресурс]. – Режим доступу: https://pidru4niki.com/1924070148642/pravo/maynoviy_stan_oblik_mayna_subyektiv_gospodaryuvannya_otsinka_mayna_maynovih_prav

2. Програма 1С : ERP [Електронний ресурс]. – Режим доступу: https://v8.1c.ua/erp/

3. Microsoft Dynamic 365 [Electronic resource]. – Access mode: https://dynamics.microsoft.com/en-us/

4. Парус: Підприємство [Електронний ресурс]. – Режим доступу: http://www.parus.ua/ua/78/

5. TypeScript [Electronic resource]. – Access mode: https://www.typescriptlang.org/

6. Ріхтер Джеффрі Курс по C# від всесвітньо відобого розробника застосунків / Джеффрі Ріхтер // CLRvia C# (Developer Reference) 4th Edition. – 2012. – 894 с.

7. Framework Angular [Electronic resource]. – Access mode: https://angular.io/

8. Документація по React library [Електронний ресурс]. – Режим доступу: https://uk.reactjs.org/

9. ASP.Net Core documentation [Electronic resource]. – Access mode: https://docs.microsoft.com/en-us/aspnet/core/?view=aspnetcore-6.0

10. Blazor for ASP.NET Core documentation [Electronic resource]. – Access mode: https://docs.microsoft.com/en-us/aspnet/core/blazor

11. Component design [Electronicresource]. – Access mode: https://gameprogrammingpatterns.com/component.html

12. MVC Pattern [Electronic resource]. – Access mode: https://en.wikipedia.org/wiki/Model-View-Controller

13. Entity Framework documentation [Electronic resource]. – Access

mode: https://docs.microsoft.com/en-us/ef/

14. Database design approaches for Entity Framework [Electronic resource]. – Access mode: https://www.c-sharpcorner.com/blogs/code-first-vs-database-first-vs-model-first-approach\

15. Code First to a New Database using Entity Framework [Electronic resource]. – Access mode: https://docs.microsoft.com/en-us/ef/ef6/modeling/code-first/workflows/new-database

ДОДАТОК А

Технічне завдання

Вступ

Майно організації дозволяє їй вести господарську діяльність та отримувати дохід. Тому ведення бухгалтерського обліку та облік майна підприємства нерозривно пов'язані. Організація обліку майна на підприємстві передбачає створення системи зі збору інформації про майнове становище підприємства з її подальшим аналізом. Саме на це і має бути направлено створення програми в даній роботі.

А.1 Підстави для розробки

Розробка в межах дипломної кваліфікаційної роботи реалізується на підставі наказу№ 102 від 27 квітня 2022 р. за Національним університетом «Запорізька політехніка» за темою Розробка програми веб застосунку «Облік майна компанії».

А.2 Призначення розробки

Призначенням розробки є створення вебзастосунку, що полегшує ведення обілку майна компанії з підрахунком втраченої вартості а також формування звітів на основі цих даних.

А.3 Основні вимоги до програми

А.3.1 Вимоги до функціональних характеристик

Функціональні характеристики, які повинна мати програма, охоплюють:

- перегляд таблиці ассетів;
- додавання нових ассетів;
- редагування існуючих ассетів;

– перегляд, додавання, редагування та видалення користувацьких типів;

– перегляд, додавання, редагування та видалення користувацьких категорій;

– імпорт ассетів з файлу;

– експорт ассетів у файл;

– продаж ассетів;

– пошук ассетів у таблиці;

– створення звіту на основі інформації в таблиці ассетів;

– експорт звіту у формати xlsx та pdf.

А.3.2 Умови експлуатації

Експлуатація програми має забезпечуватися з одного боку звичайними користувачами, які працюють з даними, наповнюють їх, а з іншого боку – адміністраторами системи та БД.

А.3.3 Вимоги до складу та параметрів технічних засобів

Мінімальний рівень технічних засобів для забезпечення роботи програмної системи включає процесор не менш ніж 2 ядра і частотою 2,9 Ггц, оперативну пам'ять у 4Гб, жорсткий диск у 8Гб.

А.4 Порядок контролю та приймання

Для контролю виконання роботи використовується календарний план, а приймання роботи виконується під час проведення процедури захисту роботи перед екзаменаційною комісією.

ДОДАТОК Б

Текст програми

Б.1 Текст файлаassets.component.ts

```
import { FormBuilder, FormControl } from '@angular/forms';
import {
Asset,
AssetStatus,
AssetSubStatus,
CategoryNode,
CompanyType,
Customer,
FlatNode,
Languages,
StatefulColumn
} from 'app/core/utilities/types/assets.types';
import { profileLocaleLanguage } from
'app/store/authentication/authentication.selectors';
import { isEqual } from 'lodash-es';
import { distinctUntilChanged, first, map, startWith, take, tap
} from 'rxjs/operators';
import { addStampNo, commentAsset, initAssetsLoading } from
'./reducers/assets/assets.actions';
import { assetsArray, isLoadingAssetsAndMetaOrChangingIdentity }
from './reducers/assets/assets.selectors';
import { getAssetFilter } from
'./reducers/filter/filter.selectors';
import {
checkSubscriptionStatus,
subscribeToNotifications,
unsubscribeFromNotifications
} from './reducers/subscription-status/subscription-
status.actions';
import { EditAssetDialogComponent } from
'@fromScreens/assets/edit-asset-dialog/edit-asset-dialog.component';
import {
isLoadingSubscriptionStatus,
isSubscribedToNotifications
} from '@fromScreens/assets/reducers/subscription-
status/subscription-status.selectors';
import { currentPageAssets } from
'@fromScreens/assets/reducers/page/page.selectors';
import { AfterViewInit, ChangeDetectorRef, Component, NgZone,
OnDestroy, OnInit } from '@angular/core';
import { AssetFilter } from
'@fromScreens/assets/reducers/filter/filter.reducer';
```

```
      import {
      changeFilterCategory,
      changeFilterCompanyType,
      changeFilterCustomer,
      changeFilterSerialNumber,
      changeFilterStatus,
      changeFilterSubStatus,
      changeFilterText
      } from '@fromScreens/assets/reducers/filter/filter.actions';
      import { MatDialog } from '@angular/material/dialog';
      import { TranslationUtilityService } from
'@fromShared/services/translation-utility.service';
      import { combineLatest, Observable, Subscription } from 'rxjs';
      import { ImportAssetsDialogComponent } from
'@fromScreens/assets/import-assets-dialog/import-assets-
dialog.component';
      import { assetsDialogConfig, ExtendedAssetColumn, formatCustomer
} from '@fromScreens/assets/utils';
      import {
      filteredColumns,
      isDoingSomethingWithColumns
      } from '@fromStore/settings/company/reducers/client/client-side-
company-preferences.selectors';
      import { setMoreFilters } from
'@fromScreens/assets/reducers/more-filters/more-filters.actions';
      import { CommentDialogComponent } from
'@fromScreens/assets/comment-dialog/comment-dialog.component';
      import { moreFilters } from '@fromScreens/assets/reducers/more-
filters/more-filters.selectors';
      import { flatCategoriesArray } from
'@fromShared/reducers/categories/categories.selectors';
      import { AssetsCsvExportService } from
'@fromScreens/assets/services/assets-csv-export.service';
      import { MoreFiltersState } from
'@fromScreens/assets/reducers/more-filters/more-filters.reducers';
      import { EditSettingsDialogComponent } from
'@fromScreens/assets/edit-settings-dialog/edit-settings-
dialog.component';
      import { switchColumn } from
'@fromStore/settings/company/reducers/client/client-side-company-
preferences.actions';
      import { onlyTruthy } from '@coreUtils/only-truthy';
      import { companyTypesArray } from '@fromShared/reducers/company-
types/company-types.selectors';
```

```
      import { customerArray } from
'@fromScreens/assets/reducers/customers/customers.selectors';
      import { BatchPostDialogComponent } from
'@fromScreens/assets/batch-post-dialog/batch-post-dialog.component';
      import { environment } from '@env/environment';
      import { isHistoricVersion } from
'@fromShared/reducers/version/version.selectors';
      import { ApplicationStore } from '@fromRoot/data/application-
store.service';
      import { v4 } from 'uuid';
      import { StampnoDialogComponent } from
'@fromScreens/assets/stampno-dialog/stampno-dialog.component';

      @Component({
      selector: 'app-assets',
      templateUrl: './assets.component.html',
      styleUrls: ['./assets.component.scss']
      })
      exportclassAssetsComponentimplementsOnInit, OnDestroy,
AfterViewInit {
      readonlyassets$: Observable<readonlyAsset[]>;
      readonlycolumns$: Observable<StatefulColumn[]>;
      readonlyisLoading$: Observable<boolean>;
      readonlyflatCategories$: Observable<FlatNode<CategoryNode>[]>;
      readonlyisDoingSomethingOnClientSide$: Observable<boolean>;
      readonlycompanyTypes$: Observable<CompanyType[]>;
      readonlymoreFilters$: Observable<MoreFiltersState>;
      readonlyisHistoricVersion$: Observable<boolean>;
      readonlyperiodSwitcherEnabled =
environment.featureSwitches.periodSwitcher;
      readonlycustomers$: Observable<Customer[]>;
      filteredCustomers$: Observable<Customer[]>;
      country$: Observable<string>;

      isLoadingSubscription$: Observable<boolean>;
      isSubscribed$: Observable<boolean>;
      status$: Observable<AssetStatus>;
      text$: Observable<string>;
      statuses: AssetStatus[] = [
      AssetStatus.Ongoing,
      AssetStatus.NotStarted,
      AssetStatus.Inventory,
      AssetStatus.Done,
      AssetStatus.Canceled,
      AssetStatus.Any
```

```
        ];
      subStatuses: AssetSubStatus[] = [AssetSubStatus.Booked,
AssetSubStatus.NotBooked];
      statusControl: FormControl = newFormControl(AssetStatus.Any);
      subStatusControl: FormControl = newFormControl(null);
      textControl: FormControl = newFormControl('');
      categoryControl: FormControl = newFormControl(null);
      companyTypeControl: FormControl = newFormControl(null);
      serialNumberControl: FormControl = newFormControl('');
      customerControl: FormControl = newFormControl(null);

      privatereadonlysubscription = newSubscription();

      constructor(
      privatereadonlydialog: MatDialog,
      privatereadonlyformBuilder: FormBuilder,
      privatereadonlyappStore: ApplicationStore,
      privatereadonlycsvExportService: AssetsCsvExportService,
      privatereadonlychangeDetectorRef: ChangeDetectorRef,
      privatereadonlyngZone: NgZone,
      privatereadonlytranslationService: TranslationUtilityService
        ) {
      conststore = appStore.raw;
      this.isHistoricVersion$ = store.select(isHistoricVersion);
      this.columns$ = store.select(filteredColumns);
      this.assets$ = store.select(assetsArray).pipe(
      onlyTruthy(),
      tap(() =>this.changeDetectorRef.markForCheck())
          );
      this.isLoading$ =
store.select(isLoadingAssetsAndMetaOrChangingIdentity).pipe(distinctUn
tilChanged());
      this.isLoadingSubscription$ =
store.select(isLoadingSubscriptionStatus).pipe(distinctUntilChanged())
;
      this.isSubscribed$ =
store.select(isSubscribedToNotifications).pipe(distinctUntilChanged())
;
      this.flatCategories$ =
store.select(flatCategoriesArray).pipe(distinctUntilChanged());
      this.companyTypes$ =
store.select(companyTypesArray).pipe(distinctUntilChanged());
      this.isDoingSomethingOnClientSide$ =
store.select(isDoingSomethingWithColumns).pipe(distinctUntilChanged())
;
```

```
      this.moreFilters$ =
store.select(moreFilters).pipe(distinctUntilChanged());
      this.customers$ =
store.select(customerArray).pipe(distinctUntilChanged());
      this.filteredCustomers$ = combineLatest([
      this.customers$,
      this.customerControl.valueChanges.pipe(startWith(''))
          ]).pipe(
      map(([customers, value]) => {
      if (!value) {
      returncustomers;
              }
      returncustomers.filter((x)
=>formatCustomer(x).includes(value.toLowerCase()));
            })
          );
        }

        _(translationKey: string): string {
      returnthis.translationService._(translationKey);
        }

      refresh(): void {
      this.appStore.raw.dispatch(initAssetsLoading());
        }

      ngOnInit(): void {
          //this.refresh();
      conststore = this.appStore.raw;
      store.dispatch(checkSubscriptionStatus.request(undefined));
      this.country$ = store.select(profileLocaleLanguage);
      this.status$ = store.select(getAssetFilter).pipe(
      map((x): AssetStatus =>x.status),
      distinctUntilChanged(isEqual)
          );

      this.text$ = store.select(getAssetFilter).pipe(
      map((x): string =>x.text),
      distinctUntilChanged(isEqual)
          );

      this.subscription.add(
      this.statusControl.valueChanges
              .pipe(distinctUntilChanged(isEqual))
```

```
            .subscribe((status)
=>store.dispatch(changeFilterStatus({ status })))
          );

      this.subscription.add(
      this.textControl.valueChanges
              .pipe(distinctUntilChanged(isEqual))
              .subscribe((text) =>store.dispatch(changeFilterText({
text })))
          );

      this.subscription.add(
      this.serialNumberControl.valueChanges
              .pipe(distinctUntilChanged(isEqual))
              .subscribe((serialNumber)
=>store.dispatch(changeFilterSerialNumber({ serialNumber })))
          );

      this.subscription.add(
      this.subStatusControl.valueChanges
              .pipe(distinctUntilChanged(isEqual))
              .subscribe((subStatus)
=>store.dispatch(changeFilterSubStatus({ subStatus })))
          );

      this.subscription.add(
      this.categoryControl.valueChanges
              .pipe(distinctUntilChanged(isEqual))
              .subscribe((category)
=>store.dispatch(changeFilterCategory({ category })))
          );

      this.subscription.add(
      this.companyTypeControl.valueChanges
              .pipe(distinctUntilChanged(isEqual))
              .subscribe((assetCompanyType)
=>store.dispatch(changeFilterCompanyType({ assetCompanyType })))
          );

      this.subscription.add(
      this.customerControl.valueChanges
              .pipe(distinctUntilChanged(isEqual))
              .subscribe((customer)
=>store.dispatch(changeFilterCustomer({ customer })))
          );
```

```
this.subscription.add(
store
        .select(getAssetFilter)
        .pipe(distinctUntilChanged(isEqual))
        .subscribe((x: AssetFilter) => {
this.statusControl.setValue(x.status);
this.subStatusControl.setValue(x.subStatus);
this.textControl.setValue(x.text);
this.categoryControl.setValue(x.category);
this.companyTypeControl.setValue(x.assetCompanyType);
this.serialNumberControl.setValue(x.serialNumber);
this.customerControl.setValue(x.customer);
        })
    );
  }

showMoreFilters(more: boolean): void {
this.appStore.raw.dispatch(setMoreFilters({ enabled: more }));
  }

ngOnDestroy(): void {
this.subscription.unsubscribe();
  }

statusStringFor(status: AssetStatus): string {
switch (status) {
caseAssetStatus.Any:
return 'all2';
caseAssetStatus.NotStarted:
return 'not_started';
caseAssetStatus.Ongoing:
return 'ongoing';
caseAssetStatus.Inventory:
return 'assets.tabs.inventory';
caseAssetStatus.Done:
return 'finished';
caseAssetStatus.Canceled:
return 'canceled';
default:
thrownewError('unknownstatus');
    }
  }
```

```
      findInCategories(categories: FlatNode<CategoryNode>[], id:
number): CategoryNode {
      returncategories.find((x) => x.id === id);
        }

      subStatusStringFor(status: AssetSubStatus): string {
      switch (status) {
      caseAssetSubStatus.Booked:
      return 'posted';
      caseAssetSubStatus.NotBooked:
      return 'not_booked';
      default:
      thrownewError('unknownstatus');
          }
        }

      editAsset(assetToEdit: Asset): void {
      if (assetToEdit) {
      this.ngZone
              .run(() =>this.dialog.open(EditAssetDialogComponent,
assetsDialogConfig(assetToEdit, true)))
              .afterClosed()
              .pipe(onlyTruthy<Asset>())
              .subscribe((asset)
=>this.appStore.entity('assets').update(asset));
          }
        }

      createNewAsset(): void {
      this.dialog
            .open(EditAssetDialogComponent, assetsDialogConfig({
uniqueId: v4() }, true))
            .afterClosed()
            .pipe(onlyTruthy<Asset>(), take(1))
            .subscribe((asset)
=>this.appStore.entity('assets').add(asset));
        }

      commentAsset(asset: Asset): void {
      if (asset) {
      this.dialog
              .open(CommentDialogComponent, {
...assetsDialogConfig(asset, false), autoFocus: true })
              .afterClosed()
              .pipe(onlyTruthy())
```

```
                .subscribe(
                  ([canceled, comment]) =>
                    !canceled&&
        this.appStore.raw.dispatch(
        commentAsset.request({
        request: {
        id: asset.id,
        payload: comment
                        }
                      })
                    )
                );
            }
          }

        addStampNo(asset: Asset): void {
        if (asset) {
        this.dialog
                .open(StampnoDialogComponent, {
...assetsDialogConfig(asset, false), autoFocus: true })
                .afterClosed()
                .pipe(onlyTruthy())
                .subscribe(
                  ([canceled, stampNo]) =>
                    !canceled&&
        this.appStore.raw.dispatch(
        addStampNo.request({
        request: {
        id: asset.id,
        payload: stampNo
                        }
                      })
                    )
                );
            }
          }

        toggleSubscriptionStatus(): void {
        letisCurrentlySubscribed: boolean;
        this.appStore.raw
              .select(isSubscribedToNotifications)
              .pipe(first())
              .subscribe((x): boolean => (isCurrentlySubscribed = x));
        this.appStore.raw.dispatch(
        isCurrentlySubscribed
```

```
                ? unsubscribeFromNotifications.request(undefined)
                : subscribeToNotifications.request(undefined)
            );
          }

        enterBatchBooking(): void {
        this.dialog.open(BatchPostDialogComponent, {
        width: '95%',
        id: 'batch-post-container' // requiredfor CSS magic
            });
          }

        enterSettings(): void {
        this.dialog.open(EditSettingsDialogComponent, {
        id: 'categories-editor-container',
        minWidth: '600px',
        autoFocus: false
            });
          }

        openReport(): void {
        window.open("../reports", "_blank", "popup")
          }

        toggleStatus($event: number): void {
        this.appStore.raw.dispatch(changeFilterStatus({ status:
this.statuses[$event] }));
          }

        exportCurrentPageToCsv(): void {

this.csvExportService.performCsvExport(this.appStore.raw.select(curren
tPageAssets));
          }

        exportAllAssetsToCsv(): void {
        this.csvExportService.performCsvExport(this.assets$);
          }

        enterImport(): void {
        this.dialog.open(ImportAssetsDialogComponent,
assetsDialogConfig({}, true));
          }
```

```
      switchColumn($event: MouseEvent, column: ExtendedAssetColumn):
void {
          $event.preventDefault();
          $event.stopPropagation();
      this.appStore.raw.dispatch(switchColumn({ column }));
        }

      ngAfterViewInit(): void {
      this.subscription.add(this.isLoading$.subscribe(()
=>this.changeDetectorRef.detectChanges()));
        }
      goToHelp(): void {
      this.subscription.add(
            this.country$.pipe(take(1)).subscribe((language) => {
      switch (language) {
      caseLanguages.Swedish: {
      window.open(

'https://support.24sevenoffice.com/hc/sv/articles/360021362680-Hur-
l%C3%A4gger-jag-till-en-tillg%C3%A5ng-i-anl%C3%A4ggningsregistret-',
                    '_blank'
                  );
      break;
                }
      caseLanguages.USA:
      caseLanguages.English: {
      window.open('https://24sevenofficeus.zendesk.com/hc/en-us',
'_blank');
      break;
                }
      default: {
      window.open(

'https://support.24sevenoffice.com/hc/no/articles/360021362680-
Hvordan-legge-til-en-eiendel-i-Anleggsregisteret-',
                    '_blank'
                  );
                }
              }
            })
          );
```

Б.2 Текст файлаreports.component.ts

```
import { AfterViewInit, ChangeDetectorRef, Component, OnDestroy,
OnInit } from '@angular/core';
import { balanceAccounts, balanceAccountsLoaded } from
'@fromStore/settings/accounts/accounts.selectors';
import { BehaviorSubject, combineLatest, Observable,
Subscription } from 'rxjs';
import { Asset, CategoryNode, CompanyAccount, CompanyType,
FlatNode } from '@coreUtils/types/assets.types';
import {distinctUntilChanged, map, startWith, take,
withLatestFrom} from 'rxjs/operators';
import { onlyTruthy } from '@coreUtils/only-truthy';
import {
categoriesArray,
categoriesLoaded,
flatCategoriesArray
} from '@fromShared/reducers/categories/categories.selectors';
import { companyTypesArray, companyTypesLoaded } from
'@fromShared/reducers/company-types/company-types.selectors';
import { FormBuilder, FormControl, FormGroup, Validators } from
'@angular/forms';
import { SearchParameters, SearchParametersKey } from
'@coreUtils/types/reports.types';
import { loadReportsAssets, setReportsCellContext } from
'@fromScreens/reports/reducers/actions';
import { assetReport, isLoadingReportsAssets, reportsAssets }
from '@fromScreens/reports/reducers/selectors';
importmomentfrom 'moment';
import { PickedYearAndMonth } from '@fromShared/controls/month-
picker/month-picker.types';
import { ScrollService } from
'@fromShared/services/scroll.service';
import { PdfReportGeneratorService } from
'@fromScreens/reports/services/pdf-report-generator.service';
import { ExcelService } from
'@fromScreens/reports/services/excel.service';
import { isEqual } from 'lodash-es';
import { ApplicationStore } from '@fromRoot/data/application-
store.service';
import {currentAuth0Identity} from
"@fromStore/authentication/authentication.selectors";
```

```
@Component({
selector: 'app-reports',
templateUrl: './reports.component.html',
styleUrls: ['./reports.component.scss']
})
exportclassReportsComponentimplementsOnInit, OnDestroy,
AfterViewInit {
balanceAccounts$: Observable<CompanyAccount[]>;
flatCategories$: Observable<FlatNode<CategoryNode>[]>;
companyAssetTypes$: Observable<CompanyType[]>;
isDataNotLoaded$: Observable<boolean>;
controls: Map<SearchParametersKey, FormControl>;
formGroup: FormGroup;
preventSearch$: Observable<boolean>;
assets$: Observable<Asset[]>;
privatereadonlysubscription: Subscription = newSubscription();
privatereadonlyparametersSubject$ =
newBehaviorSubject<SearchParameters>(undefined);

constructor(
privatereadonlyappStore: ApplicationStore,
privatereadonlyfb: FormBuilder,
privatereadonlypdfReportGenerator: PdfReportGeneratorService,
privatereadonlyexcelService: ExcelService,
privatereadonlychangeDetectorRef: ChangeDetectorRef
  ) {}

  // eslint-disable-next-line @angular-eslint/no-empty-
lifecycle-method
ngOnInit(): void {
conststore = this.appStore.raw;
constnow = moment();
constcontrols: { [k inSearchParametersKey]: FormControl } = {
type: this.fb.control(undefined),
category: this.fb.control(undefined),
accountsFrom: this.fb.control(1000, Validators.required),
accountsTo: this.fb.control(1299, [Validators.required]),
period: this.fb.control(
        {
month: now.month(),
year: now.year()
        },
```

```
      Validators.required
            ),
      includeFinishedAssets: this.fb.control(false,
Validators.required)
          };

      this.controls = Object.entries(controls).reduce(
            (m, [k, v]) =>m.set(k asSearchParametersKey, v),
      newMap<SearchParametersKey, FormControl>()
          );
      this.formGroup = this.fb.group(controls);
      this.balanceAccounts$ = store.select(balanceAccounts).pipe(
      onlyTruthy(),
      map((x) =>x.slice())
          );

      this.flatCategories$ = store.select(flatCategoriesArray).pipe(
      onlyTruthy(),
      map((x) =>x.slice())
          );

      this.companyAssetTypes$ = store.select(companyTypesArray).pipe(
      onlyTruthy(),
      map((x) =>x.slice())
          );

      this.isDataNotLoaded$ = combineLatest([
      store.select(companyTypesLoaded),
      store.select(categoriesLoaded),
      store.select(balanceAccountsLoaded)
          ]).pipe(
      map((x) =>x.some((s) => !s)),
      distinctUntilChanged()
          );

      this.assets$ = store.select(reportsAssets);
      this.subscription.add(
            this.assets$.subscribe((x) => {
      this.formGroup.enable({ onlySelf: false });
      this.formGroup.markAllAsTouched();
            })
          );
      this.preventSearch$ = combineLatest([
```

```
      this.formGroup.statusChanges.pipe(
      map((x) => x === 'VALID'),
      startWith(false)
            ),
      this.isDataNotLoaded$,
      store.select(isLoadingReportsAssets)
          ]).pipe(map(([isValid, isDataNotLoaded, isLoading])
=>isDataNotLoaded || !isValid || isLoading));


this.subscription.add(this.formGroup.valueChanges.subscribe(this.param
etersSubject$));

      this.subscription.add(
            this.isDataNotLoaded$.subscribe((x) => {
      if (x) {
      this.formGroup.disable({
      onlySelf: false
                });
      return;
              }
      this.formGroup.enable({
      onlySelf: false
              });
      this.formGroup.markAllAsTouched();
            })
          );

      constpc = this.controls.get('period');
      this.subscription.add(

pc.valueChanges.pipe(distinctUntilChanged(isEqual)).subscribe((x) => {
      if (pc.status === 'VALID') {
      constperiod = x asPickedYearAndMonth;
      store.dispatch(
      setReportsCellContext({
      now: moment({
                      y: period.year,
                      M: period.month,
                      d: 1
                    })
                      .local(true)
                      .utc(true)
```

```
                  })
                );
              }
            })
          );

      constat = this.controls.get('accountsTo');
      this.subscription.add(
      this.controls.get('accountsFrom').valueChanges.subscribe((v) =>
{
      at.setValidators([Validators.required, Validators.min(v || 0)]);
      at.markAsTouched();
      at.updateValueAndValidity();
            })
          );
        }

      findInCategories(categories: FlatNode<CategoryNode>[], id:
number): CategoryNode {
      returncategories.find((x) => x.id === id);
        }

      exportToExcel(): void {
      conststore = this.appStore.raw;
      combineLatest([store.select(assetReport),
store.select(categoriesArray), store.select(companyTypesArray),
store.select(currentAuth0Identity)])
            .pipe(take(1))
            .subscribe(([reports, categories, companyTypes,
auth0Identity]) => {
      this.excelService.generateExcel(reports, categories,
companyTypes, auth0Identity);
            });
        }

      onSearchClick(): void {
      this.appStore.raw.dispatch(loadReportsAssets.request({ request:
this.parametersSubject$.value }));
        }

      onExportPdfClick(): void {
      this.appStore.raw
            .select(assetReport)
```

```
            .pipe(take(1),
withLatestFrom(this.appStore.raw.select(currentAuth0Identity)))
            .subscribe(([data, auth0Identity])
=>this.pdfReportGenerator.buildAndSave(data, auth0Identity));
        }

      ngOnDestroy(): void {
      this.subscription.unsubscribe();
        }

      ngAfterViewInit(): void {
      this.subscription.add(this.isDataNotLoaded$.subscribe(()
=>this.changeDetectorRef.detectChanges()));
        }
      }
```

ДОДАТОК В

Слайди презентації

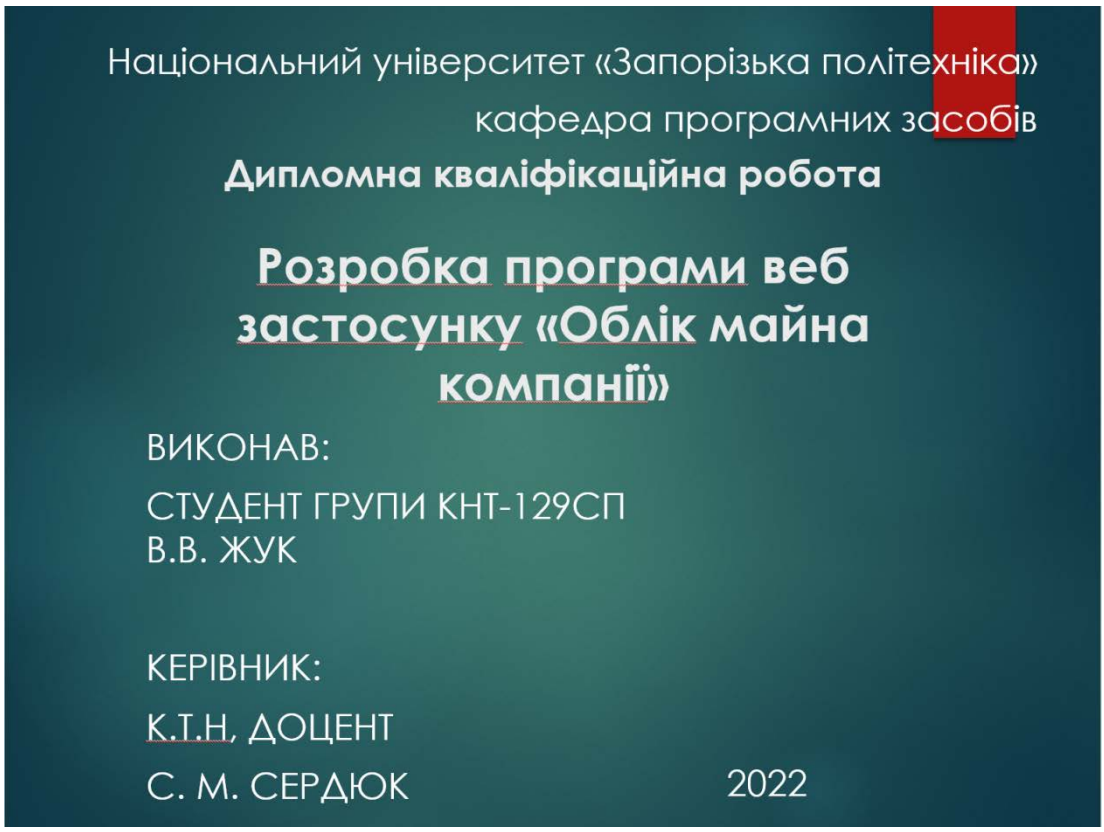

Рисунок В.1 – Слайд № 1

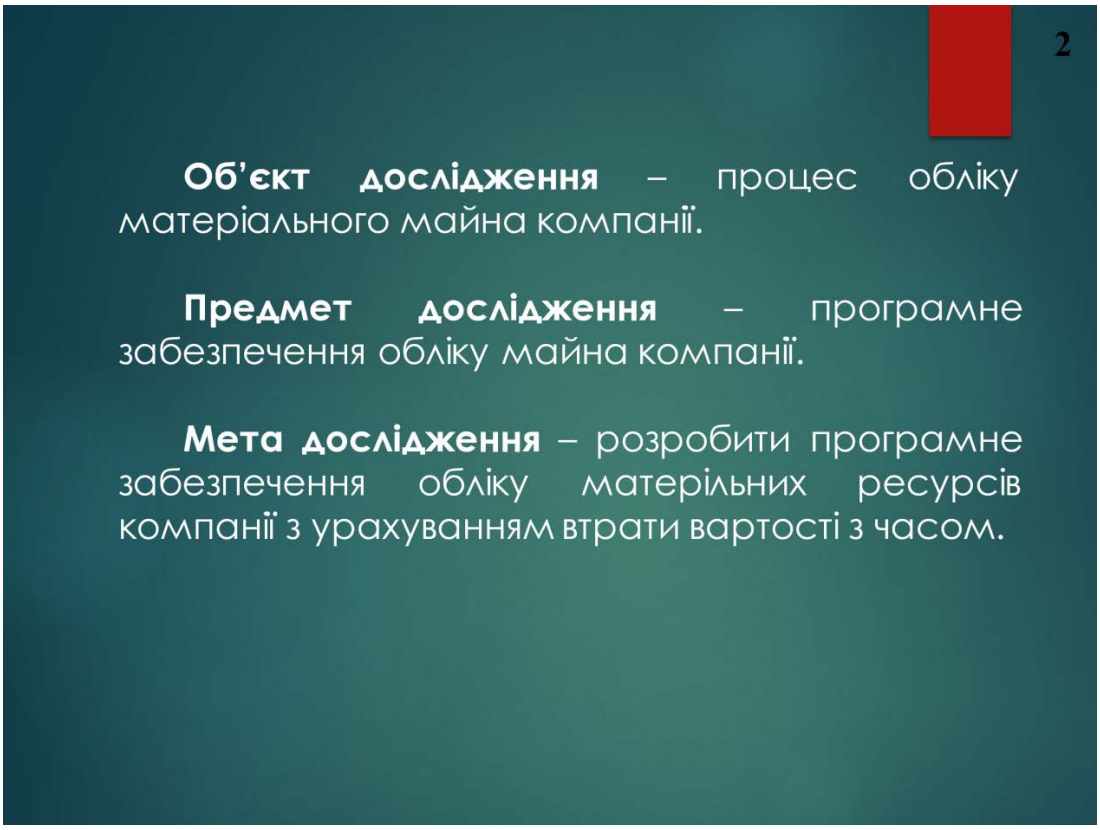

Рисунок В.2 – Слайд № 2

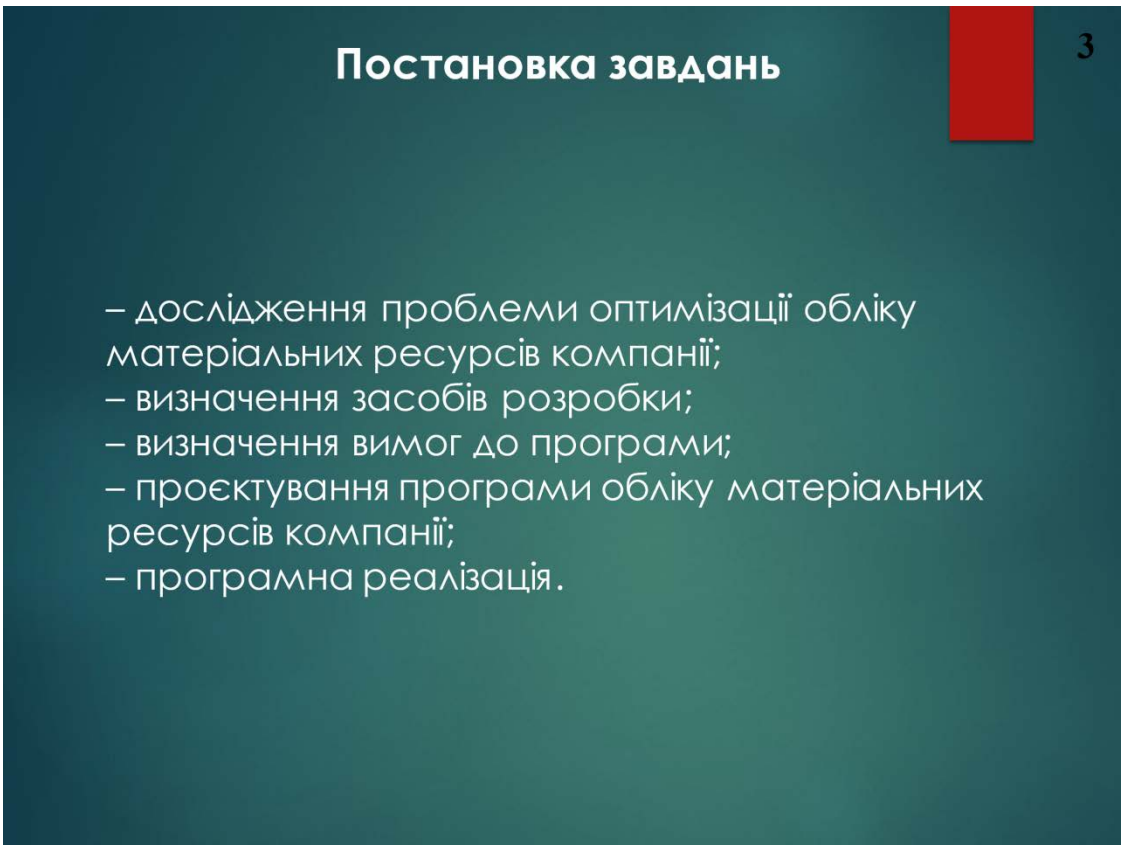

Рисунок В.3 – Слайд № 3

Показник	1С: ERP	Microsoft Dynamics 365	Парус-Підприємство	Розробка
Орієнтація на малий бізнес	Ні	Ні	Ні	Так
Веб-застосунок	Ні	Так	Ні	Так
Інтеграція зі сторонніми сервісами	Ні	Так	Ні	Так
Хмарне сховище	Ні	Так	Ні	Так
Орієнтація на матеріальні ресурси	Так	Ні	Ні	Так
Оновлений інтерфейс	Ні	Так	Ні	Так
відстеження втрати вартості	Ні	Так	Ні	Так

Рисунок В.4 – Слайд № 4

Рисунок В.5 – Слайд № 5

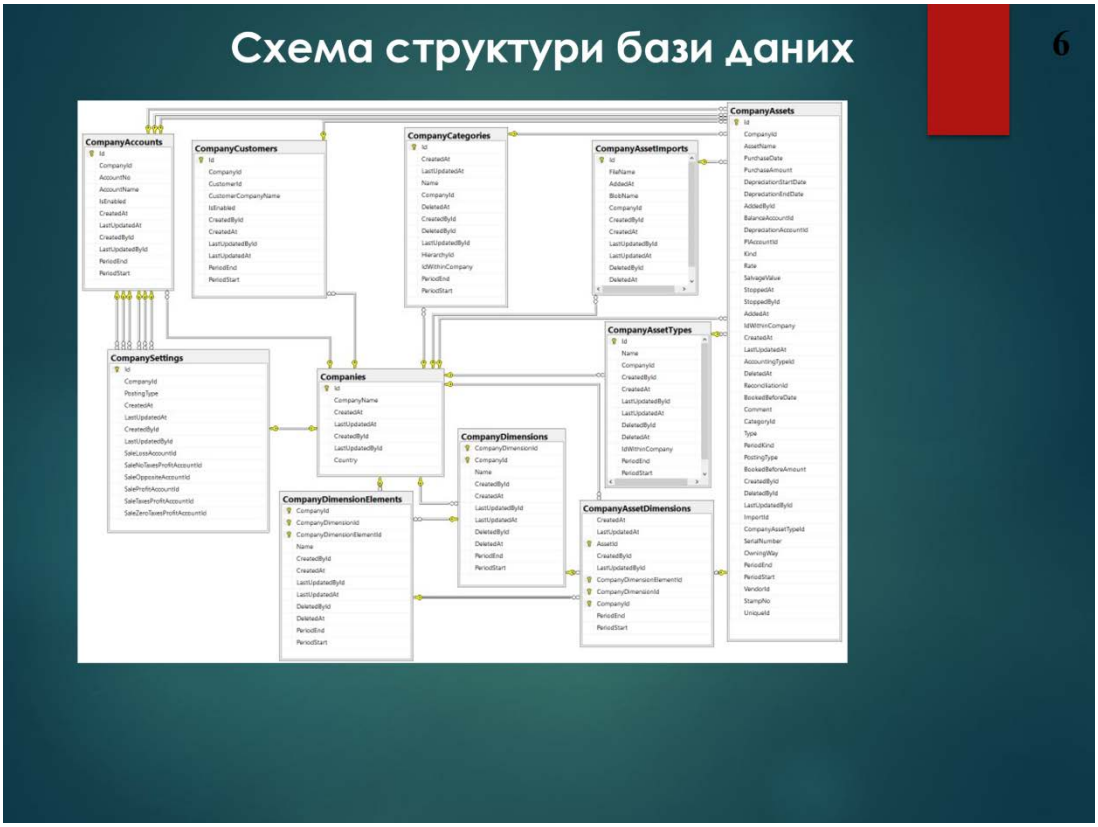

Рисунок В.6 – Слайд № 6

Рисунок В.7 – Слайд № 7

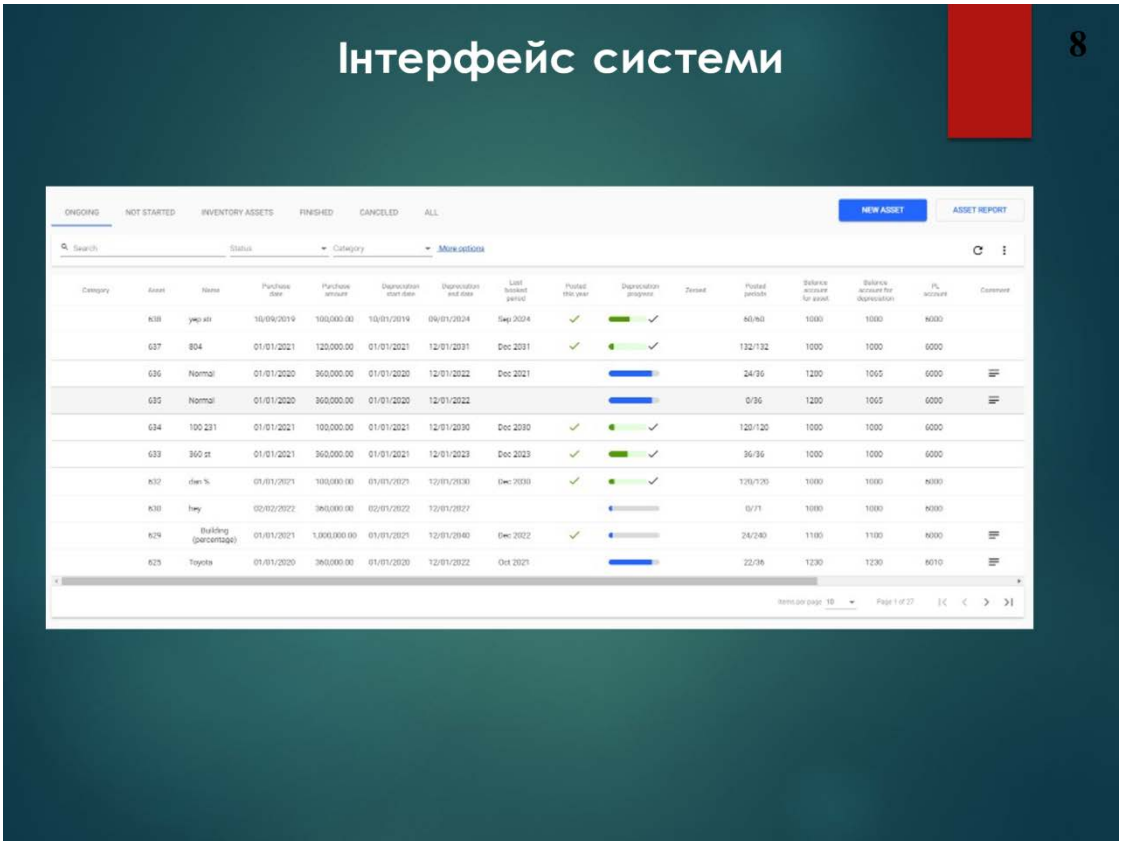

Рисунок В.8 – Слайд № 8

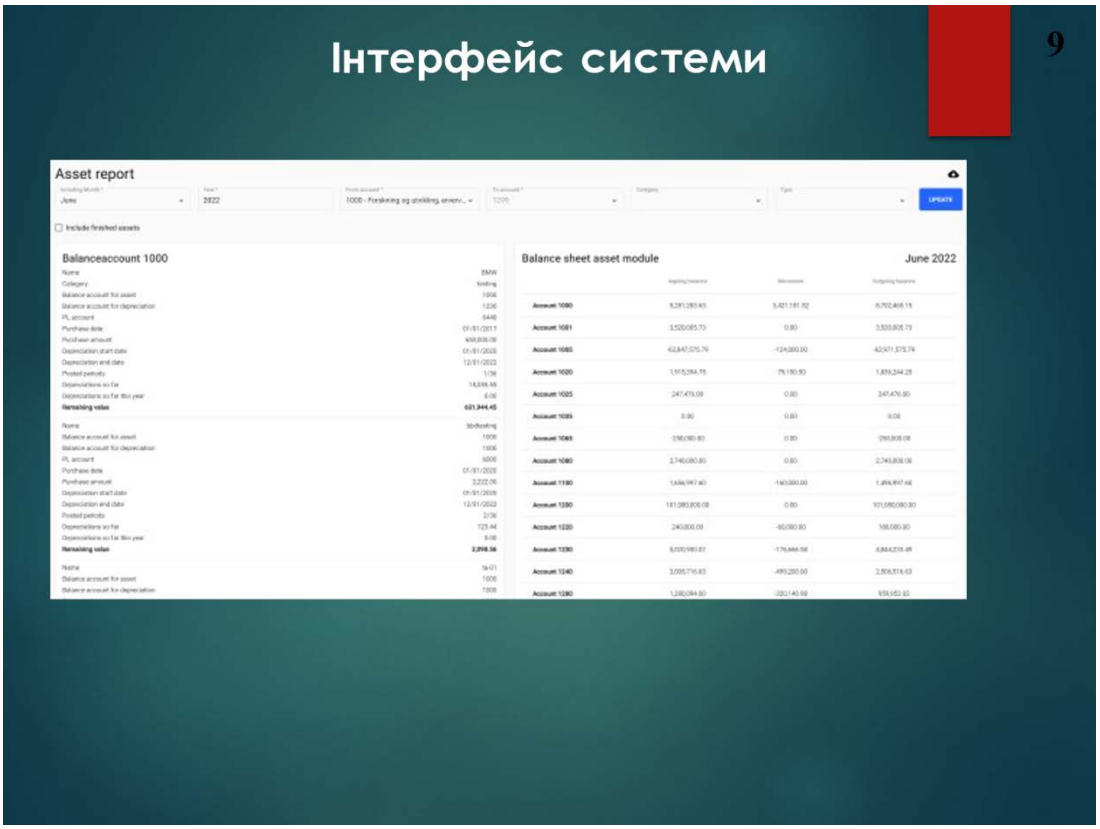

Рисунок В.9 – Слайд № 9

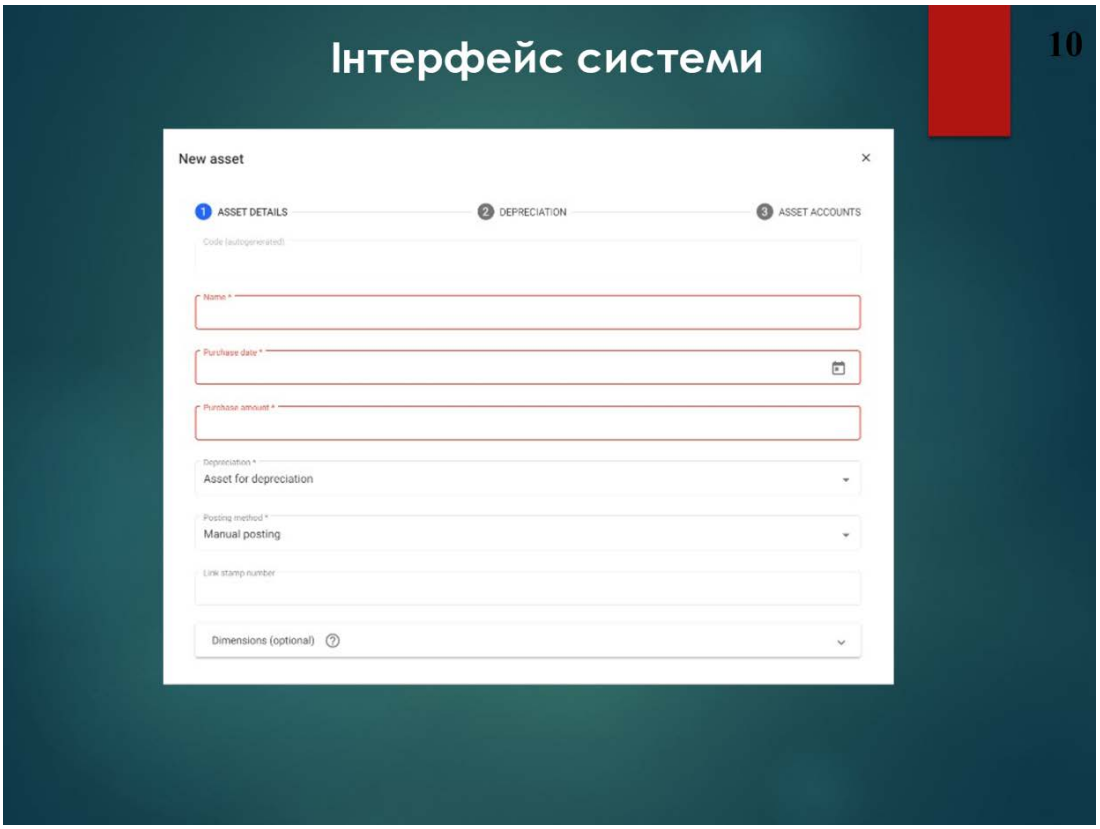

Рисунок В.10 – Слайд № 10

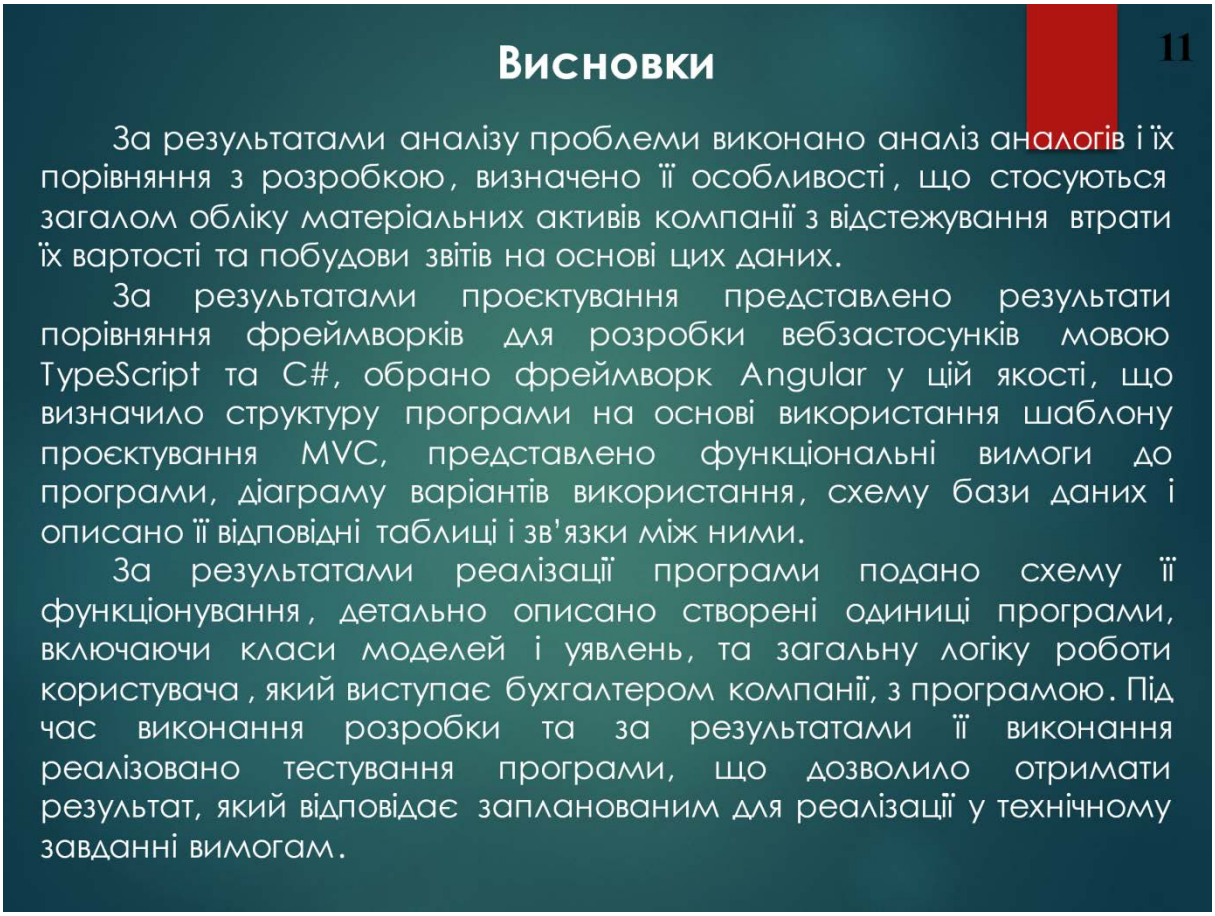

Рисунок В.11 – Слайд № 11