МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

МЕТОДИЧНІ ВКАЗІВКИ

до виконання лабораторних робіт з дисципліни

Технологія створення програмних продуктів

для студентів 3-го року навчання
Рівень вищої освіти – «Бакалавр»
Спеціальність – 122 – «Комп'ютерні науки»

ЗАТВЕРДЖЕНО
на засіданні групи забезпечення
спеціальності 122 Комп'ютерні науки
протокол № 1 від «14» серпня 2023 року
Голова групи _____(Кузніченко С.Д.)

ЗАТВЕРДЖЕНО
на засіданні кафедри АСМНСІ
Протокол № 1 від 14.08.2023
ВО Зав. кафедри _____Перелигін В.Б.

Одеса, 2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

МЕТОДИЧНІ ВКАЗІВКИ

до виконання лабораторних робіт з дисципліни

Технологія створення програмних продуктів

для студентів 3-го року навчання
Рівень вищої освіти – «Бакалавр»
Спеціальність – 122 – «Комп'ютерні науки»

ЗАТВЕРДЖЕНО
на засіданні групи забезпечення
спеціальності 122 Комп'ютерні науки
протокол № 1 від «14» січня 2023 року

Одеса, 2023

Методичні вказівки до виконання лабораторних робіт з дисципліни «Технологія створення програмних продуктів» для студентів 3-го року навчання, рівня вищої освіти «Бакалавр», за спеціальністю 122 – «Комп’ютерні науки»

Укладачі:

Гнатовська Г.А., к.т.н., доцент кафедри АСМНСІ

ЗМІСТ

ПЕРЕДМОВА

Дисципліна «Технологія створення програмних продуктів» викладається для рівня вищої освіти – «Бакалавр» спеціальності 122 – «Комп'ютерні науки» та входить до складу фахових за стандартом освітніх компонент навчального плану підготовки бакалаврів.

Викладається відповідно до навчального плану підготовки бакалаврів та силабусу.

Мета методичних вказівок – забезпечити отримання студентами теоретичних знань і практичних навичок щодо сучасних технологій створення програмних продуктів.

В процесі навчання основна увага приділяється розгляду та застосуванню теорії, знань і практики для ефективної побудови програмних систем, що задовольняють вимогам користувача і замовника. У рамках дисципліни вивчаються всі процеси, що ведуть до створення програмних продуктів (ПП): від розробки вимог до ПП через проектування, розробку та атестацію до модернізації програмних систем.

Ці методичні вказівки містять теоретичні відомості та розглянуті приклади виконання лабораторних робіт з дисципліни «Технологія створення програмних продуктів» та завдання. В методичних вказівках розглядаються питання, які відповідають силабусу дисципліни.

1 ЗАГАЛЬНІ ВІДОМОСТІ ПРО ДИСЦИПЛІНУ

1.1 Мета дисципліни та її місце у навчальному процесі

Створення програмної системи – вельми трудомістке завдання, і фахівець повинен мати уявлення про методи аналізу, проектування, реалізації та тестування програмних систем, орієнтуватися в існуючих підходах і технологіях.

Метою дисципліни «Технологія створення програмних продуктів» є ознайомлення студентів з теоретичними основами та практичними навичками застосування сучасних методологій, технологій та інструментальних засобів для управління процесами життєвого циклу інформаційних і програмних систем, продуктів і сервісів інформаційних технологій відповідно до вимог замовника.

В процесі навчання основна увага спрямована на вивчення та використання засобів та методів управління життєвим циклом програмного забезпечення, продуктів і сервісів інформаційних технологій відповідно до вимог і обмежень замовника, а також знань щодо розробки проектної документації (техніко-економічне обґрунтування, технічне завдання, бізнес-план, угода, договір). У рамках дисципліни вивчаються всі процеси, що ведуть до створення програмних продуктів (ПП): від розробки вимог до ПП через проектування, розробку та атестацію до модернізації програмних систем. Структурно-логічне місце дисципліни: попереднє вивчення дисциплін «Алгоритми та структури даних», «Об'єктно-орієнтоване програмування».

В результаті вивчення дисципліни «Технологія створення програмних продуктів» студенти повинні:

ЗНАТИ:

- технології створення та документування програмних продуктів;
- Методологію проектування MSF – Microsoft Solutions Framework

/ Azure DevOps Server;

- сучасні програмні засоби розробки програмного забезпечення (Azure DevOps) , умови їх застосування;
- основні моделі та методи проектування архітектури ПЗ, патерни та шаблони проектування;

ВМІТИ:

- розробляти згідно стандартів та вимог технічну документацію до програмного забезпечення: технічне завдання, ескізний проект, технічне завдання;
- застосовувати інтегроване середовище розробки ПП Visual Studio (Azure DevOps) для здійснення керування розробкою ПП відповідно до вимог і обмежень замовника;
- виконувати всі етапи проектування згідно методології MSF – Microsoft Solutions Framework / Azure DevOps Server;
- використовувати методи аналізу, проектування, реалізації та тестування програмних систем, орієнтуватися в існуючих підходах, методологіях і технологіях розробки ПП;

Володіти компетенціями:

ЗК6.Здатність вчитися й оволодівати сучасними знаннями

ЗК9. Здатність працювати в команді.

ЗК12. Здатність оцінювати та забезпечувати якість виконуваних робіт.

СК10. Здатність застосовувати методології, технології та інструментальні засоби для управління процесами життєвого циклу інформаційних і програмних систем, продуктів і сервісів інформаційних технологій відповідно до вимог замовника.

По кожній лабораторній роботі студент повинен скласти звіт, який містить в собі:

- Назву роботи.

– Мету.

– Умову завдання згідно варіанту.

– Хід виконання роботи.

– Відповіді на контрольні питання.

Оформлений звіт захищається студентом усно. Види контролю поточних знань – усне опитування під час лекцій та лабораторних занять, контрольні роботи, модульний контроль, залік.

Оцінювання лабораторних робіт:

За весь **практичний модуль ЗМ-П1** встановлена максимальна оцінка ***25 балів***, який передбачає виконання наступних лабораторних робот:

Оцінювання лабораторних робот:

За кожну з 5 лабораторних робот встановлена максимальна оцінка 5 балів.

За лабораторну роботу №1 встановлена максимальна оцінка 5 балів.

За лабораторну роботу №2 встановлена максимальна оцінка 10 балів.

За лабораторну роботу №3 встановлена максимальна оцінка 5 балів.

За лабораторну роботу №4 встановлена максимальна оцінка 5 балів.

Контроль по кожному лабораторному заняттю проводиться в формі:

– перевірки виконання лабораторної роботи (максимальна кількість балів – 2); – усного опитування (кількість запитань – до 3, максимальна кількість балів – 1); –захисту результатів (максимальна кількість балів – 1).

До оцінки за лабораторні роботи входить:

– оцінка за виконання лабораторної роботи 50%

– усного опитування 20%

– захист лабораторної роботи 30%.

Підсумковою оцінкою за кожну лабораторну роботу буде сума балів за усне опитування, перевірку виконання лабораторної роботи та захист лабораторної роботи.

Підсумковою оцінкою за весь лабораторний модуль буде сума балів за всі лабораторні роботи.

Критерії оцінювання результатів контрольного заходу для ЗМ-П1: 25–23 балів – відмінно, 22–19 балів – добре, 18-15 балів – задовільно, менше 15 балів – незадовільно.

За весь **практичний модуль ЗМ-П2** встановлена максимальна оцінка ***25 балів***, який передбачає виконання наступних лабораторних робот:

За кожну з 4 лабораторних робот встановлені наступні оцінки:

За лабораторну роботу №5 встановлена максимальна оцінка 5 балів.

За лабораторну роботу №6 встановлена максимальна оцінка 10 балів.

За лабораторну роботу №7 встановлена максимальна оцінка 5 балів.

За лабораторну роботу №8 встановлена максимальна оцінка 5 балів.

До оцінки за лабораторні роботи входить:

- оцінка за виконання лабораторної роботи –50%
- усного опитування – 20%
- захист лабораторної роботи – 30%

Підсумковою оцінкою за кожну лабораторну роботу буде сума балів за усне опитування, перевірку виконання лабораторної роботи та захист лабораторної роботи. Підсумковою оцінкою за весь лабораторний модуль буде сума балів за всі лабораторні роботи.

Критерії оцінювання результатів контрольного заходу для ЗМ-П2: 25–23 балів – відмінно, 22–19 балів – добре, 18-15 балів – задовільно, менше 15 балів – незадовільно.

Загальний обсяг навчального часу для денної форми навчання – 120 годин: лекцій – 30 годин, лабораторних занять – 30 годин, самостійна робота – 60 годин.

Правила техніки безпеки та охорона праці

Згідно з «Правилами техніки безпеки в лабораторіях» студентам забороняється:

- з`являтися та знаходитись приміщенні в нетверезому стані;
- ставити поруч з клавіатурою ємності з рідиною;
- перебувати в приміщенні в верхній одежі та завалювати нею робочі столи та стільці;
- працювати в лабораторії більше 6-ти годин на день (для вагітних жінок – більше 4-х годин);
- за власною ініціативою змінювати закріплені за ними робочі місця та знаходитись в приміщенні під час роботи іншої учбової групи;
- самостійно виконувати вмикання електроживлення лабораторії та заміну складових частин ПК, що вийшли із ладу.

У випадку виявлення несправностей обчислювальної техніки студент повинен сповістити про це викладача чи будь-кого з навчально-допоміжного персоналу лабораторії.

Лабораторна робота №1

Етапи розробки програмного забезпечення: Стадія «Технічне завдання»

Мета роботи: ознакомиться с правилами складання технического задания на розробку програмного продукту та виконати власну розробку шаблону «Технічного завдання» згідно з варіантом завдання.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Розробка технічного завдання

Технічне завдання являє собою документ, в якому сформульовані основні цілі розробки, вимоги до програмного продукту, визначено терміни та етапи розробки та регламентований процес приймально-здавальних випробувань. У розробці технічного завдання беруть участь як представники замовника, так і представники виконавця. В основі цього документа лежать вихідні вимоги замовника, аналіз передових досягнень техніки, результати виконання науково-дослідних робіт, передпроектних досліджень, наукового прогнозування і т. п.

Порядок розробки технічного завдання

Розробка технічного завдання виконується в наступній послідовності. Насамперед, встановлюють набір виконуваних функцій, а також перелік і характеристики вихідних даних. Потім визначають перелік результатів, їх характеристики і способи подання.

Далі уточнюють середовище функціонування програмного забезпечення: конкретну комплектацію і параметри технічних засобів, версію операційної системи і, можливо, версії і параметри іншого встановленого програмного забезпечення, з яким належить взаємодіяти майбутньому програмному продукту.

У випадках, коли розробляється програмне забезпечення збирає і зберігає деяку інформацію або включається в управління будь-яким технічним процесом, необхідно також чітко регламентувати дії програми у разі збоїв обладнання та енергопостачання.

1. Загальні положення

1.1. Технічне завдання оформляють відповідно до діючих міжнародних стандартів на аркушах формату А4 та/або А3, як правило, без заповнення полів аркуша. Номери аркушів (сторінок) проставляють у верхній частині аркуша над текстом.

1.2. Лист затвердження і титульний лист оформляють відповідно до діючих міжнародних стандартів. Інформаційну частина (анотацію і зміст), лист реєстрації змін допускається в документ не включати.

1.3. Для внесення змін і доповнень в технічне задні на наступних стадіях розробки програми або програмного виробу випускають доповнення до нього. Узгодження і затвердження доповнення до технічного завдання проводять у тому ж порядку, який встановлений для технічного завдання.

1.4. Технічне завдання повинне містити наступні розділи:

- введення;
- найменування та область застосування;
- підставу для розробки;
- призначення розробки;
- технічні вимоги до програми або програмного виробу;
- техніко-економічні показники;
- стадії і етапи розробки;
- порядок контролю та приймання;
- додатки.

Залежно від особливостей програми або програмного виробу допускається уточнювати зміст розділів, вводити нові розділи або об'єднувати окремі з них. При необхідності допускається в технічне завдання включати додатки.

2. Зміст розділів

2.1. Вступ повинен включати коротку характеристику області застосування програми або програмного продукту, а також об'єкта (наприклад, системи), в якому передбачається їх використовувати. Основне призначення введення – продемонструвати актуальність даної розробки і показати, яке місце ця розробка займає в ряду подібних.

2.2. У розділі «Найменування та область застосування» вказують найменування, коротку характеристику області застосування програми або програмного виробу та об'єкта, в якому використовують програму або програмне виріб.

2.3. У розділі «Підстава для розробки» повинні бути зазначені:

- документ (документи), на підставі яких ведеться розробка. Таким документом може служити план, наказ, договір і т. п.
- організація, що затвердила цей документ, і дата його затвердження;
- найменування і (або) умовне позначення теми розробки.

2.4. У розділі «Призначення розробки» повинно бути вказано функціональне та експлуатаційне призначення програми або програмного виробу.

2.5. Розділ «Технічні вимоги до програми або програмного виробу» повинен містити такі підрозділи:

- вимоги до функціональних характеристик;
- вимоги до надійності;

- умови експлуатації;
- вимоги до складу і параметрів технічних засобів;
- вимоги до інформаційної та програмної сумісності;
- вимоги до маркування та упаковки;
- вимоги до транспортування і зберігання;
- спеціальні вимоги.

2.5.1. У підрозділі «Вимоги до функціональних характеристик» повинні бути зазначені вимоги до складу виконуваних функцій, організації вхідних та вихідних даних, тимчасовим характеристикам і т. п.

2.5.2. У підрозділі «Вимоги до надійності» повинні бути зазначені вимоги до забезпечення надійного функціонування (забезпечення сталого функціонування, контроль вхідної та вихідної інформації, час відновлення після відмови і т. п.).

2.5.3. У підрозділі «Умови експлуатації» повинні бути зазначені умови експлуатації (температура навколишнього повітря, відносна вологість і т. п. для обраних типів носіїв даних), при яких повинні забезпечуватися задані характеристики, а також вид обслуговування, необхідну кількість і кваліфікація персоналу.

2.5.4. У підрозділі «Вимоги до складу і параметрів технічних засобів» вказують необхідний склад технічних засобів із зазначенням їх технічних характеристик.

2.5.5. У підрозділі «Вимоги до інформаційної та програмної сумісності» мають бути вказані вимоги до інформаційних структур на вході і виході і методам вирішення, вихідних кодів, мов програмування. При необхідності повинна забезпечуватися захист інформації та програм.

2.5.6. У підрозділі «Вимоги до маркування та упаковки» в загальному випадку вказують вимоги до маркування програмного виробу, варіанти і способи упаковки.

2.5.7. У підрозділі «Вимоги до транспортування і зберігання" мають бути вказані для програмного виробу умови транспортування, місця зберігання, умови зберігання, умови складування, терміни зберігання в різних умовах.

2.5.8. У розділі «Техніко-економічні показники» повинні бути зазначені: орієнтовна економічна ефективність, передбачувана річна потреба, економічні переваги розробки в порівнянні з кращими вітчизняними і зарубіжними зразками або аналогами.

2.6. У розділі «Стадії та етапи розробки» встановлюють необхідні стадії розробки, етапи і зміст робіт (перелік програмних документів, які повинні бути розроблені, узгоджені та затверджені), а також, як правило, терміни розробки і визначають виконавців.

2.7. У розділі «Порядок контролю і приймання» повинні бути зазначені види випробувань і загальні вимоги до приймання роботи.

2.8. У додатках до технічним завданням при необхідності наводять:

- перелік науково-дослідних та інших робіт, що обгрунтовують розробку;
- схеми алгоритмів, таблиці, описи, обгрунтування, розрахунки та інші документи, які можуть бути використані при розробці;
- інші джерела розробки. У випадках, якщо будь-які вимоги, передбачені технічним завданням, замовник не пред'являє, слід у відповідному місці зазначити «Вимоги не пред'являються».

ПРАКТИЧНА ЧАСТИНА

Завдання

Розробити технічне завдання (ТЗ) на програмний продукт за наданим з переліку варіантом завдання, який визначається і надається викладачем. Перелік варіантів наведено у цих методичних вказівках на сторінках 19-21. 2. Оформити ТЗ відповідно до діючих міжнародних стандартів, при оформленні використовувати будь-який текстовий редактор. (Наприклад, пакет MS Office, текстовий редактор Word). 3. Здати і захистити роботу.

Приклад виконання

В якості виконання Лабораторної роботи №1 розглянемо рішення типового прикладу для варіанту завдання «Відділ кадрів». Цей варіант буде розглянуто в якості прикладу для всіх лабораторних робот. В межах лабораторної роботи розглянемо розробку технічного завдання на програмний продукт «Інформаційна система «Відділ кадрів підприємства».

Для виконання всіх подальших лабораторних робіт будемо використовувати виданий один раз варіант завдання, який чинний для всіх лабораторних робіт.

Технічне завдання на програмний продукт «Інформаційна система "Відділ кадрів підприємства"

Співробітники	**Кваліфікація**	**Посади**	**Відділи**
код співробітника	код кваліфікації	код посади	№ відділу
Прізвище	назва кваліфікації	назва посади	назва відділу
Ім’я		мінімальний оклад	макс. число співробітників
По батькові		максимальний оклад	
Адреса			
код кваліфікації			
код посади			
дата прийому на роботу			
№ відділу			
Освіта			

1. Введення

Це технічне завдання на розробку програмного продукту (ПП) «Інформаційна система "Відділ кадрів підприємства"**,** який призначено для автоматизації роботи працівників відділу кадрів у будь-якому підприємстві, який буде використовуватися для зберігання та обробки даних про кадри, які забезпечують роботу підприємства. Вирішення проблем відділу кадрів є важливою частиною ефективного управління персоналом.

Наприклад, можна детально описати поточний процес роботи з кадрами на підприємстві, вказати, які проблеми виникають при цьому, скільки часу займає робота з документами, які ризики існують у зв'язку з можливими помилками ведення кадрових даних. У введенні також можна описати конкурентні переваги розроблюваного програмного виробу порівняно зі схожими рішеннями на ринку, вказати, як він покращить ефективність роботи з кадрами, зменшить кількість помилок та забезпечить точність ведення документів.

Окрім того, розділ «Введення» може містити інформацію про технології, які будуть використовуватися під час розробки програмного виробу. Наприклад, можна зазначити мови програмування, фреймворки, бази даних та інші інструменти, які будуть використовуватися. В цілому, введення є важливим розділом технічного завдання, оскільки воно визначає основні мети та завдання проекту, а також описує поточну проблематику, яку розробка програмного виробу має вирішити. Це допоможе зрозуміти, які функції та можливості повинен мати розроблений програмний виріб, щоб він задовольняв потреби підприємства.

2. Найменування та область застосування

Назва програмного виробу: «Інформаційна система "Відділ кадрів підприємства". ПП буде використовуватися відділом кадрів на підприємствах різних галузей та форм власності.

Призначення програмного виробу може полягати у наступному:

- облік кадрових даних;
- ведення документообігу;
- створення звітів та аналітики;
- контроль за виконанням графіків роботи;
- планування та організація навчання та розвитку персоналу;
- забезпечення взаємодії з іншими відділами та підрозділами підприємства.

У залежності від специфіки галузі та вимог клієнта можуть бути визначені інші завдання, які повинен вирішувати програмний виріб.

Загальна мета цього розділу – чітко визначити цілі та область застосування програмного виробу, щоб зрозуміти, які задачі він повинен вирішувати, та забезпечити успішну реалізацію проекту.

3. Підстава для розробки

Розробка програмного виробу «Інформаційна система "Відділ кадрів підприємства" є необхідною для вдосконалення процесів роботи відділу

кадрів та забезпечення точної та своєчасної інформації про працівників підприємства.

Розділ "Підстава для розробки" містить інформацію про причини, які стали підставою для розробки програмного виробу. Він може містити наступні підпункти:

- Бізнес-потреби. Наприклад, відділ кадрів підприємства може потребувати автоматизації процесів обліку персоналу та забезпечення ефективного взаємодії з іншими відділами підприємства.
- Технічні обмеження. Наприклад, в разі збільшення кількості співробітників підприємства може стати важкою ручна обробка кадрових даних, тому необхідно розробити програмний виріб для автоматизації цього процесу.
- Потреби користувачів. Наприклад, користувачі можуть вимагати зручного інтерфейсу користувача та швидкої роботи програмного виробу для ефективного управління кадрами.
- Розширення функціоналу. Наприклад, можливість інтеграції з іншими програмними засобами підприємства.
- Покращення якості роботи. Наприклад, підвищення точності та швидкості обробки кадрових даних.

Підстава для розробки програмного виробу допомагає зрозуміти, які потреби вирішує програмний виріб, які проблеми він допомагає вирішувати, та допомагає визначити основні вимоги до програмного виробу. Він є важливим етапом у процесі розробки програмного виробу, оскільки допомагає встановити цілі та завдання, які потрібно буде вирішувати під час розробки програмного виробу.

4. Призначення розробки

Призначенням розробки є створення програмного виробу, який забезпечить ефективне управління персоналом підприємства. Розділ "Призначення розробки" містить інформацію про те, для чого розробляється програмний виріб та які цілі він має вирішити. Він може містити наступні підпункти:

- Функціональні цілі. Наприклад, цілі, пов'язані з автоматизацією процесів обліку персоналу та забезпечення ефективного взаємодії з іншими відділами підприємства.
- Експлуатаційні цілі. Наприклад, цілі, пов'язані зі зменшенням часу, необхідного для обробки кадрових даних, та забезпечення максимальної точності при обробці цих даних.
- Цілі відносин з користувачами. Наприклад, цілі, пов'язані зі забезпеченням зручного та легкого у використанні інтерфейсу користувача та забезпеченням максимального задоволення користувачів від роботи з програмним виробом.
- Цілі відносин з ринком. Наприклад, цілі, пов'язані зі збільшенням конкурентоспроможності підприємства, забезпеченням його стабільного розвитку та збільшенням доходів підприємства.

Інші цілі. Наприклад, цілі, пов'язані з можливістю інтеграції з іншими програмними засобами підприємства або забезпеченням безпеки та конфіденційності даних. Призначення розробки допомагає визначити, які проблеми програмний виріб має вирішити, які задачі він має виконувати та які результати він має надати. Воно також допомагає встановити пріоритети в розробці програмного виробу та визначити, на яких етапах розробки слід виконати ті чи інші пункти.

5. Технічні вимоги до програми або програмного виробу

При здійсненні опису цього розділу можливо зазначити наступне:

- Мова програмування: мова програмування JS;
- Мова програмування: мова програмування C#;
- База даних: MySQL;
- Можливість збереження та обробки даних про працівників, включаючи інформацію про особисті дані, дату прийняття на роботу, посаду, заробітну плату, кадрову документацію, звіти про роботу та інше;
- Можливість створення та збереження відомостей про відпустки, лікарняні, відрядження та інші види відпусток;
- Можливість створення графіку роботи працівників;
- Можливість використання програмного виробу для формування звітів та статистики про роботу відділу кадрів та працівників підприємства;
- Можливість автоматичного підрахунку заробітної плати та інших виплат працівникам;
- Можливість забезпечення захисту персональних даних працівників згідно з вимогами законодавства.

6. Техніко-економічні показники

В цьому розділі доцільно зазначити опис наступних пунктів з обов’язковим зазначенням і роз’ясненням значень цих складових:

- Термін розробки, наприклад 6 місяців;
- Вартість розробки, наприклад не більше 100 000 грн;
- Очікувана кількість користувачів, наприклад до 100 осіб;
- Орієнтована потужність серверів, наприклад 4 ГБ оперативної пам'яті, 500 ГБ жорсткого диска.

7. Стадії і етапи розробки

Розробка програмного виробу «Інформаційна система "Відділ кадрів підприємства" складається з таких стадій і етапів:

- Аналіз вимог;
- Розробка технічного завдання;
- Проектування архітектури програмного виробу;
- Розробка та тестування програмного коду;
- Введення даних та налаштування програмного виробу;
- Тестування та налагодження;
- Підготовка до випуску та введення в експлуатацію;
- Підтримка та обслуговування.

При складанні технічного завдання в цьому розділі доцільно зазначити стислий опис про роботи, які передбачаються на кожному з етапів при створенні ПП «Інформаційна система "Відділ кадрів підприємства".

8. Порядок контролю та приймання

В даному розділі необхідно зазначити, що контроль за розробкою програмного виробу «Інформаційна система "Відділ кадрів підприємства" здійснюється керівником проекту. Приймання програмного виробу здійснюється за погодженням з замовником та на підставі відповідних актів.

9. Висновок

Розробка програмного виробу «Інформаційна система "Відділ кадрів підприємства" є важливою задачею для підприємства, яке має значну кількість працівників. Це дозволить оптимізувати процеси роботи з кадрами та забезпечити ефективний контроль над персоналом. Програмний виріб повинен мати зручний та інтуїтивно зрозумілий інтерфейс користувача, а також відповідати всім вимогам щодо захисту персональних даних працівників. Розробка має бути проведена відповідно до стандартів програмної інженерії та з урахуванням всіх техніко-економічних показників.

Звіт

Звіт з лабораторної роботи повинен складатися з документу «Протокол лабораторної роботи», що містить:

- Назву роботи, мету;
- Хід виконання роботи (графічні результати виконання роботи);
- Відповіді на контрольні питання;
- Висновки.

Захист звіту з лабораторної роботи полягає в пред'явленні викладачеві отриманих результатів, відповідях на питання викладача.

Контрольні питання

1. Наведіть етапи розробки програмного забезпечення.

2. Що включає в себе постановка задачі та передпроектні дослідження?

3. Перерахуйте функціональні та експлуатаційні вимоги до програмному продукту.

4. Перерахуйте правила розробки технічного завдання.

5. Назвіть основні розділи технічного завдання.

ВАРІАНТИ ЗАВДАНЬ

Варіант 1. Передплата на періодичні видання

Видання	**Видавництва**	**Передплатник**	**Передплата**
Індекс	код видавництва	код передплатника	код квитанції
Назва	Назва	назва	код передплатника
вартість номеру	Адреса	адреса	Індекс видання
періодичність	e-mail		термін підписки
Код видавництва	ПІБ головного редактора		дата підписки

Варіант 2. Футбольна ліга

Гравці	**Команди**	**Персонал**	**Посади**
Код гравця	Код команди	код	код посади
Прізвище	Назва	прізвище	назва
Ім’я	Стадіон	Ім’я	
Дата народження	Місто	По батькові	
Зріст		дата народження	
Амплуа		код команди	
код команди		код посади	
Номер			

Варіант 3. Вироби народної творчості

Вироби	**Кваліфікація**	**Майстри**	**Промисли**
Код виробу	код кваліфікації	код майстра	код промислу
назва виробу	назва	прізвище	Назва промислу
код промислу		Ім’я	
Код майстра		По батькові	
		адреса	
		дата народження	
		код кваліфікації	

Варіант 4. Розклад занять викладачів

Предмети	**Заняття**	**Викладачі**	**Кафедри**
Код предмету	Код	код викладача	код кафедри
			назва
Назва	код предмету	Прізвище	
Скорочена назва	код викладача	Ім’я	
	день тижня	По батькові	
	номер пари	посада	
	№ аудиторії	код кафедри	
	Группа		

Варіант 5. Вироби кондитерських фабрик

Вироби	**Види виробів**	**Виробники**	**Форми власності**
код виробу	код виду	код виробника	код форми власності
Назва	назва виду	Назва	Назва
код виду		Адреса	
одиниця вимірювання		код форми власності	
ціна за одиницю		відповідальна особа	
Код виробника		телефон	
		e-mail	

Варіант 6. Склад

Споживач	**Товари**	**Накладні**	**Поставка**
код споживача	код товару	№ накладної	№ накладної
Назва	Назва	дата	код товару
Адреса	одиниця вимірювання	код споживача	Кількість
Телефон	ціна за одиницю		
Примітки	Виробник		

Варіант 7. Будівельна організація

Об’єкти	**Типи об’єктів**	**Матеріали**	**Витрата**
код об’єкту	код типу об’єкту	код матеріалу	код об’єкту
Назва об’єкту	назва типу	назва матеріалу	код матеріалу
код типу об’єкту		Одиниця вимірювання	витрата матеріалу
Адреса		Ціна за одиницю	

Варіант 8. Видавництво

Автори	**Книги**	**Замовники**	**Замовлення**
код автора	код книги	код замовника	код замовлення
Прізвище	Тип видання	Назва	код замовника
Ім’я	назва книги	Адреса	дата замовлення
По батькові	код автора	телефон	код книги
дата народження	вартість	примітки	Тираж
Країна			Сплачено

Варіант 9. Автосалон

Фірми	**Моделі**	**Автомобілі**	**Рахунок**	**Клієнти**
код фірми	код моделі	код автомобіля	№ рахунку	код клієнта
Назва	Назва	код моделі	код клієнта	Назва
	код фірми	дата випуску	код автомобіля	Адреса
	макс. швидкість	колір	дата	Телефон
	кількість дверей	пробіг		Примітки
	кількість місць	ціна		

Варіант 10. Відділ кадрів

Співробітники	**Кваліфікація**	**Посади**	**Відділи**
код співробітника	код кваліфікації	код посади	№ відділу
Прізвище	назва кваліфікації	назва посади	назва відділу
Ім’я		мінімальний оклад	макс. число співробітників
По батькові		максимальний оклад	
Адреса			
код кваліфікації			
код посади			
дата прийому на роботу			
№ відділу			
Освіта			

Варіант 11. Спортивне товариство

Спортсмени	**Тренери**	**Види спорту**	**Тренування**
код спортсмена	код тренера	код виду спорту	код тренера
Прізвище	прізвище	назва	код спортсмена
Ім’я	Ім’я		
По батькові	По батькові		
дата народження	дата народження		
Звання	Звання		
	код виду спорту		

Варіант 12. Ландшафтне озеленення

Замовники	**Рослини**	**Співробітники**	**Договори**
код замовника	код рослини	Код співробітника	код договору
Назва	Назва	Прізвище	дата підписання
тип замовника	кімнатне	Ім’я	Термін виконання
Адреса	морозостійке	По батькові	код співробітника
Телефон	Ціна	Дата народження	код замовника
Примітки	щільність висадки	адреса	Площа
		дата прийому на роботу	код рослини

Варіант 13. Замовлення на виклик лікаря

Лікарі	**Пацієнти**	**Виклики**	**Посади**
№ ліцензії	№ карточки	код виклику	код посади
дата отримання ліцензії	Прізвище	№ карточки	Назва
Прізвище	Ім’я	№ ліцензії	
Ім’я	По батькові	дата/час виклику	
По батькові	Стать	відмітка про відвідування	
код посади	дата народження	діагноз	
Адреса	Адреса		
Телефон	Телефон		

Лабораторна робота №2

Етапи розробки програмного забезпечення: Стадія «Ескізний проект»

Мета роботи: ознакомиться с правилами написания та навчитися створювати формальні моделі і на їх основі визначати специфікації до розробки програмних продуктів.

Підготовка до лабораторної роботи

1. Ознайомитися з лекційним матеріалом по темі «Етапи розробки програмного забезпечення. Аналіз вимог і визначення специфікацій програмного забезпечення »навчальної дисципліни «Технологія розробки програмного забезпечення».

ТЕОРЕТИЧНІ ВІДОМОСТІ

Розробка специфікацій

Розробка програмного забезпечення починається з аналізу вимог до нього. У результаті аналізу отримують специфікації розроблюваного програмного забезпечення, будують загальну модель його взаємодії з користувачем або іншими програмами і конкретизують його основні функції. При структурному підході до програмування на етапі аналізу і визначення специфікацій розробляють три типи моделей: моделі функцій, моделі даних і моделі потоків даних. Оскільки різні моделі описують проектоване програмне забезпечення з різних сторін, рекомендується використовувати відразу кілька моделей, що розробляються у вигляді діаграм, і пояснювати їх текстовими описами, словниками і т. п.

Структурний аналіз передбачає використання наступних видів моделей:

- діаграм потоків даних (DFD – Data Flow Diagrams), описують взаємодію джерел і споживачів інформації через процеси, які повинні бути реалізовані в системі;
- діаграм «сутність-зв'язок» (ERD – Entity-Relationship Diagrams), що описують бази даних розроблюваної системи;
- діаграм переходів станів (STD – State Transition Diagrams), що характеризують поведінку системи в часу;
- функціональних діаграм (методика SADT);

- специфікація процесів;
- словника термінів.

Специфікації процесів зазвичай представляють у вигляді короткого текстового опису, схем-алгоритмів, псевдокоду, Flow-форм або діаграм Насс-Шнейдермана.

Словник термінів. Словник термінів являє собою короткий опис основних понять, що використовуються при складанні специфікацій. Він повинен включати визначення основних понять предметної області, опис структур елементів даних, їх типів і форматів, а також усіх скорочень і умовних позначень.

Діаграми переходів станів. За допомогою діаграм переходів станів можна моделювати подальше функціонування системи на основі її попереднього та поточного функціонування. Модельована система в будь-який заданий момент часу знаходиться точно в одному з кінцевого безлічі станів. З плином часу вона може змінити свій стан, при цьому переходи між станами повинні бути точно визначені.

Функціональні діаграми. Функціональні діаграми відображають взаємозв'язок функцій розроблюваного програмного забезпечення. Вони створюються на ранніх етапах проектування систем, для того щоб допомогти проектувальнику виявити основні функції і складові частини проектованої системи і, по можливості, виявити і усунути істотні помилки. Для створення функціональних діаграм пропонується використовувати методологію SADT.

Діаграми потоків даних. Для опису потоків інформації в системі застосовуються діаграми потоків даних (DFD – Data flow diagrams). DFD дозволяє описати необхідну поведінку системи у вигляді сукупності процесів, що взаємодіють за допомогою зв'язують їх потоків даних. DFD показує, як кожен з процесів перетворює свої вхідні потоки даних у вихідні потоки даних і як процеси взаємодіють між собою.

Діаграми «сутність-зв'язок». Діаграма сутність-зв'язок – інструмент розробки моделей даних, що забезпечує стандартний спосіб визначення даних і відносин між ними. Вона включає сутності та взаємозв'язок, що відображають основні бізнес-правила предметної області. Така діаграма не надто деталізована, в неї включаються основні сутності і зв'язки між ними,

які задовольняють вимогам, що пред'являються до інформаційної системи (ІС).

ПРАКТИЧНА ЧАСТИНА

Завдання

На основі ТЗ для програмного продукту з лабораторної роботи № 1 виконати аналіз функціональних та експлуатаційних вимог до програмного продукту. Документ «Ескізний проект», повинен містити наступні розділи: основні технічні рішення (вибір мови програмування, структура програмного продукту, склад функцій ПП, режими функціонування). В межах «Ескізного проекту» ПП, згідно з варіантом завдання:

- розробити діаграми потоків даних
- розробити діаграму «сутність-зв'язок»;
- розробити функціональні діаграми
- розробити діаграми переходів станів.

Приклад виконання

В якості виконання Лабораторної роботи №2 розглянемо рішення типового прикладу для варіанту завдання «Відділ кадрів». Цей варіант буде розглянуто в якості прикладу для всіх лабораторних робот.

Згідно з обраною предметною областю (Інформаційна система «Відділ кадрів підприємства» для опису потоків інформації в системі застосовуються діаграми потоків даних (DFD – Data flow diagrams).

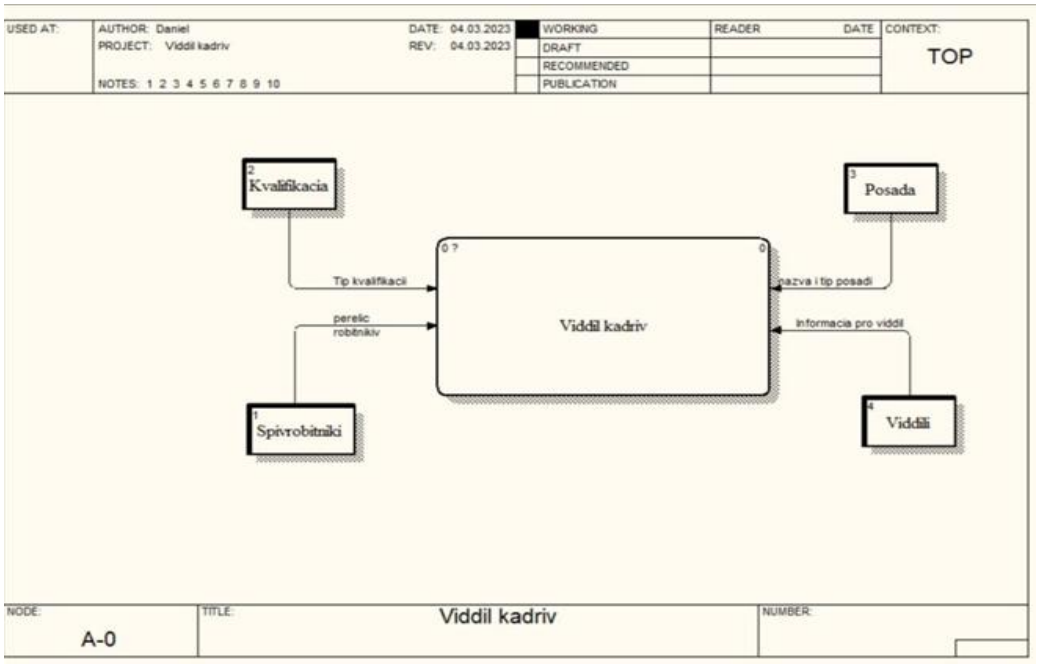

Рис. 1 – Діаграма потоків даних ІС «Відділ кадрів підприємства»

Діаграма потоків даних – це візуальне представлення ідей або інформації та обробки даних у системі.

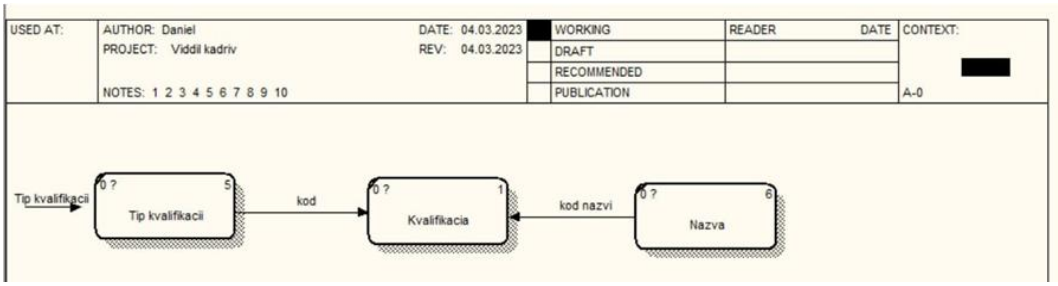

Рис. 2 – Діаграма потоків даних ІС «Відділ кадрів підприємства»

На наступному етапі створення ескізного проекту ІС «Відділ кадрів підприємства», згідно з визначеними в варіанті завдання сутностями створюється діаграма «Сутність-зв'язок», що забезпечує стандартний спосіб визначення даних і відносин між ними.

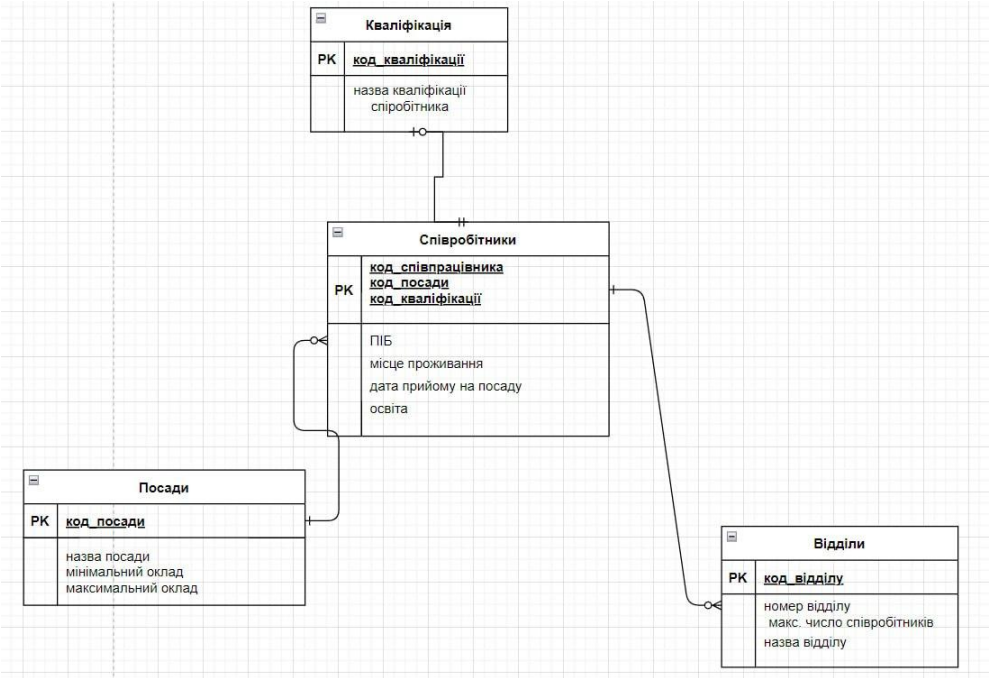

Рис. 3 – Діаграма “сутність-зв’язок”

Сутність – це клас однотипних об’єктів, інформація про які повинна бути врахована в моделі. Сутність має назву, виражену іменником в єдиному числі, і позначається прямокутником з найменуванням.

Екземпляр сутності – це конкретний представник даної сутності.

Атрибут сутності – це іменована характеристика, яка є деякою властивістю сутності. Найменування атрибута повинно бути виражено

іменником в єдиному числі (можливо, з описовими оборотами чи прийменниками).

Ключ сутності – це набір атрибутів, значення яких у сукупності є унікальними для кожного екземпляра сутності. При видаленні любого атрибута з ключа порушується його унікальність. Ключів у сутності може бути декілька. Зв’язок – це відношення однієї сутності до іншої чи до самої себе. Можливо по одній сутності знаходити інші, зв’язані з нею.

При створенні Діаграми станів ІС «Відділ кадрів підприємства» отримаємо граф спеціального виду, який представляє певний автомат. Вершинами графа є можливі стани автомата, зображувані відповідними графічними символами, а дуги позначають його переходи зі стану в стан. Діаграми станів можуть бути вкладені одна в одну для більш детального представлення окремих елементів моделі.

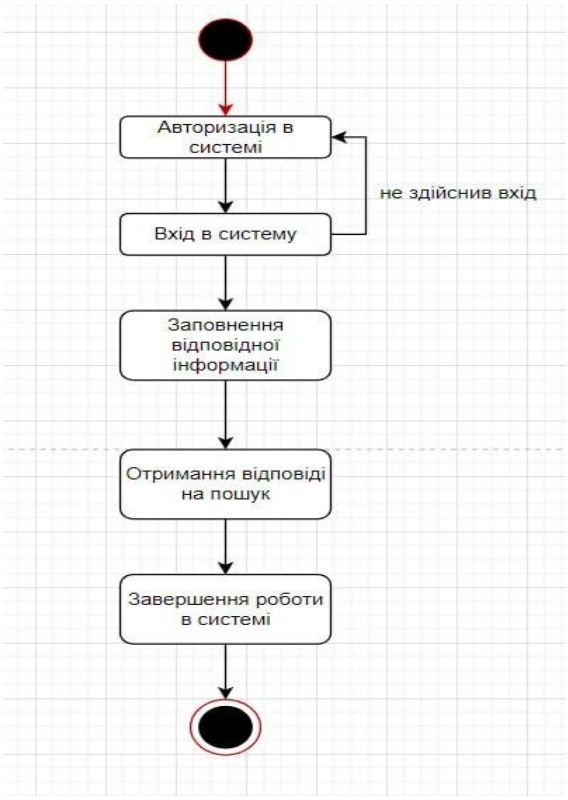

Рис. 4 – Діаграма переходів станів для клієнта ІС

Функціональна діаграма для ІС «Відділ кадрів підприємства» за методикою SADT представляє собою схему, що роз'яснює певні процеси, що відбуваються у певних функціональних частинах виробу (устаткування) чи у виробі (устаткуванні) в цілому.

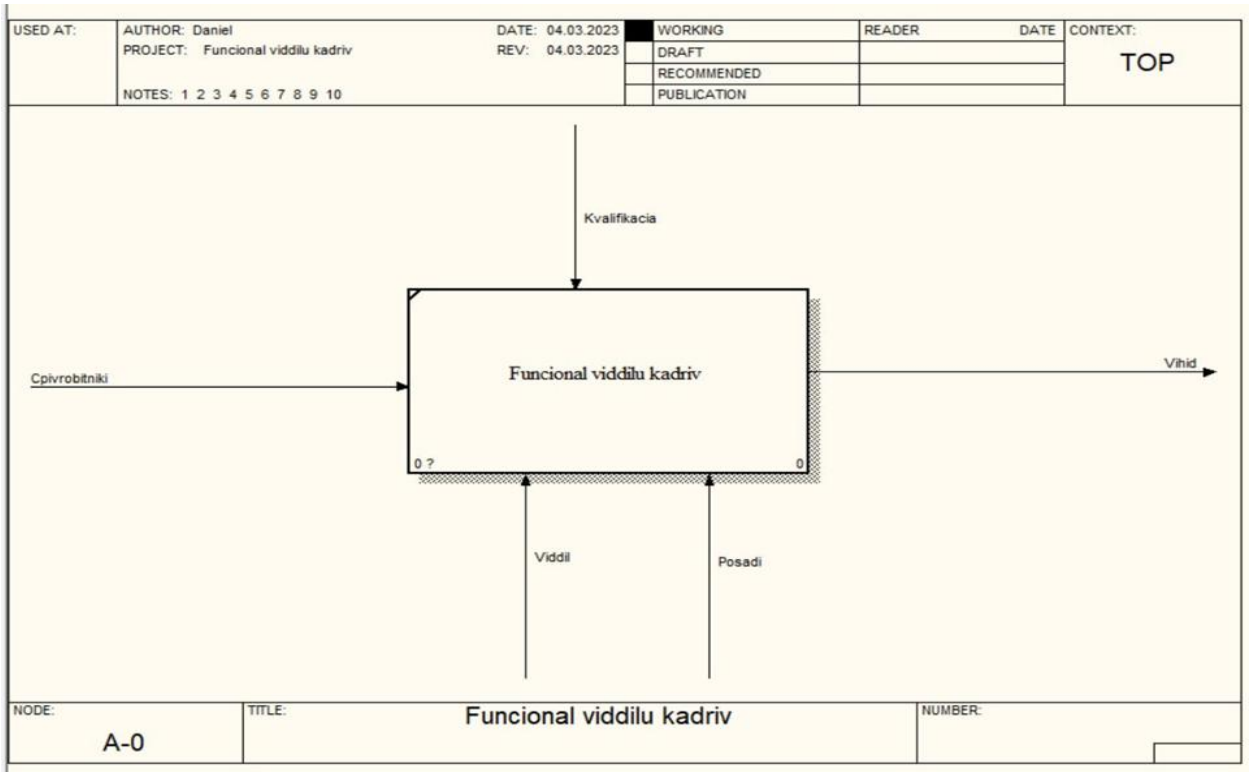

Рис. 5 – Функціональних діаграм за методикою SADT

Звіт

Звіт з лабораторної роботи повинен складатися з документу «Протокол лабораторної роботи», що містить:

- Назву роботи, мету;
- Хід виконання роботи (графічні результати виконання роботи);
- Відповіді на контрольні питання;
- Висновки.

Захист звіту з лабораторної роботи полягає в пред'явленні викладачеві отриманих результатів, відповідях на питання викладача.

Контрольні питання

1. Назвіть етапи розробки програмного забезпечення.

2. Що таке життєвий цикл програмного забезпечення?

3. У чому полягає постановка задачі та передпроектні дослідження?

4. Назвіть функціональні та експлуатаційні вимоги до програмному продукту.

5. Перелічіть складові ескізного проекту.

6. Охарактеризуйте специфікації і моделі.

Лабораторна робота №3

Етапи розробки програмного забезпечення: Стадія «Технічний проект»

Мета роботи: ознакомиться с правилами створення документу «Технічний проект»: його складових, а саме опису, схем, креслень, розрахунків, обґрунтувань, числових показників щодо ПП.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Розглянемо основні складові елементи технічного проекту ПП. Технічний проект – це образ наміченого до створення об'єкта, представлений у вигляді його опису, схем, креслень, розрахунків, обгрунтувань, числових показників. Мета технічного проекту – визначення основних методів, які використовуються при створенні інформаційної системи, і остаточне визначення її кошторисної вартості. Технічне проектування підсистем здійснюється відповідно до затвердженого технічного завдання.

Технічний проект програмної системи докладно описує:

- виконувані функції і варіанти їх використання;
- відповідні їм документи;
- структури оброблюваних баз даних;
- взаємозв'язок даних;
- алгоритми обробки даних.

Технічний проект повинен включати дані про обсяги та інтенсивність потоків оброблюваної інформації, кількості користувачів програмної системи, характеристику обладнання та програмного забезпечення, що взаємодіє програмним продуктом що проектується.

При розробці технічного проекту висвітлюються наступні розділи:

- загальна інформація щодо призначення ПП;
- пояснювальна записка до технічного проекту, яка передбачає наявність вступної інформації, що дозволяє її споживачеві швидко остворити дані по конкретному проекту;
- опис систем класифікації та кодування;
- перелік вхідних даних (документів): перелік інформації, яка використовується як вхідний потік і служить джерелом накопичення;

- перелік вихідних даних (документів): перелік інформації, яка використовується для аналізу накопичених даних;
- опис програмного забезпечення: перелік програмного забезпечення і СУБД, які планується використовувати для створення інформаційної системи;
- опис технічних засобів: перелік апаратних засобів, на яких планується робота програмного продукту;
- проектна оцінка надійності системи: експертна оцінка надійності з виявленням найбільш благополучних ділянок програмної системи та її вузьких місць;
- відомість обладнання та матеріалів: перелік обладнання та матеріалів, які будуть потрібні в ході реалізації проекту.

Технічне завдання на створення ПП в обраній предметній області передбачає наявність структурної, функціональної схем, розробку алгоритмів та структурних карт.

Структурна схема. Структурної називають схему, що відображає склад і взаємодія з управління частинами розроблюваного програмного забезпечення. Структурна схема визначається архітектурою розроблюваного ПЗ.

Функціональна схема. Функціональна схема – це схема взаємодії компонентів програмного забезпечення з описом інформаційних потоків, складу даних в потоках та зазначенням використовуваних файлів і пристроїв.

Розробка алгоритмів. Метод покрокової деталізації реалізує спадний підхід до програмування і передбачає покрокову розробку алгоритму.

Структурні карти. Методика структурних карт використовується на етапі проектування ПП для того, щоб продемонструвати, яким чином програмний продукт виконує системні вимоги. Структурні карти Константайна призначені для опису відносин між модулями.

Техніка структурних карт Джексона заснована на методі структурного програмування Джексона, який виявляє відповідність між структурою потоків даних і структурою програми. Основна увага в методі сконцентровано на відповідності вхідних і вихідних потоків даних.

ПРАКТИЧНА ЧАСТИНА

Завдання

На основі ТЗ ПП з лабораторної роботи №1 і специфікацій з лабораторної роботи №2, розробити уточнені алгоритми програмного продукту, які складають програмні модулі, використовувати метод покрокової деталізації. Виконати аналіз функціональних та експлуатаційних вимог до програмного продукту. На основі алгоритмів розробити структурну схему програмного продукту згідно з варіантом завдання.

- розробити структурну схему у вигляді структурних карт Константайна;
- представити структурну схему у вигляді структурних карт Джексона.
- розробити функціональну схему програмного продукту.

Оформити результати в вигляді технічного проекту ПП, використовуючи текстовий редактор.

Приклад виконання

В якості виконання Лабораторної роботи №3 розглянемо рішення типового прикладу для варіанту завдання «Відділ кадрів».

Технічний проект при реалізації ПП «ІС «Відділ кадрів підприємства» повинен включати опис наступних пунктів:

1. *Функції,* які повинні бути реалізовані у ПП та їх варіанти використання (згідно з таблицею у варіанті завдання):

- Збір і зберігання персональних даних працівників підприємства (ім'я, прізвище, контактна інформація, дата народження, освіта, досвід роботи тощо).
- Обробка даних про кваліфікацію, навички та досвід роботи працівників.
- Керування процесами прийому на роботу та звільнення працівників, оформлення документів та реєстрація в системі.
- Підтримка бази даних вакансій та резюме працівників, пошук відповідностей між ними.
- Організація навчання та розвитку працівників, відслідкування їх кваліфікації, професійного розвитку та здібностей.

Визначимо *відповідні документи,* які потрібні для функціонування інформації у ПП «ІС «Відділ кадрів підприємства»:

- Перелік обов'язкових документів при прийомі на роботу (анкета, трудовий договір, реєстрація податкових відрахувань тощо).
- Перелік документів, що необхідні при звільненні працівників (заява на звільнення, висновок комісії про причину звільнення, документи, що підтверджують виплату заробітної плати та компенсацій).
- Документація, пов'язана з професійним розвитком та навчанням працівників (плани навчання, звіти про навчання, сертифікати тощо).
- Структури баз даних, які обробляються:
- База даних про працівників (персональні дані, кваліфікація, навички, досвід роботи).
- База даних вакансій та резюме (опис вакансій, персональні дані працівників, що надіслали свої резюме).

Взаємозв'язок даних:

- Дані про працівників пов'язані з іншими базами даних, такими як база даних оплати праці та база даних відпусток.
- Дані про вакансії пов'язані з базою даних з інформацією про проекти компанії та з базою даних з інформацією про робочі місця.

Алгоритми обробки даних:

- Алгоритми для збору та зберігання персональних даних працівників.
- Алгоритми для пошуку відповідностей між вакансіями та резюме.
- Алгоритми для виконання операцій прийому на роботу та звільнення працівників.
- Алгоритми для реєстрації та обробки запитів на навчання та розвиток працівників.

У загальному, технічний проект ПП «ІС «Відділ кадрів підприємства» повинен детально описувати всі важливі аспекти функціонування та обслуговування кадрової діяльності підприємства. Він допомагає забезпечити ефективну та оптимальну роботу відділу кадрів, а також сприяє ефективному використанню ресурсів підприємства.

Відділ кадрів – це важлива складова будь-якої організації, яка забезпечує збір та аналіз інформації про працівників, управління кадровими процесами та розвиток персоналу.

Нижче наведено уточнені алгоритми, структурну та функціональну схеми програмного продукту, а також структурні карті Константайна та Джексона для ПП «ІС «Відділ кадрів підприємства».

Уточнені алгоритми ПП «Інформаційна система «Відділ кадрів підприємства»

Крок 1: Збір інформації про нового працівника.

- Запросити відділ кадрів заповнити форму нового працівника.
- Зберегти інформацію про працівника в базі даних.
- Підготувати документи для оформлення прийому на роботу.
- Надіслати документи на підпис до керівництва.

Крок 2: Управління даними про працівників.

- Запросити відділ кадрів внести зміни до даних про працівників.
- Перевірити відповідність змін діючому законодавству.
- Оновити інформацію про працівника в базі даних.
- Забезпечити доступ до актуальних даних для відповідних відділів організації.

Крок 3: Розвиток персоналу.

- Перевірити наявність потреб у навчанні та розвитку працівників.
- Розробити план навчання та розвитку.
- Відправити пропозиції про навчання та розвиток працівників на затвердження керівництву.
- Організувати навчання та розвиток працівників відповідно до затвердженого плану.

Рис. 6 – Структурна схема ПП «ІС «Відділ кадрів підприємства»

Функції модуля збору даних про працівників:

- Запит форми для нового працівника
- Збереження інформації про працівника в базі даних

Функції модуля управління даними про працівників:

- Оновлення інформації про працівників в базі даних
- Забезпечення доступу до актуальних даних про працівників

Функції модуля розвитку персоналу:

- Перевірка потреби у навчанні та розвитку працівників
- Розробка плану навчання та розвитку працівників
- Організація навчання та розвитку працівників.

Наступним етапом при виконанні технічного завдання було складення структурних карт.

Карти Константайна наведені нижче:

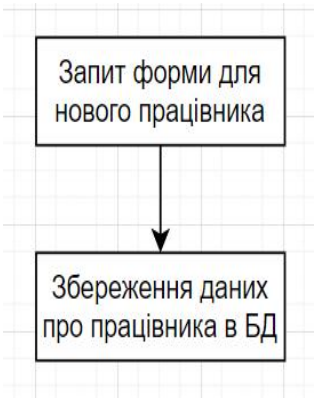

Рис. 7 – Карта Константайна для модуля збору даних про працівників

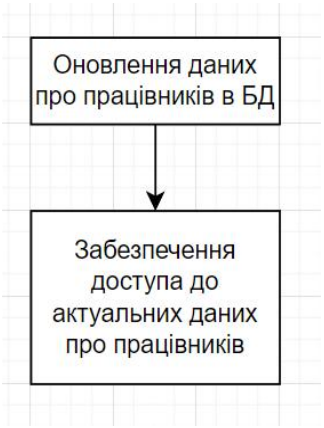

Рис. 8 – Карта Константайна для модуля управління даними про працівників

Рис. 9 – Структурна карта Константайна для модуля розвитку персоналу

Структурні карти Константайна є моделлю зв'зків між модулями програми. Вузли структурних карт відповідають модулям і областям даних, потоки зображають міжмодульні зв'язки. На діаграмі спеціальними вузлами зображають циклічні і умовні виклики модулів, а потоки проходять через ці спеціальні вузли.

Техніка структурних карт Джексона базується на методі структурного програмування Джексона, який виявляє відповідність між структурою потоків даних і структурою програми. Основна увага у методі сконцентрована на відповідності вхідних і вихідних потоків даних.

Структурна карта Джексона наведена нижче:

Рис. 10 – Структурна карта Джексона

Звіт

Звіт з лабораторної роботи повинен складатися з документу «Протокол лабораторної роботи», що містить:

- Назву роботи, мету;
- Хід виконання роботи (графічні результати виконання роботи);
- Відповіді на контрольні питання;
- Висновки.

Захист звіту з лабораторної роботи полягає в пред'явленні викладачеві отриманих результатів, відповідях на питання викладача.

Контрольні питання

1. Назвіть етапи розробки програмного забезпечення.
2. У чому полягає проектування програмного забезпечення?
3. Перерахуйте складові технічного проекту.
4. Охарактеризуйте структурний підхід до програмування.
5. З чого складаються структурна і функціональна схеми?
6. Охарактеризуйте метод покрокової деталізації при складанні алгоритмів програм.
7. Наведіть поняття псевдокоду.
8. У чому полягає методика Константайна?
9. У чому полягає методика Джексона?

Лабораторна робота №4

Проектування програмної системи за методологією Azure DevOps

Мета роботи: познайомити студентів та здійснити проектування згідно методології MSF – Microsoft Solutions Framework / Azure DevOps Server. За завданням викладача створити командний проект з розробки програмного продукту, використовуючи інтегровану середу Visual Studio та Azure DevOps. Налаштувати параметри створеного командного проекту.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Microsoft Solutions Framework (MSF) є методологією розробки ПЗ, яка являє собою узагальнення кращих проектних практик, які використовувалися командами розробників Microsoft. Дана методологія описує управління людьми і робочими процесами при розробці IT-рішень. Концепція управління життєвим циклом додатків, прийнята розробниками ПЗ, призвела до того, що методологія MSF стала складовою частину продукту Visual Studio Team System (VSTS), який реалізовував підхід Microsoft в плані – ALM.

У продукт VSTS увійшли шаблони процесів для реалізації положень моделі CMMI – MSF for CMMI і моделей гнучкої розробки ПЗ - MSF for Agilei Scrum. Таким чином, VSTS є інструментарієм управління життєвим циклом додатків, який дозволяє створювати програмні системи різного призначення в командах, які дотримуються різних підходів до управління процесами розробки ПЗ.

Microsoft Team Foundation Server, або скорочено Microsoft TFS, вперше був випущений в 2005 році, щоб забезпечити управління вихідним кодом, управління вимогами, управління проектами, звітування, автоматизовані збірки, управління лабораторією, тестування та управління випуском для команд розробників програмного забезпечення будь-якого розміру.

Версії Microsoft TFS, випущені після вересня 2018 року, називаються Azure DevOps Server та Azure DevOps Services. Перший доступний локальним рішенням, яке передає потужність Azure DevOps у спеціальні середовища. Останній - це хмарний сервіс, що працює на хмарній платформі Microsoft Azure. Він використовує той самий код з локальною версією, але не вимагає жодних налаштувань, оскільки користувачі можуть просто увійти в

систему зі своїми обліковими записами Microsoft і миттєво налаштувати середовище та створювати проекти.

Microsoft TFS – це багатофункціональне рішення для відстеження роботи, обміну кодами та доставки програмного забезпечення, яке пропонує велику колекцію інструментів, розроблених для спрощення та ефективнішої спільної розробки програмного забезпечення. Він підтримує широкий спектр мов програмування, включаючи C #, Python та Java, що дозволяє розробникам висловлюватись на тих мовах, які вони почувають найбільш впевнено.

DevOps – сукупність практик, що поєднують процеси розробки (Dev) та експлуатації (Ops), що спрямовані на скорочення часу між фіксацією змін в системі та впровадженням цих змін у виробництво, забезпечуючи високу якість продукту.

Мета DevOps – скоротити етапи розробки в життєвому циклі та забезпечити гнучку співпрацю між різними командами. Методологія DevOps добре зарекомендувала себе в IT галузі. Процес розробки системи зі застосуванням практик DevOps може виглядати таким чином: формування вимог – ТЗ – розробка – введення в дію – отримання метрик роботи – формування вимог – і знову цикл повторюється. Таким чином нові вимоги добре простежуються, а метрики інтегруються до суміжних рівнів.

Оскільки DevOps має на меті функціональний режим роботи, для його провадження на практиці використовують різні набори інструментів, які відповідають одній або декільком наступним категоріям:

- building – інструменти розробки і збірки
- інструменти для автоматизації тестування
- інструменти для організації розгортання
- Runtime – інструменти
- інструменти для спільної роботи

IT-рішення - розуміється як скоординована поставка набору елементів (таких як програмні засоби, документація, навчання та супровід), необхідних для задоволення бізнес-потреби конкретного замовника.

Основними є такі принципи MSF.

1. Єдине бачення проекту, яке передбачає розуміння усіма зацікавленими особами цілей і завдань створення ПЗ.

2. Гнучкість – готовність до змін, що забезпечує можливість уточнення і зміни вимог у процесі розробки ПЗ, оперативного і швидкого реагування на поточні зміни умов проекту при незмінній ефективності управлінської діяльності.

3. Концентрація на бізнес-пріоритетах, що передбачає створення продукту з високим споживчим якістю і формування певної вигоди або віддачі. Для організацій, як правило, це отримання прибутку.

4. Заохочення вільного спілкування, що припускає відкритий і чесний обмін інформацією як усередині команди, так і з ключовими зацікавленими особами.

Універсальність моделі MSF визначається тим, що завдяки своїй гнучкості і відсутності жорстко встановлених зв'язків і процедур вона може бути застосована при розробці різних програмних додатків, які можуть використовуватися в бізнесі і повсякденному житті. Модель MSF базується на поєднанні двох моделей життєвого циклу програмних систем: каскадної і спіральної.

Завдяки своїй гнучкості дана модель може використовуватися для розробки широкого кола IT-рішень, вона охоплює весь життєвий цикл створення рішення, з самих ранніх етапів до запровадження. Модель процесів MSF поєднує в собі якості двох класичних моделей: каскадної і спіральної.

Процес MSF орієнтований на "віхи" (milestones). Віхи – ключові точки процесу розробки, які характеризують досягнення якого-небудь істотного результату. Модель процесів MSF враховує постійні зміни вимог до кінцевого продукту, процес розробки складається з коротких циклів і являє собою поступальний рух від найпростіших ранніх версій продукту до його остаточного виду.

ПРАКТИЧНА ЧАСТИНА

Завдання

Ознайомитися з теоретичним материалом. Згідно з наданим викладачем варіантом завдання створити командний проект у середовищі Visual Studio (не нижче версії 2017 року), використовуючи інструменти Azure DevOps Server та Azure DevOps Services. Налаштувати параметри створеного командного проекту за наведеним прикладом..

Приклад виконання

Для того, щоб розробити командний проект – використовуємо методологію розробки ПЗ Azure DevOps. Для підключення Azure DevOps запустіть інтегровану середу Visual Studio (не нижче версії 2017 року). На початковій сторінці виберіть клонування репозиторія (рис. 4.1).

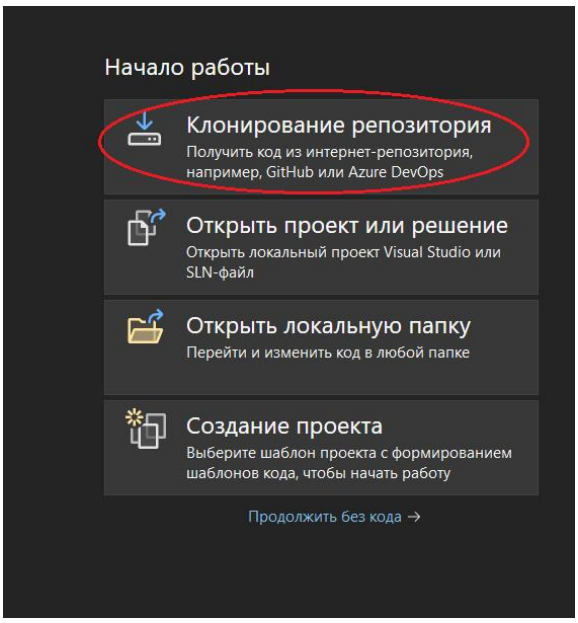

Рис. 4.1 – Початкова сторінка Visual Studio 2022

Якщо є посилання на готовий репозиторій, то вписуємо URL в строку. Якщо немає, то створюємо через Azure DevOps (рис. 4.2).

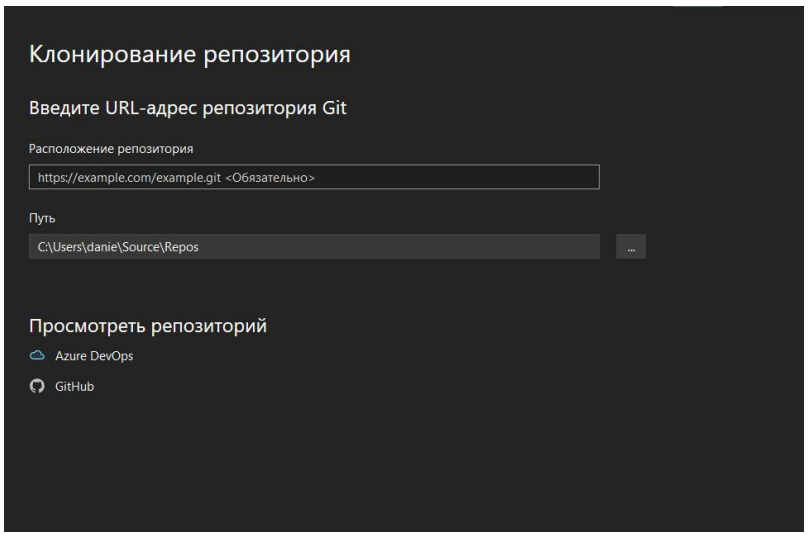

Рис. 4.2 – Клонування репозиторія

Якщо у діалоговому вікні Підключення до Azure DevOps не знаходить потрібний сервер, то натисніть кнопку Додати чи Створити заліковий запис в Azure DevOps (рис. 4.3).

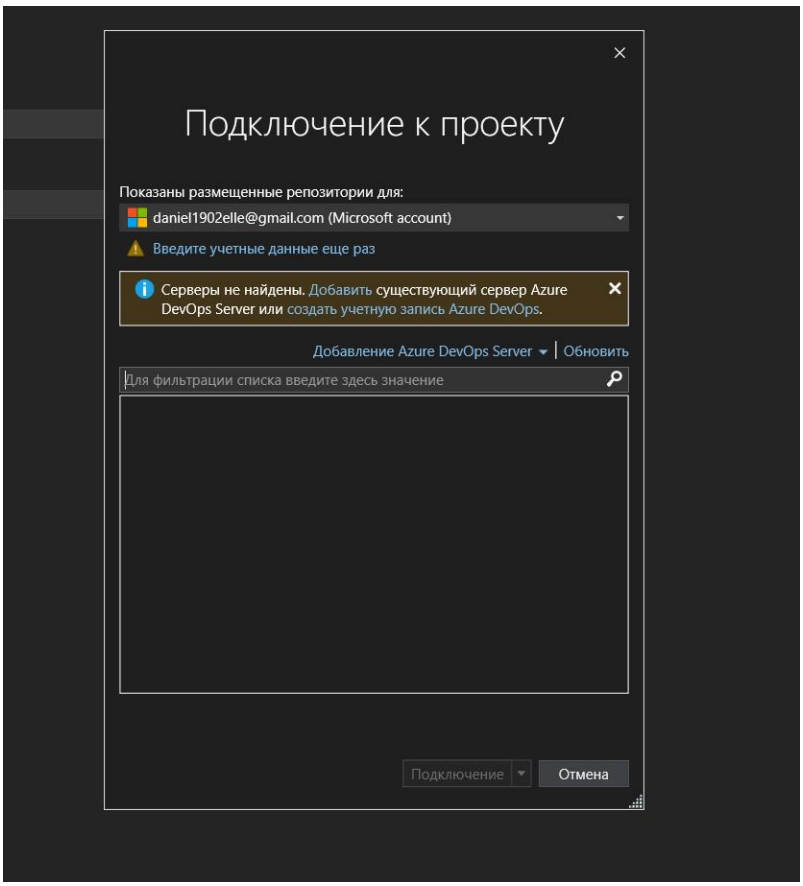

Рис. 4.3 – Діалогове вікно Підключення до Azure DevOps

Після реєстрації користувач може обрати будь-яку країну, обираємо Україну і натискаємо продовжити (рис. 4.4.).

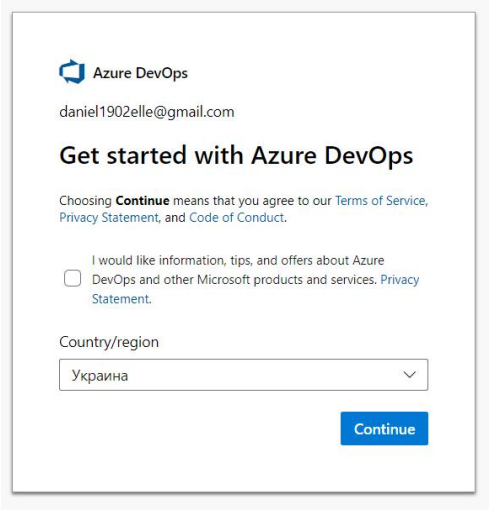

Рис. 4.4 – Діалогове вікно створення облікового записа

Проходимо далі реєстрацію в Azure DevOps, заповнюючи певні пункти діалогового вікна.(рис. 4.5).

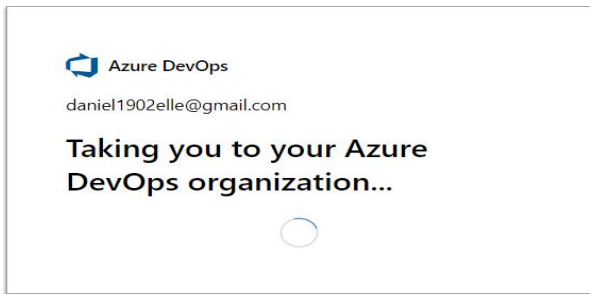

Рис. 4.5 – Проходимо реєстрацію

Після реєстрації одразу з'являється можливість створити проект. Згідно р розглянутими у попередніх лабораторних завданнях, створюємо проект «Інформаційна система «Відділ кадрів підприємства». Після створення проекту з'являється вікно з назвою проекту «Відділ кадрів», але цей проект є порожнім. (рис 4.6).

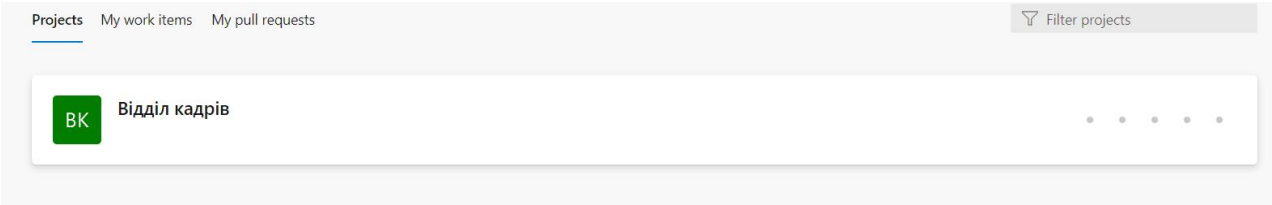

Рис. 4.6 – Створений проект «Відділ кадрів»

Після того, як було створено проект, відкриваємо його і ліворуч з'являється віно з багатьма функціями, але перш за все потрібно створити репозиторій, тому обираємо вікно "Repos" і тоді вже "Files" (рис. 4.7).

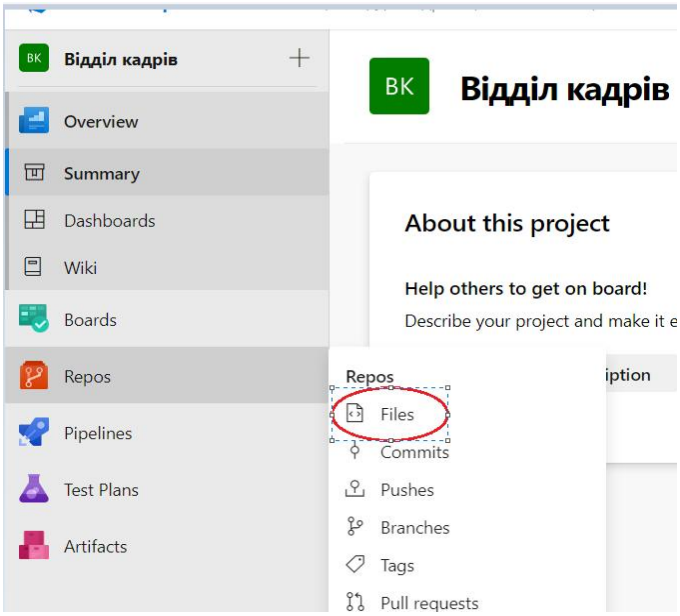

Рис 4.7 – Створення репозиторія

Коли створено репозиторій, в діалоговому вікні відображено створені файли майбутнього сервера (рис. 4.8).

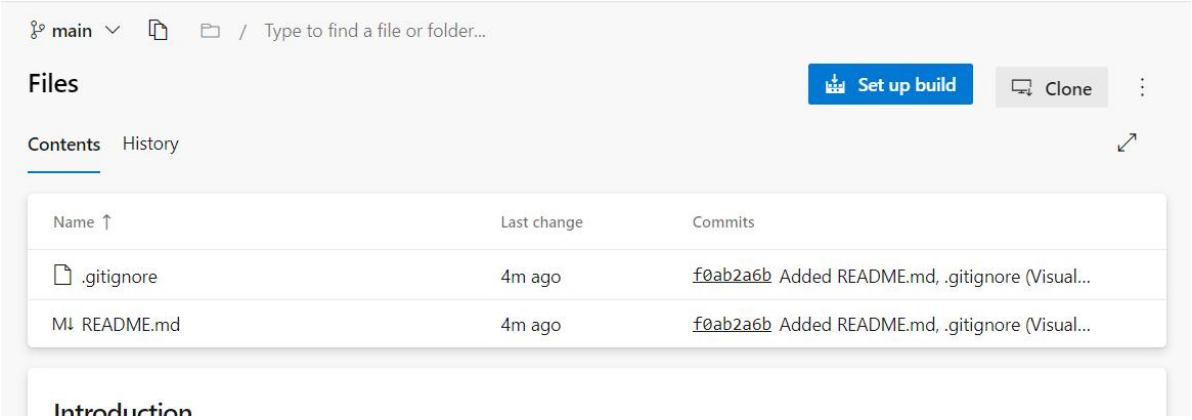

Рис. 4.8 – Створені файли майбутнього сервера

Подальшим кроком для здійснення клонування користувачу потрібно копіювати ці файли, після чого відкриється вікно клонування (рис. 4.9.).

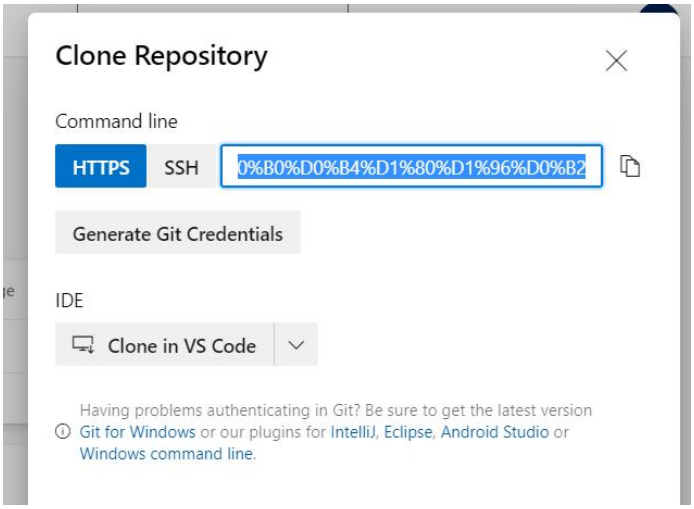

Рис. 4.9 – Вікно клона

Потім нам потрібно обрати область в якій будемо копіювати файли і для цього відкриваємо перелік (рис. 4.10).

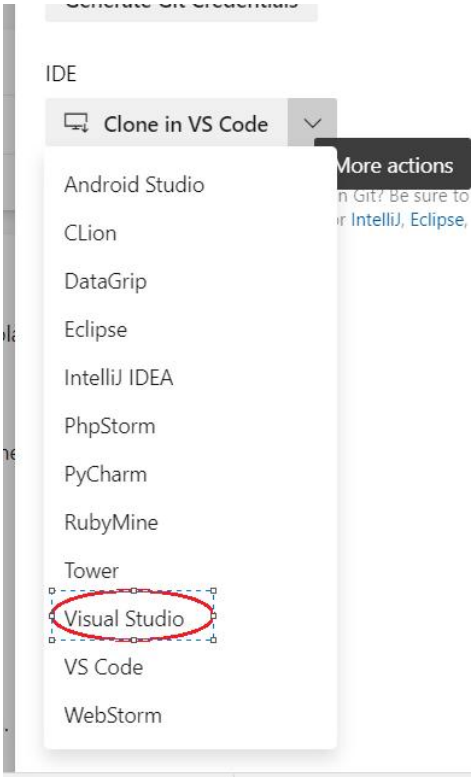

Рис. 4.10 – Перелік областей

З наведеного переліку обираємо область Visual Studio, щоб відкрити репозиторій в цьому середовищі (рис. 4.11).

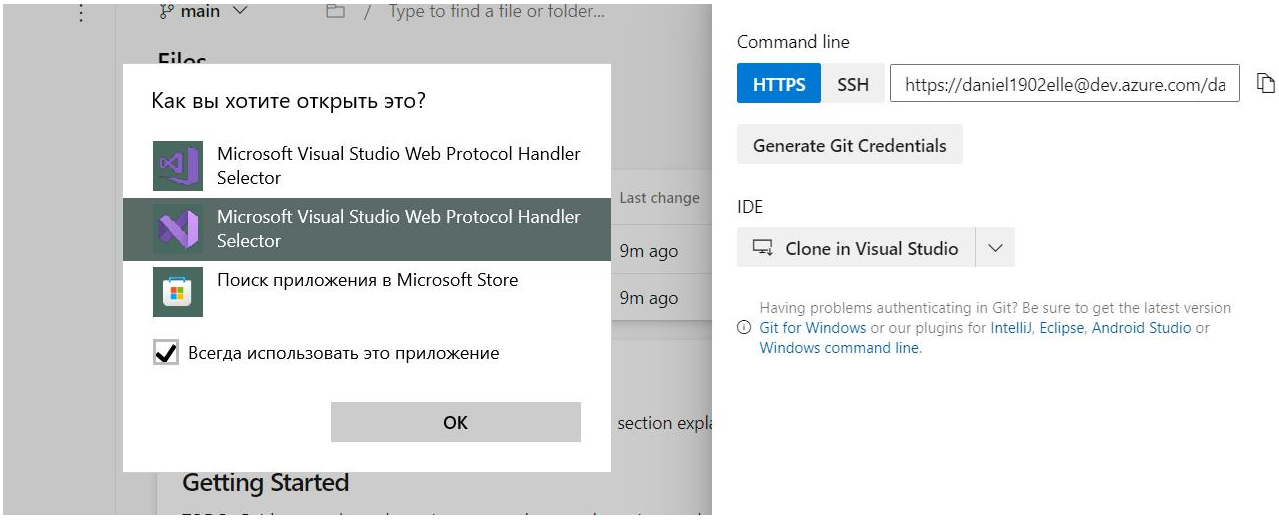

Рис. 4.11 – Обираємо спосіб відкриття репозиторія

Коли користувач обрав серед переліку саме середовище Visual Studio, то репозиторій користувача є нарешті створеним (рис. 4.12).

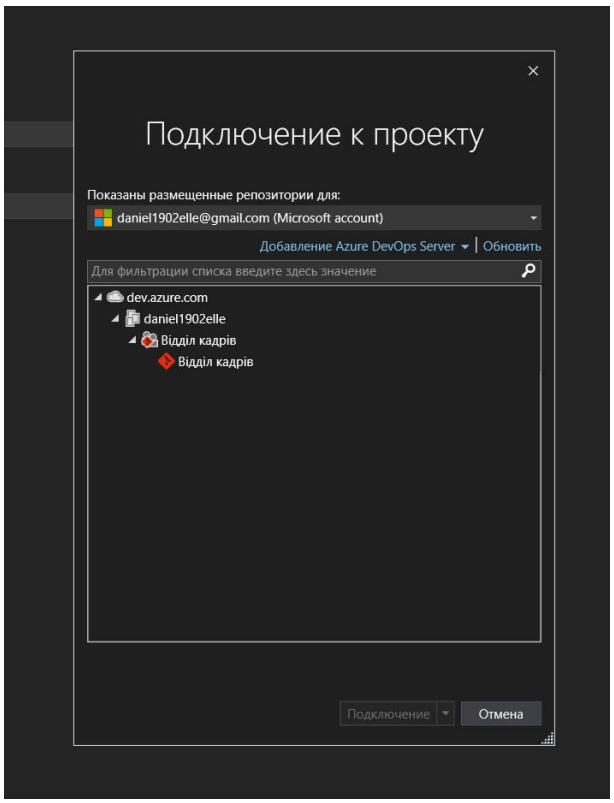

Рис. 4.12 – Створений репозиторій

Після створення репозиторію, подальшим кроком є клонування, яке потрібно здійснити у середовищі Visual Studio (рис. 4.13).

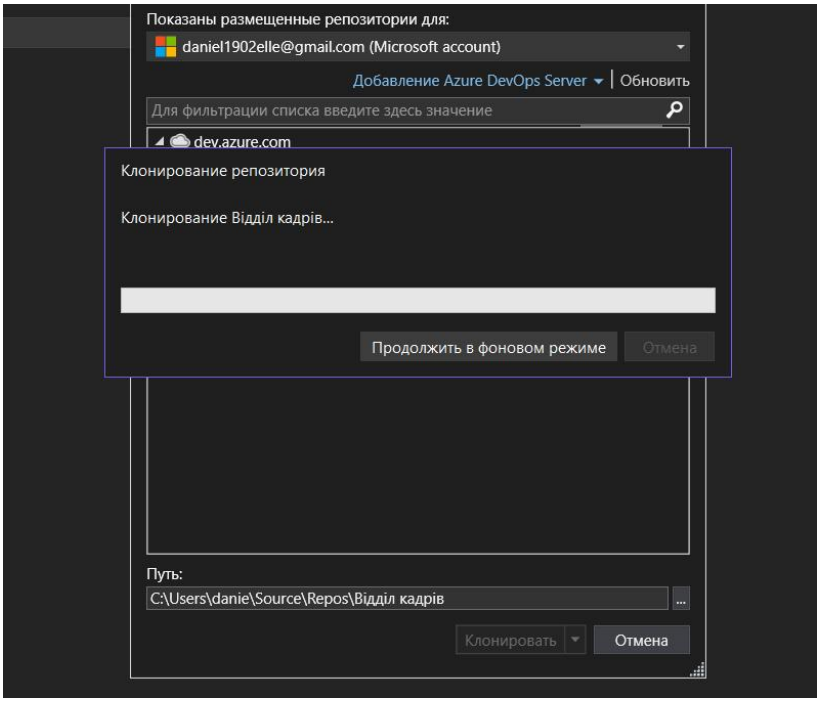

Рис. 4.13 – Клонування репозиторія в Visual Studio

Після успішного клонування репозиторія з’являється вікно, яке відображає створений новий проект (рис. 4.14).

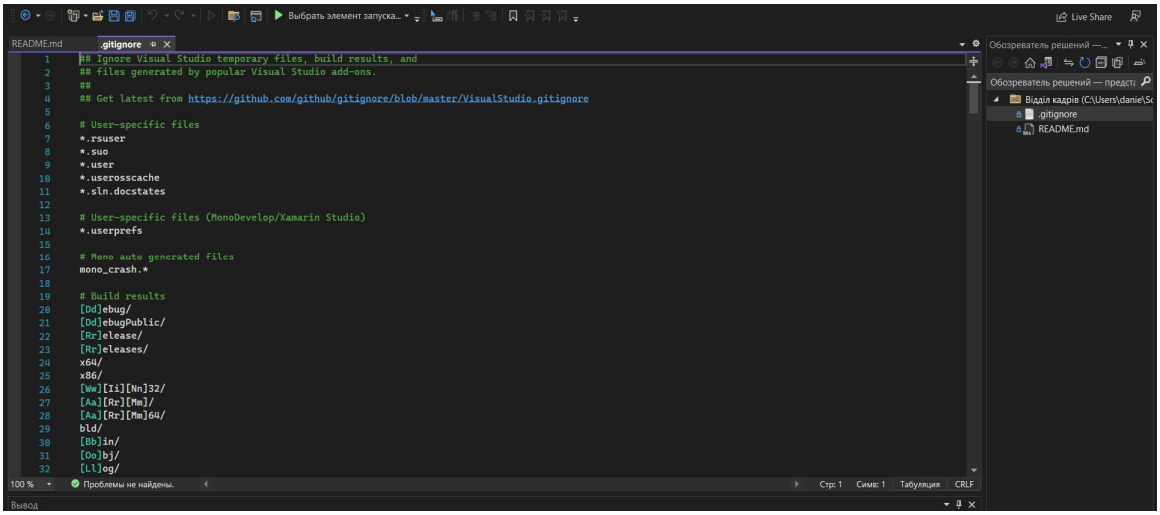

Рис. 4.14 – Вікно створеного проекту

Коли репозиторій було успішно перенесено в область Visual Studio, то користувач повертається до Azure DevOps і додає там нових членів команди до проекту (рис 4.15).

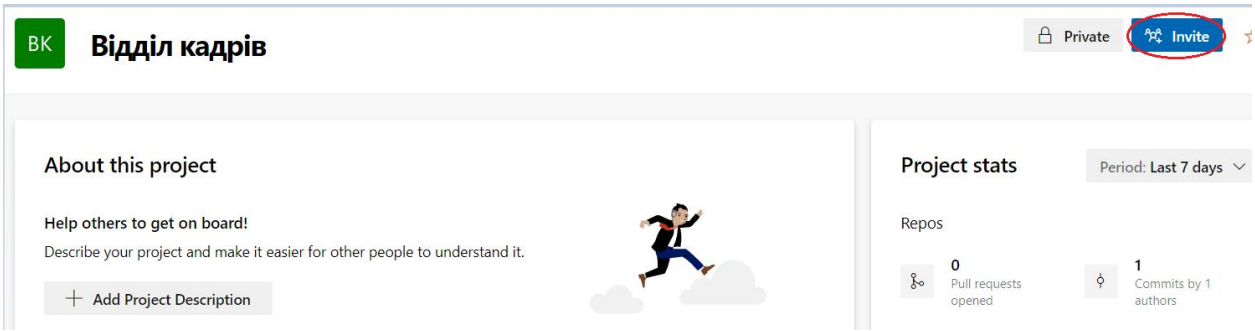

Рис. 4.15 – Додавання члена команди

Щоб додати члена команди в проект, треба додати його e-mail адресу на яку буде відправлено лист з зазначенням посилання. Якщо новий член команди перейде за цим посиланням, в проект буде додано нового члена команди (рис 4.16).

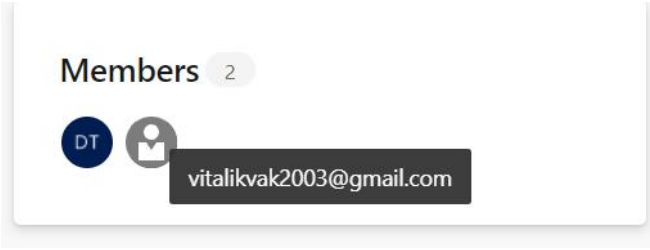

Рис. 4.15 – Віконце з доданими членами команди

Коли в проект додані всі члени команди, налаштуємо відображення фото створеного командного проекту (рис. 4.16).

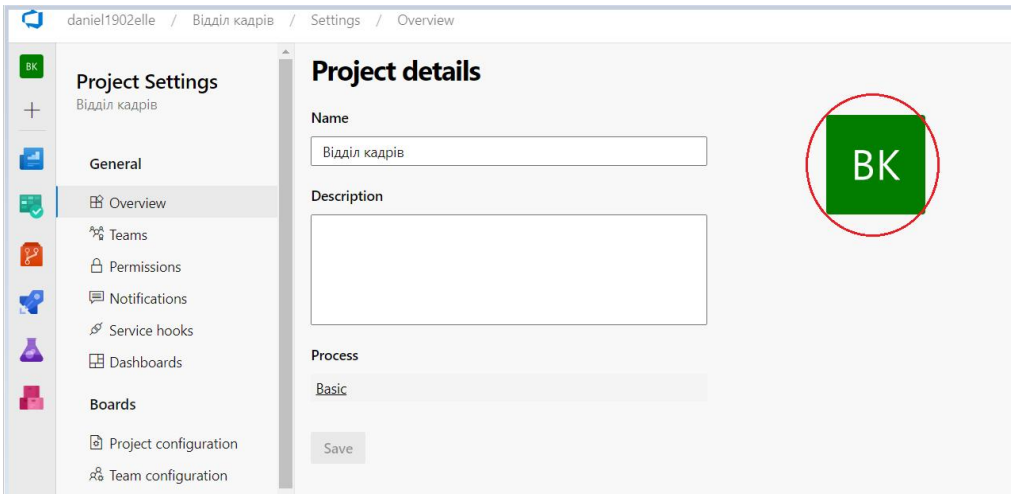

Рис. 4.16 – Налаштування проекту

Щоб змінити фотографію проекту, треба натиснути на квадратик, де знаходяться перші літери назви нового проекту. Після цього вам відкриється вікно, яке дозволяє нам обрати фотографію для проекту (рис 4.17).

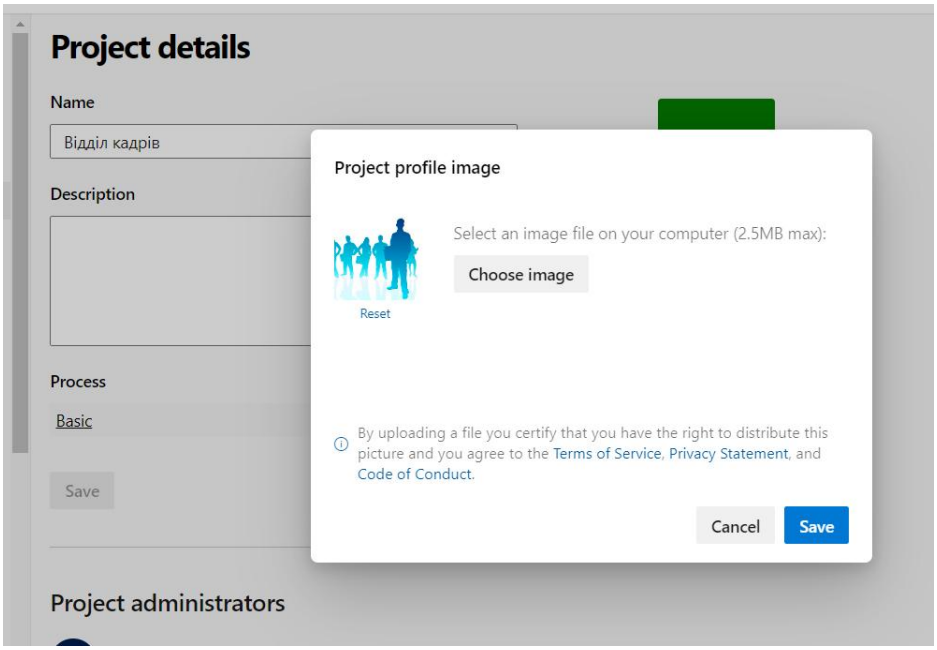

Рис. 4.17 – Зміна фотографії проекта

Коли змінено фотографія проекту, наступним кроком треба додати стисли опис проекту (рис. 4.17).

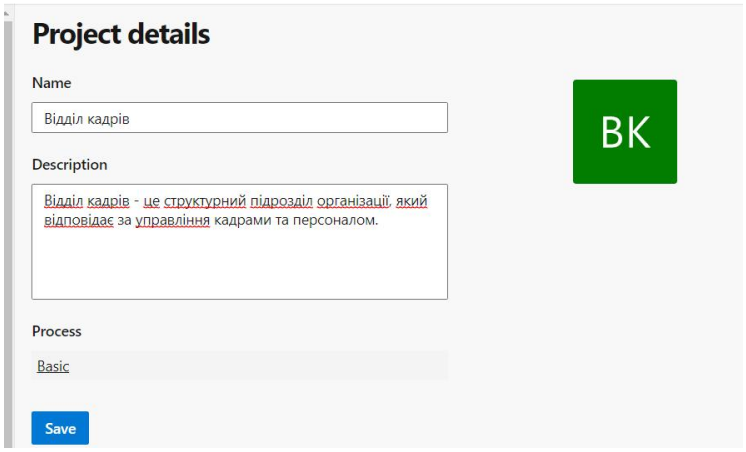

Рис. 4.17 – Додавання опису проекту

Так буде виглядати створений проект «ІС «Відділ кадрів» після додавання фотографії проекту, стислого опису про призначення проекту і членів команди, які задіяні у проекті (рис. 4.18).

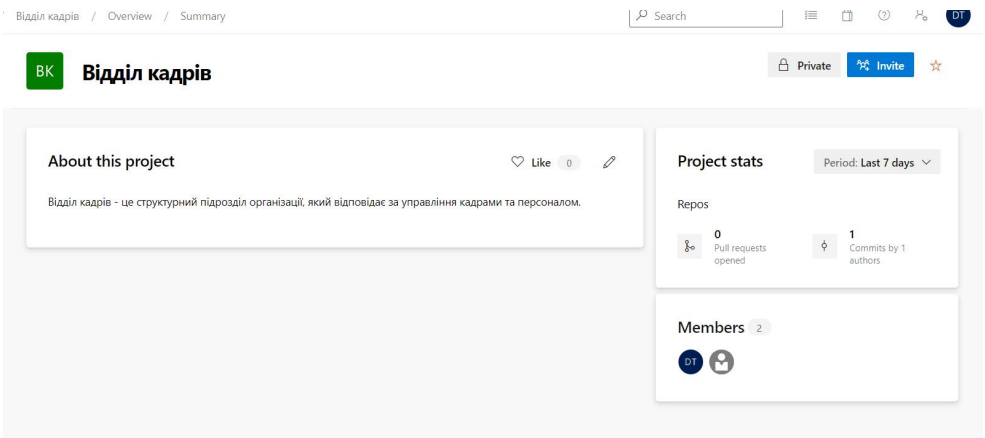

Рис. 4.18 – Відображення налаштованого нового проекту «ІС «Відділ кадрів»

Наступним кроком буде створення спитнів у проекті. Для того, щоб створити спринт треба перейти у вкладку Дошки – відставання.(рис 4.19)

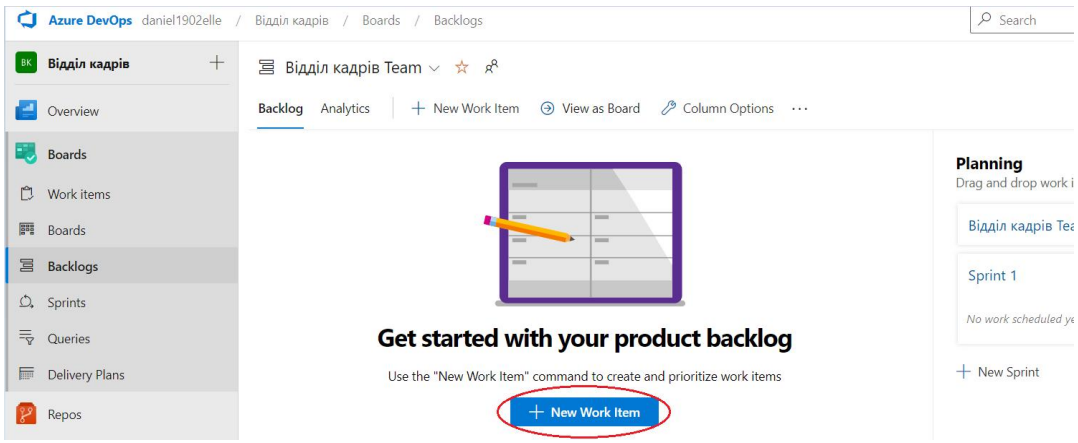

Рис. 4.19 – Вікно відставання для створення спинтів

Для створення нового спринту переходимо до вікна проекту (рис. 4.20).

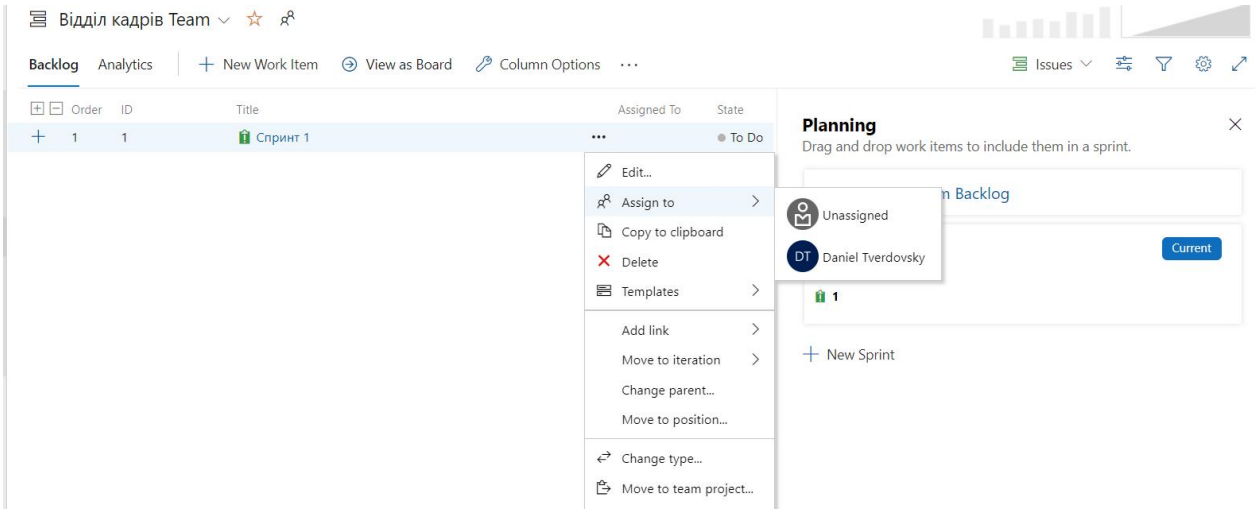

Рис 4.20 – Створення спринта

Щоб створити спринт необхідно надати комусь з членів команди проекту завдання для якого і потрібно створити в подальшому спринт (рис. 4.21).

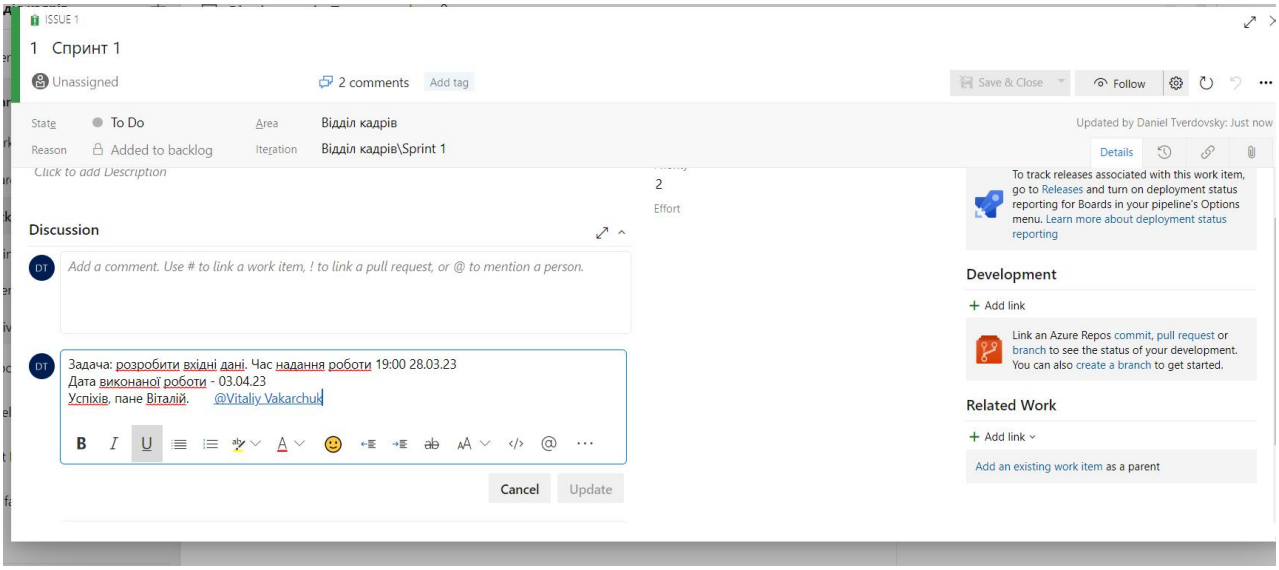

Рис 4.21 – Вікно призначення завдання для члена команди проекту

Кожне з створених завдань повинно мати план виконання. Тоді це завдання вважається повноцінним. Щоб завдання було повноцінним, треба створити план (рис. 4.22).

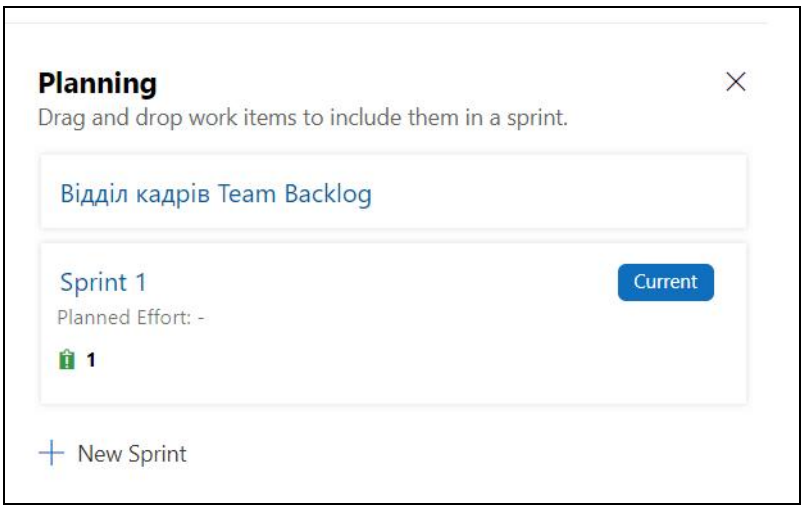

Рис. 4.22 – Планування створеного завдання

Після створення плану завдання, можно вважати спринт готовим (рис. 4.23).

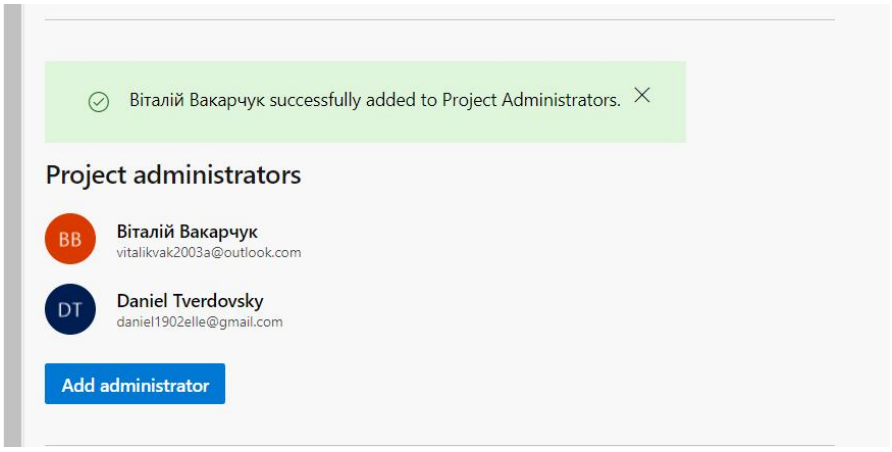

Рис. 4.23 – Створений новий спринт

Для відстеження виконання завдання надається можливість перегляду та перевірки (рис. 4.24)

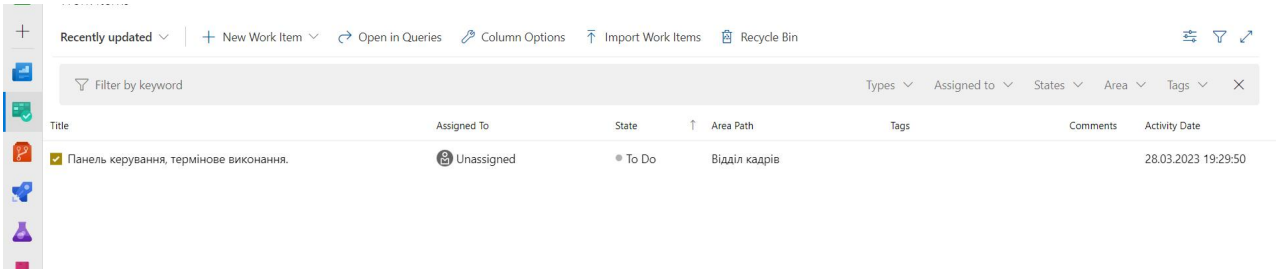

Рис. 4.24 – Перевірка завдання

Якщо необхідно додати або змінити щось у створеному завданні, то це можливо здійснити у вікні (рис. 4.25).

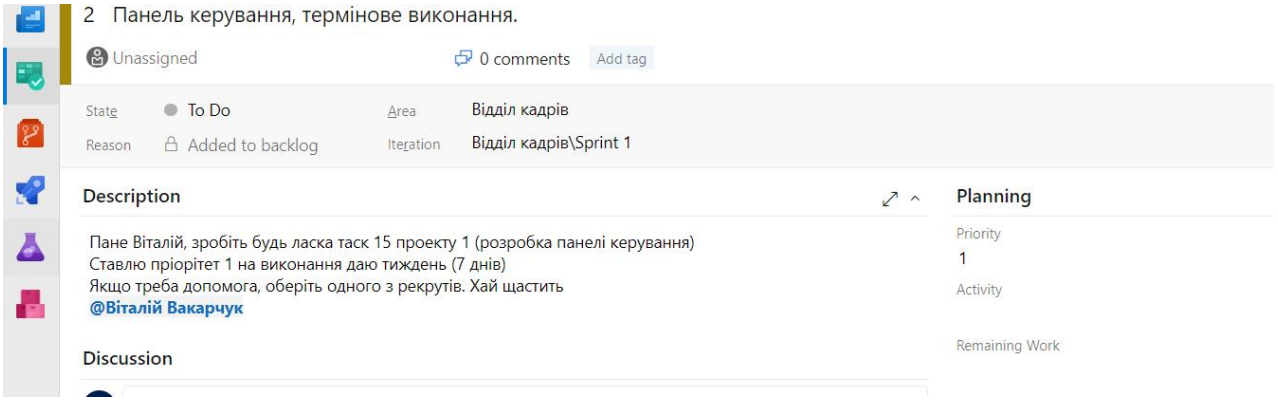

Рис. 4.25 – Внесення змін до завдання

Звіт

Звіт з лабораторної роботи повинен складатися з текстового документу «Протокол лабораторної роботи», що містить:

- Назву роботи, мету;
- Хід виконання роботи (графічні результати виконання роботи (скріпи));
- Відповіді на контрольні питання;
- Висновки.

Захист звіту з лабораторної роботи полягає в пред'явленні викладачеві отриманих результатів, демонстрації отриманих навичок і відповідях на питання викладача.

Контрольні питання

1. Основні принципи методології MSF – Microsoft Team Foundation Server (Azure DevOps)
2. Чим визначається універсальність моделі MSF / DevOps?
3. Яка модель життєвого циклу програмної системи використовується в MSF/ DevOps?
4. Інструменти DevOps?
5. Наведіть призначення інструментів розробки і збірки DevOps – building?
6. Поясніть призначення інтеграції в методології MSF?
7. Як можна масштабувати команду, що використовує методологію MSF?

Лабораторна робота №5

Розробка програмного продукту засобами Azure DevOps

Мета роботи: Отримати практичні навички в розробці користувальницьких вимог до програмного продукту за допомогою Visual Studio та створення робочих елементів проекту.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Основним засобом розробки в Visual Studio є інтегрована середовище розробки (IDE). IDE-середовище інтегрована із засобами модульного тестування та забезпечує можливості виявлення неефективного, небезпечного чи погано написаного коду, управління змінами і модульне тестування як коду, так і бази даних.

Visual Studio підтримує безліч модулів, що підключаються, що розширюють його функціональні можливості. Зокрема, модуль Team Обозреватель дозволяє клієнту Visual Studio підключатися до Azure DevOps для підтримки операцій управління версіями, відстеження роботи, складання та тестування.

Microsoft Team Foundation Server використовується для координації роботи з кодом з іншими учасниками команди для розробки проекту програмного забезпечення. Крім того, можливо керувати роботою, призначеною вам, вашій команді або вашим проектам. Командний обозрівач – це модуль, що підключається, який встановлюється разом з Visual Studio. Розробники можуть ефективно співпрацювати за допомогою командних браузеров, підключених до проектів, розміщених на Azure DevOps Services або локальної Azure DevOps Server.

Azure Boards – це веб-служба, яка дозволяє командам планувати, відстежувати та обговорювати роботу в рамках всього процесу розробки, а також підтримує гнучкі методології, включаючи Scrum. Azure Boards надає платформу, що настроюється, для управління робочими елементами, дозволяючи командам ефективно співпрацювати і оптимізувати свій робочий процес. Керівники проектів, які хочуть використовувати знайомі засоби, можуть імпортувати та експортувати запити робочих елементів у Microsoft

Office Excel та з неї, а також імпортувати та експортувати робочі елементи за допомогою .csv файлів.

Для підтримки зусиль для відстеження роботи можна використовувати Microsoft Excel. Ви можете працювати в режимі «мережа», де ви підключені до Azure Boards або Azure DevOps Server. Або працювати в автономному режимі, де ви звертаєтеся до локального комп'ютера та документа.

Завдяки Azure Boards ви отримуєте переваги повної інтеграції із платформою Azure DevOps. Azure DevOps призначений для забезпечення комплексної можливості трасування, відстеження роботи від вимог до розгортання. Отримуйте аналітичні відомості на кожному етапі прийняття рішень та розгортання програмного забезпечення.

В якості клієнтського інтерфейсу для Azure DevOps можна використовувати Microsoft Excel і Microsoft Project (рис. 5.1).

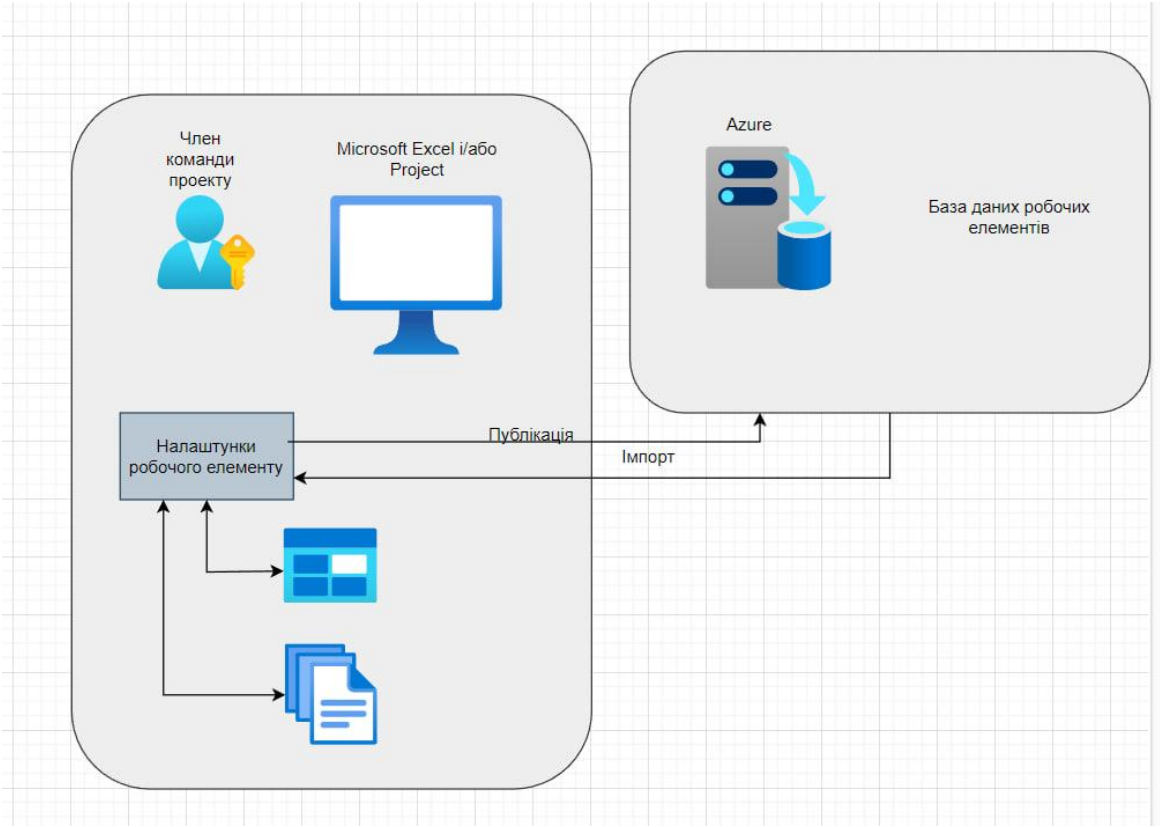

Рис. 5.1 – Клієнти Microsoft Office для Azure DevOps

Серед інструментів Azure DevOps є інтегрований засіб Microsoft Excel, завдяки якому можливо надалі працювати з проектом. Але для початку у веб-

додатку треба переконатись, щоб ви обрали певний проект, після чого треба перейти в Дошки – Запроси і вже там обрати всі необхідні інструменти.

Інструмент main, який можна використовувати для імпорту даних відстеження роботи, експортованих з іншого місця, це Microsoft Excel. Excel підтримує публікацію неструктурованого списку робочих елементів або ієрархічного дерева пов'язаних батьківських та дочірніх робочих елементів.

ПРАКТИЧНА ЧАСТИНА

Завдання

Згідно з обраним варіантом завдання створити інтерфейс користувача, який задовольнить вимогам зазначеним в технічному завданні (лабораторна робота №1). Для розроблюваного проекту створити користувальницькі вимоги, використовуючи Microsoft Excel та Azure Boards, який інтегровано у середовище розробки Visual Studio. Налаштувати параметри створеного командного проекту.

Приклад виконання

Розглянемо виконання цієї лабораторної роботи на прикладі створення проекту «ІС «Відділ кадрів підприємства»

Першим кроком буде створення користувальницьких вимог за допомогою Microsoft Excel. Щоб додати чи змінити робочі елементи за допомогою Excel, необхідно підключити Лист з Книги до проекту. Завантаження цього під'єднання пов'язує документ до проекту Azure DevOps для обміну інформацією (рис. 5.2).

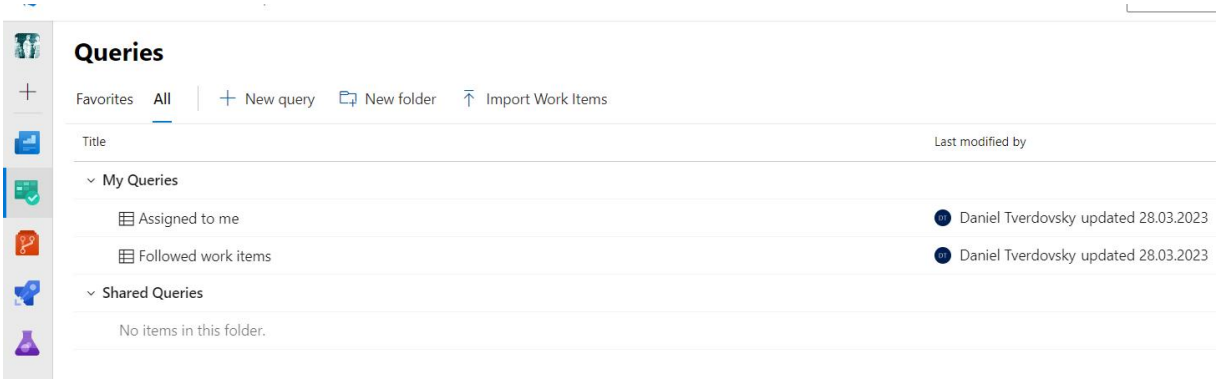

Рис.5.2 – Вікно запитів

Після того, як користувач перейде до вікна запросів, нам потрібно обрати наш запрос та натиснути на три крапки (…), щоб випав перелік інструментів в якому обираємо Excel (рис. 5.3).

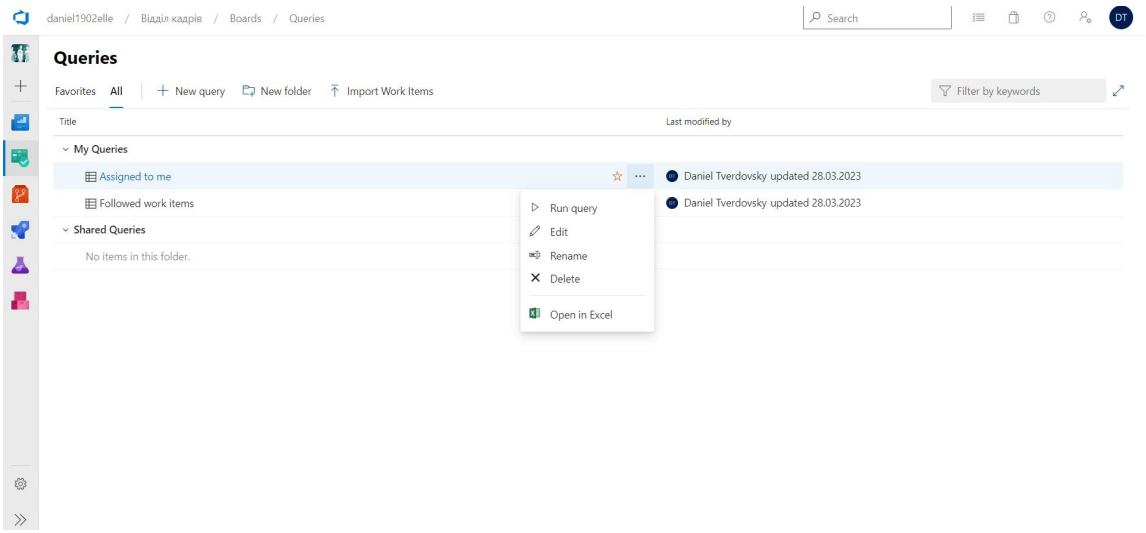

Рис. 5.3 –Перелік з інстументом Excel

Можливо використання декількох Листів в Книзі Excel для роботи з різноманітними вхідними спискам чи переліком запитів. Однак для кожної Книги можливо під'єднання тільки одного проекту (рис. 5.4).

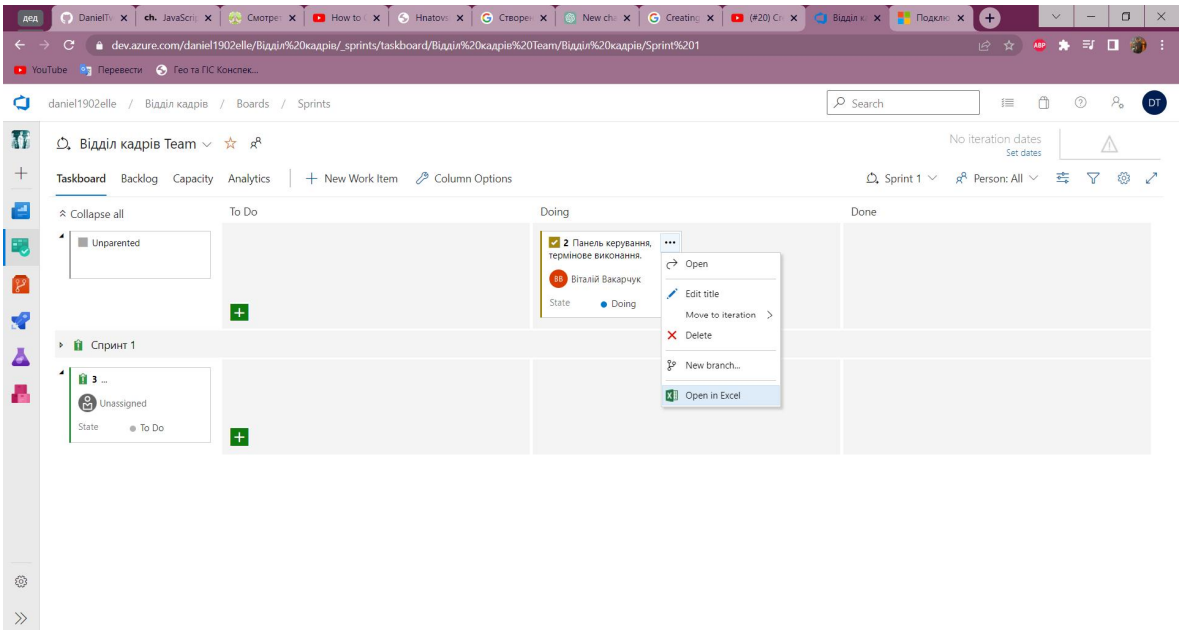

Рис 5.4 – Під'єднання проекту до таблиці Excel

Розглянемо масове додавання чи змінення в Azure Boards робочих елементів за допомогою інструменту Microsoft Excel.

Якщо потрібно додати чи змінити багато робочих елементів, використовуючи саме інструмент Microsoft Excel можливо заощадити час. Excel підтримую додавання робочих елементів, оновлення існуючих робочих елементів, додавлення посилань та вложень в декількох робочих елементів та багато інших функцій. Також надається можливість використовувати

особисті функції Excel для підтримки інших дій, таких як підсумовування даних стовбця, копіювання та вставка строк, заповнення данних в осередках та багато іншого. Для виконання цих діє існує низка типу запитів (рис. 5.5).

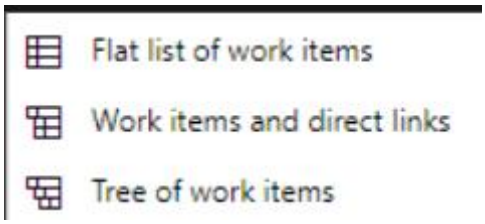

Рис. 5.5 – Типи запитів

Azure Boards підтримує три типу запитів. Значок поблизу з кожним запитом вказує тип запиту. Перші два типу запитів: неструктурований список робочих елементів та робочі елементи. Також прямі посилання імпортуються, як запити неструктурованого списку.

Тільки запити дерева робочих елементів імпортуються у вигляді древовиглядкового списку. Запити прямих посилань імпортуються в Excel у вигляді неструктурованого списку, так як зміни декількох типів посилань не підтримується в Excel.

Посилання “батьки-дочірні” чи інші типи зв’язків топології дерева підтримується створенням ієрархічної структури невиконаної роботи. Типи робочих елементів які беруть участь в ієрархії, відрізняються в різноманітних процесах та показані на наступному малюнку (рис. 5.6).

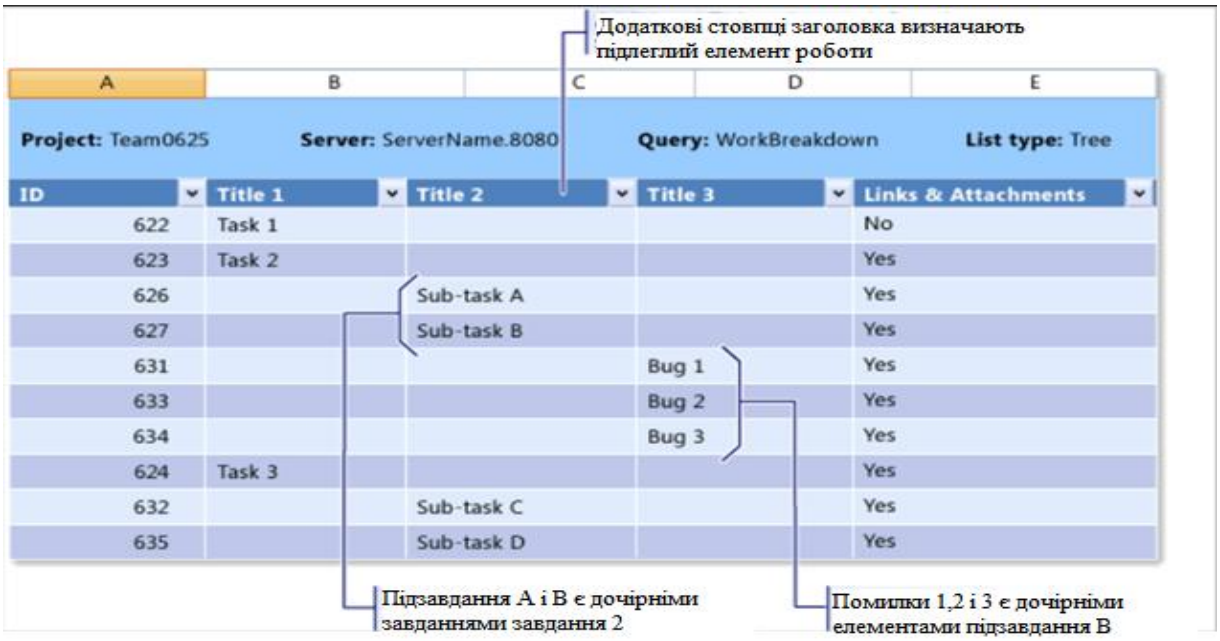

Рис. 5.6 – Приклади таблиці Excel з завданням

В Excel можна відкрити в будь-який запит, певний в Azure Boards. Сюди входить запити, певний в розділи “Мої запити” та “Спільні запити”. Однак, якщо ви плануєте продемонтрувати спільний доступ до книги з іншими учасниками групи, вам слідує працювати тільки зі спільним запитом. Інші учасники групи не можуть використати Книгу чи Лист Excel, якщо вони основані на особистому запиті, що зберігається в папці “Мої запити”.

Використовування функцій Excel.

Ви можете використовувати більшість функції Excel при роботі зі списком робочих елементів. Наприклад, можливо використовувати насту Excel пні функції:

1. Форматування осередку чи використовування умовного форматування до осередку чи стовпцю;
2. Вирізати та вставити з одного осередка в інший осередок;
3. Вирізати та вставити в іншу строку;
4. Сумування стовпця та додавання інших формул;
5. Заповнення осередків;
6. Фільтрація;
7. Додавання декількох Листів в Книгу.

Кожен з Листів в Excel може утримувати різні вхідні списки чи запити. Однак, всі Листи в Книзі повинні під’єднатись до одного проекту в організації в колекції проектів.

Задачі, які можливо виконувати на Листі Excel:

- додавання тегів та масові оновлення робочих елементів;
- додавання тегів робочих елементів для класифікації та фільтрації списків та дошок;
- додавання полів тегів на Лист;
- додавання декількох тегів, розділених крапкою з комою(;).
- додавати простий текст в полі форматування тексту, але при цьому масове оновлення декількох робочих елементів форматування в існуванні робочих елементів може бути втрачено.
- працювати в автономному режимі, а потім повторно підключити та опублікувати зміни.

Задачі, які НЕ можливо виконувати на Листі Excel:

- видаляти робочі елементи ;

– змінювати тип робочого елементу існуючого робочого елементу;
– переміщати робочі елементи в інший проект;
– імпортувати чи оновлювати кроки тестового випадку чи інші тестові артефакти;
– додавати робочі елементи в будь-який інший стан, окрім нового стану;
– додати в потік обговорення робочого елементу;
– пов’язувати з видаленими робочими елементами.

Інструкція підключення дошок Excel.

Імпорт робочих елементів у вигляді плоского списку:

1. Відкрийте Excel та підключить до проекту Azure Boards. Використовуйте один з чотирьох методів, представлених в розділі “Підключення проекту Azure DevOps к Excel”;

Додаток: При підключенні к Azure Boards в хмарі автоматично обирається колекцію командних проектів, так як з вашого Azure DevOps Services організації пов’язана тільки одна колекція. При підключенні до Azure Boards на локальний сервер перед вибором проекта оберіть колекцію командних проектів.

2. В Excel почніть з порожнього листа;

3. Оберіть створений список на стрічці “Команда”(рис. 5.7);

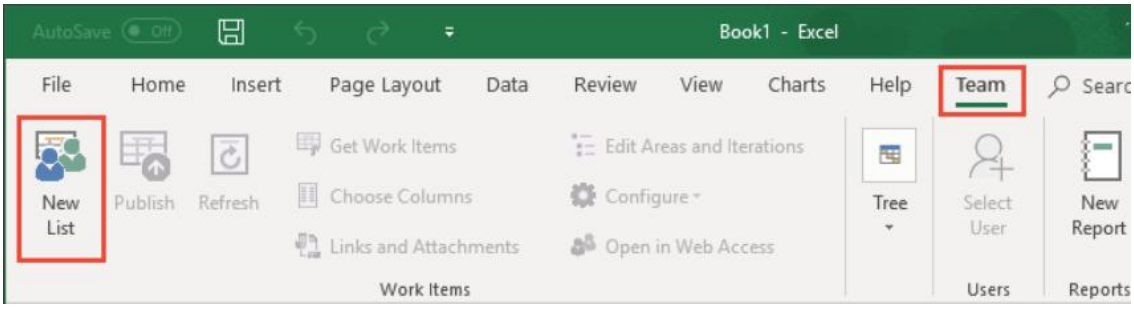

Рис. 5.7 – Створення списку на стрічці “команди”

4. В діалоговому вікні “Новий список” оберіть “Вхідний список” (рис. 5.8);

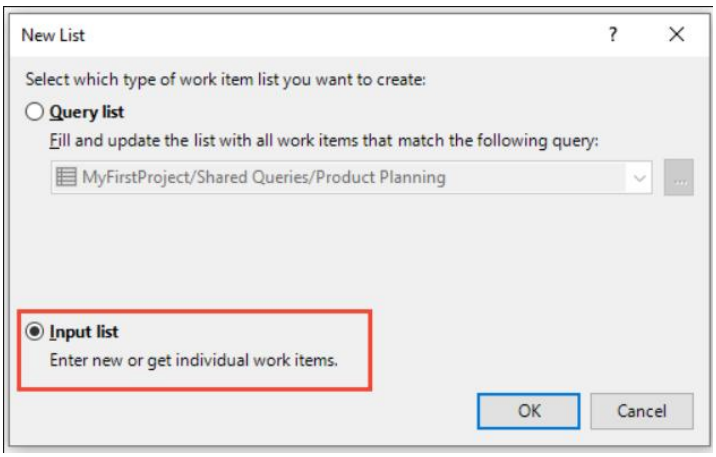

Рис. 5.8 – Створення вхідного списку

Зараз Лист прив'язано до проекту як вхідний список (Запит:[Hi]), плоский список (рис. 5.9);

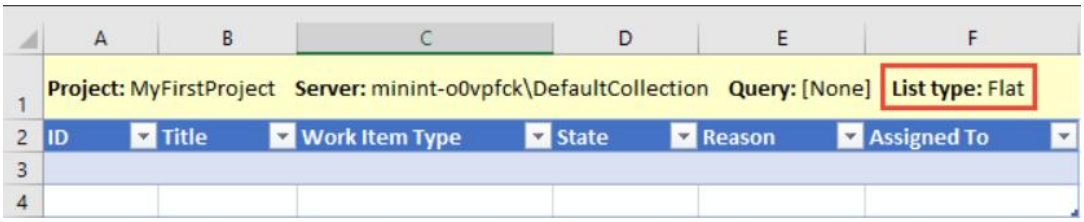

Рис. 5.9 – Проект, як вхідний список

5. Вкажіть назву доданих робочих елементів та їх тип робочого елементу (зверніть увагу, що поля "Стан" та "Причина" автоматично заповнюються значеннями по замовчуванню після вибору типу робочого елементу) (рис. 5.10);

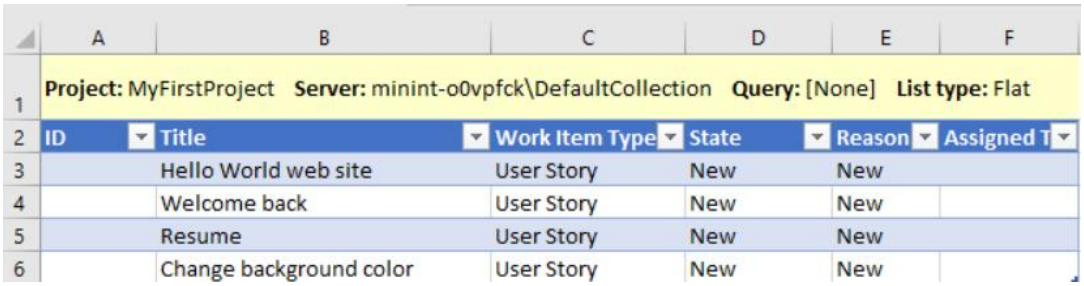

Рис. 5.10 –Створені робочі елементи проекту

6. Опублікуйте лист (переконайтесь в тому, що курсор, який знаходиться в осередку з даними. В гіршому випадку кнопка “Опублікувати” може відображатись відключеною) (рис. 5.11);

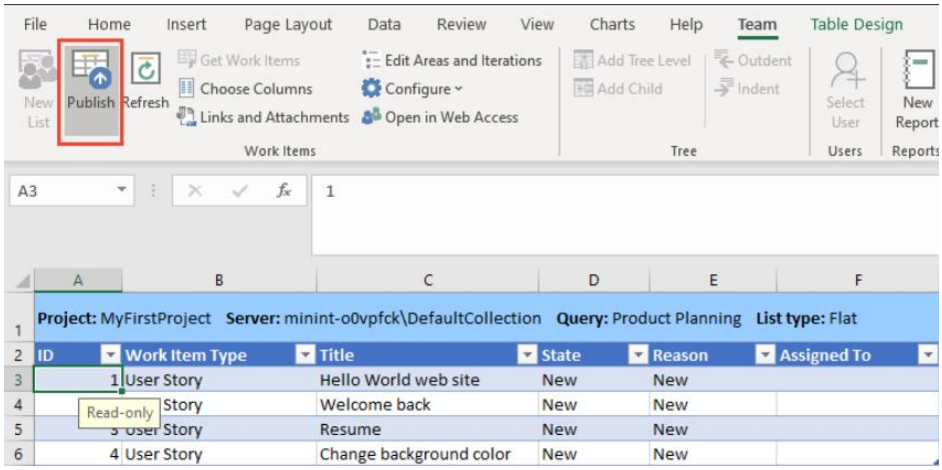

Рис. 5.11 – Публікація Листа Excel

Зверніть увагу, що зараз робочим елементам присвячені ідентифікатори(рис. 5.12).

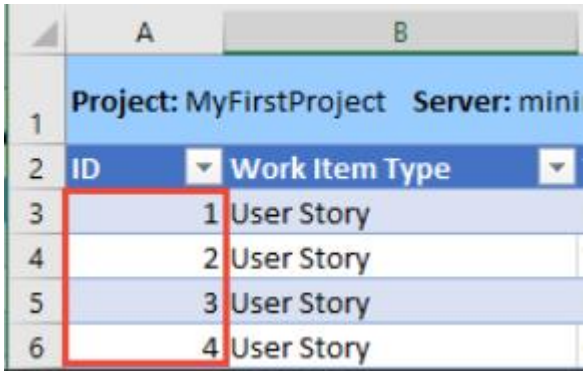

Рис. 5.12 – Ідентифікатори робочих елементів

Щоб присвоїти значення іншим полям, відкрийте вкладку “Вибір стовпця”, додайте поля, зробіть призначення та опублікуйте зміни.

Щоб відкрити робочий елемент для додавання доповнення відомостей, оберіть робочий елемент, який потрібно відкрити, та натисніть кнопку “Відкрийте у веб-застосунку”. Перед тим, як це зробити, обов’язково опублікуйте внесені зміни (рис. 5.13).

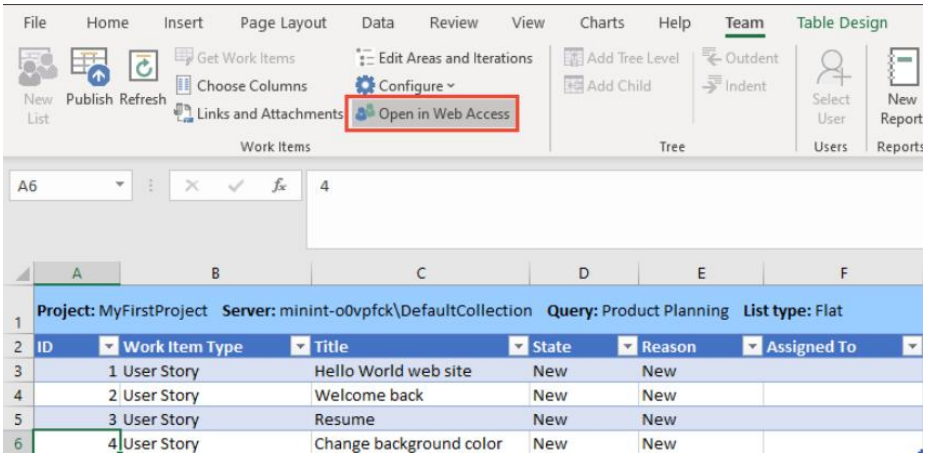

Рис. 5.13 – Відкривання у веб-застосунку

Звіт

Звіт з лабораторної роботи повинен складатися з документу «Протокол лабораторної роботи», що містить:

- назву роботи, мету;
- хід виконання роботи (графічні результати виконання роботи (скріпи));
- відповіді на контрольні питання;
- висновки.

Захист звіту з лабораторної роботи полягає в пред'явленні викладачеві отриманих результатів, демонстрації отриманих навичок і відповідях на питання викладача.

Контрольні питання

1. Наведіть основні функції інструмента Azure Boards?
2. Як додати робочі елементи в Azure Boards за допомогою інструменту Microsoft Excel?
3. Які типи запитів підтримує Azure Boards?
4. Які функції Excel можливо використовувати при роботі зі списком робочих елементів?
5. Які задачі дозволено виконувати на Листі Excel?

Лабораторна робота №6

Аналіз вимог на розробку, проектування архітектури ПП. Архітектурне моделювання ПП засобами Azure DevOps

Мета роботи: Отримати практичні навички архітектурного проектування програмних додатків при розробці схем варіантів використання UML і схем класів UML. Вибір архітектурних шаблонів. Формування команди та розподіл ролей. Проектування схеми БД. Розробка діаграм прецедентів, ієрархії класів, послідовності, розгортання.

ТЕОРЕТИЧНІ ВІДОМОСТІ

При командній розробці програмного забезпечення важливими питаннями є планування робіт, складання розкладу, управління областю проекту, комунікацій, складання звітів, аналіз і постійне вдосконалення процесу . Для вирішення цих питань TeamFoundationServer пропонує наступні інструменти:

- шаблон процесу, який визначає процес, який використовується командним проектом;
- керівництво по процесу, яке містить опис шаблонів процесу;
- колекція командних проектів, яка представляє собою контейнер для декількох командних проектів;
- командний проект, який зберігає і організує дані про всьому життєвому циклі розробки програмного забезпечення;
- відстеження робочих елементів, яке дозволяє відстежувати стан робочих елементів та іншу інформацію, пов'язану з ними;
- портал проекту / панелі моніторингу, на яких надається інформація по проекту для всіх членів команди;
- елементи планування для управління списками вимог як на рівні проекту, так і на рівні ітерації;
- звітність, яка дозволяє формувати звіти в ході життєвого циклу проекту;
- інтеграція з Microsoft Project і Microsoft Excel для управління проектами з середовища Microsoft Office.

Розробка програмного забезпечення починається зі створення командного проекту. Командний проект містить інформацію про кожен крок життєвого циклу розробки програмного забезпечення, включаючи вимоги користувачів, завдання, тестові випадки, помилки, перешкоди, побудови.

При створенні командного проекту необхідно задати його ім'я, опис, визначитися з шаблоном процесу, вимогами до системи контролю версій і необхідністю створення порталу для проекту.

Після створення командного проекту керівник формує команду. Для кожного члена команди керівник проекту визначає доступ до проекту в цілому і окремим артефактів, які важливі для роботи кожного члена команди. Для структурування проекту використовуються області та ітерації. Області можуть визначатися виходячи з певного функціонального призначення етапу робіт, а ітерації – у вигляді набору робіт на заданому часовому інтервалі. робочі елементи

При плануванні командного проекту ключовими сутностями є робочі елементи. Залежно від шаблону командного проекту набір і найменування робочих елементів дещо відрізняються. Так для гнучкої методології Agile робочими елементами є :

- Користувача опис функціональності (UserStory);
- Завдання (Task);
- Помилка (Bug);
- Перешкода (Issue);
- Тестовий випадок.

Робочий елемент користувача "опис функціональності" представляє собою користувальницьке вимога, яку необхідно виконати при реалізації проекту.

Робочий елемент "Завдання" створюється в проекті для призначення та виконання роботи. Завдання надають деталі реалізації для користувача вимог.

Робочий елемент "Помилка" використовується для відстеження та моніторингу проблем в програмному продукті.

Робочий елемент "Перешкода" використовується для фіксації в проекті подій або об'єктів, які створюють проблеми у виконанні проекту і повинні бути усунені в ході поточної або майбутньої ітерації.

Робочий елемент "Тестовий випадок" описує умови перевірки правильності виконання програмним продуктом вимог користувача.

Робочі елементи можуть включати найменування, опис призначення, стан, кому призначено, цінність для бізнесу, приналежність до робочої області.

При плануванні командного проекту користувальницькі вимоги записуються в Елемент невиконана робота по продукту. Дану роботу можна проводити з використанням: Командного оглядача в VisualStudio; Microsoft Project; Microsoft Excel.

Користувальницькі опису функціональності можуть бути пов'язані між собою, а також з дочірніми або батьківськими елементами. В якості дочірніх елементів можуть бути, наприклад, завдання або помилки.

Створення проекту моделювання програмного додатка. Після розробки початкового списку вимог користувача, які є змістом робочого елементу невиконані роботи по продукту, доцільно обговорити зафіксовані в проекті вимоги із зацікавленими особами. При проведенні обговорення користувача вимог до програмного продукту доцільно представити зацікавленим особам не тільки текстову документацію, але графічний матеріал, який більш наочно відображає користувальницькі вимоги. Для цього використовуються UML-діаграми. Власник продукту, виконуючи роль архітектора, для підготовки графічних діаграм може використовувати можливості Visual Studio з архітектурного моделювання.

ПРАКТИЧНА ЧАСТИНА

Завдання

Ознайомитися з теоретичним матеріалом. Для проекту, за варіантом завдання виконати архітектурне моделювання засобами Azure DevOps наступних об’єктів: моделювання функціональності і класів; схему варіантів використання; схему класів; проектування архітектури ПЗ, проектування схеми БД; розробку діаграм прецедентів; ієрархію класів, послідовності розгортання.

Приклад виконання

Розглянемо виконання цієї лабораторної роботи на прикладі створення проекту «ІС «Відділ кадрів підприємства»

В Azure DevOps можливо створювати діаграми UML, використовуючи вбудовані можливості платформи або сторонні розширення. Для створення необхідних схем та діаграм використовуйте вбудований редактор діаграм Azure DevOps.

Для цього виконаємо наступні кроки:

– Відкрийте свій проект в Azure DevOps.
– У меню ліворуч виберіть «Wiki» і створіть нову сторінку в розділі «Документація» (рис. 6.1).
– Натисніть кнопку "Вставити/редагувати діаграму" і виберіть потрібний тип діаграми UML (наприклад, "Діаграма класів" або "Діаграма послідовностей").
– Здійсніть створення діаграми за допомогою інструментів вбудованого редактора.

Рис. 6.1 – Панель на головному екрані з підпунктом “Wiki”

Після цього з’являється віконце, яке пропонує користувачу створити свій проект (рис. 6.2).

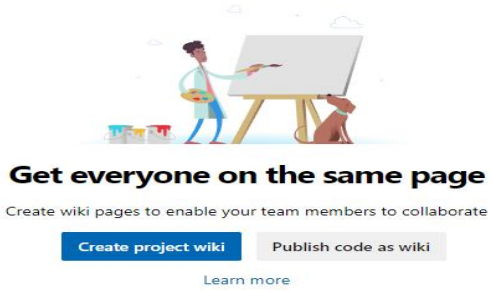

Рис. 6.2 – Створення проекту

Коли користувач обрав «створити проект», то з’являється панель розробки на якій вже будемо працювати та створювати UML-діаграми (рис.6.3).

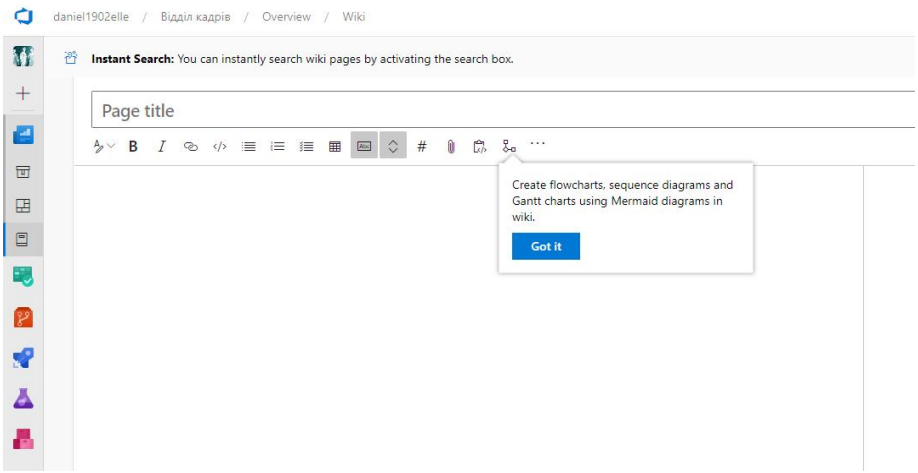

Рис. 6.3 – Панель розробки діаграм

Панель розбивається на 2 екрана. Екран, який розташовано ліворуч – це панель на якій будемо прописувати властивості нашої діаграми, а панель розташована праворуч – буде відображати нашу діаграму (рис. 6.4). Це варіант служить для тих, хто бажає, щоб UML-діаграма була побудована одразу засобами Azure DevOps.

Рис. 6.4 – Приклад інструменту UML в Azure DevOps

Але є і другий варіант створення UML-діаграм. У Visual Studio можна створювати UML-діаграми та імпортувати їх в Azure DevOps. Для цього потрібно виконати наступні кроки:

– Створіть UML-проект в Visual Studio: в VS – створити новий проект.
– Оберіть тип проекта "UML Class Library" чи "UML Activity Library" (рис. 6.5).

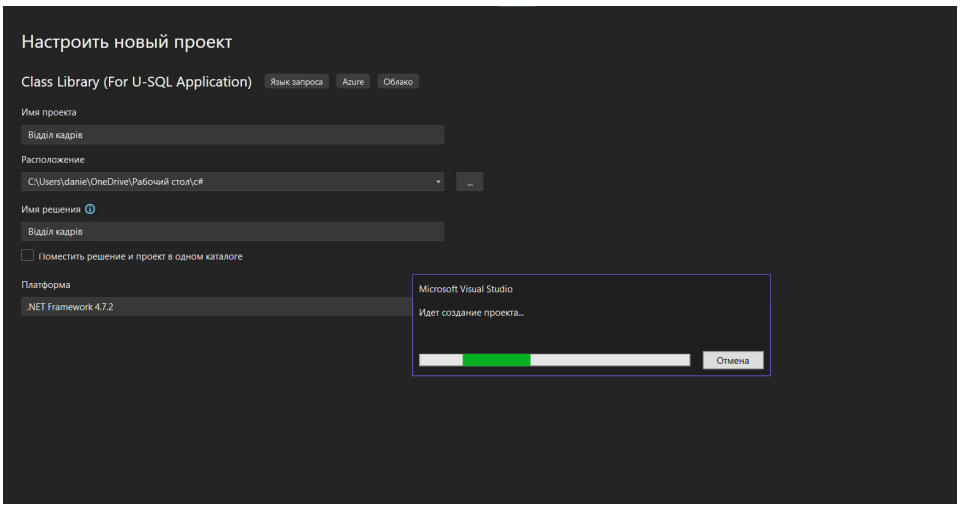

Рис. 6.5 – Створення проекту в VS з використанням UML Class Library

- Створіть UML-діаграму засобами редактора в Visual Studio.
- Експортуйте схему UML із Visual Studio:
  - Відкрийте UML-схему в Visual Studio.
  - В меню "Architecture" виберіть "Export Diagram as Image".
  - Виберіть формат файлу зображення та збережіть його на комп'ютері.
- Імпортуйте діаграму UML в Azure DevOps:
  - Відкрийте свій проект в Azure DevOps.
  - У меню ліворуч виберіть «Wiki» і створіть нову сторінку в розділі «Документація».
  - Натисніть кнопку "Вставити/редагувати діаграму" та виберіть тип діаграми "Зображення".
  - Завантажте файл зображення схеми UML, експортований із Visual Studio.

Тепер ви можете використовувати UML-діаграму в Azure DevOps для документування вашого проекту. Зверніть увагу, що якщо ви редагуєте діаграму в Visual Studio, вам потрібно буде повторно експортувати та імпортувати її в Azure DevOps.

Після цього у Visual Studio можна створювати UML-діаграми за допомогою засобів редактора:

1. Відкрийте проект у Visual Studio:
   - Відкрийте Visual Studio та виберіть свій проект;
   - В меню "Project" виберіть "Add New Item";
   - У діалоговому вікні, що з'явиться, виберіть тип "UML Class Diagram" або "UML Activity Diagram";
   - Введіть назву діаграми та натисніть кнопку Додати.

2. Додайте елементи до діаграми (рис. 6.6):

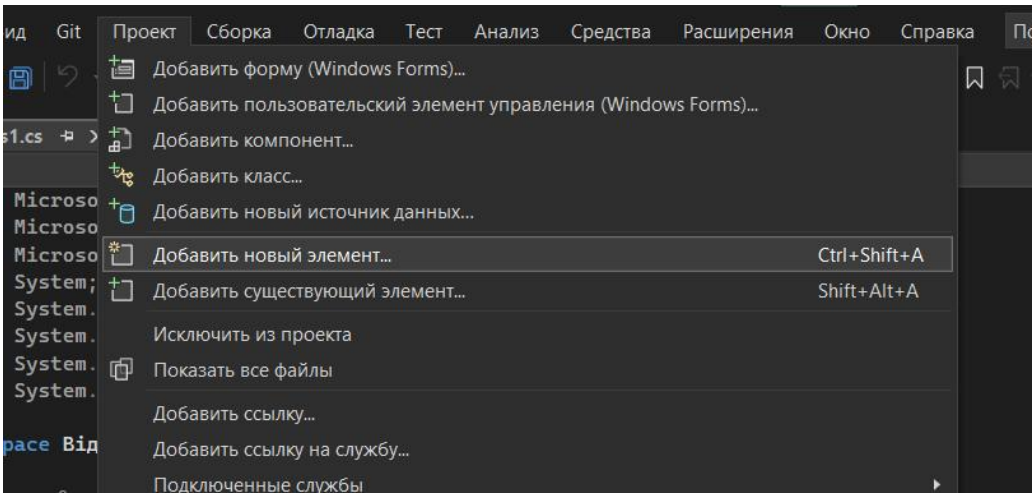

Рис. 6.6 – Додавання нового елементу

3. Дайте назву вашому елементу (рис. 6.6);

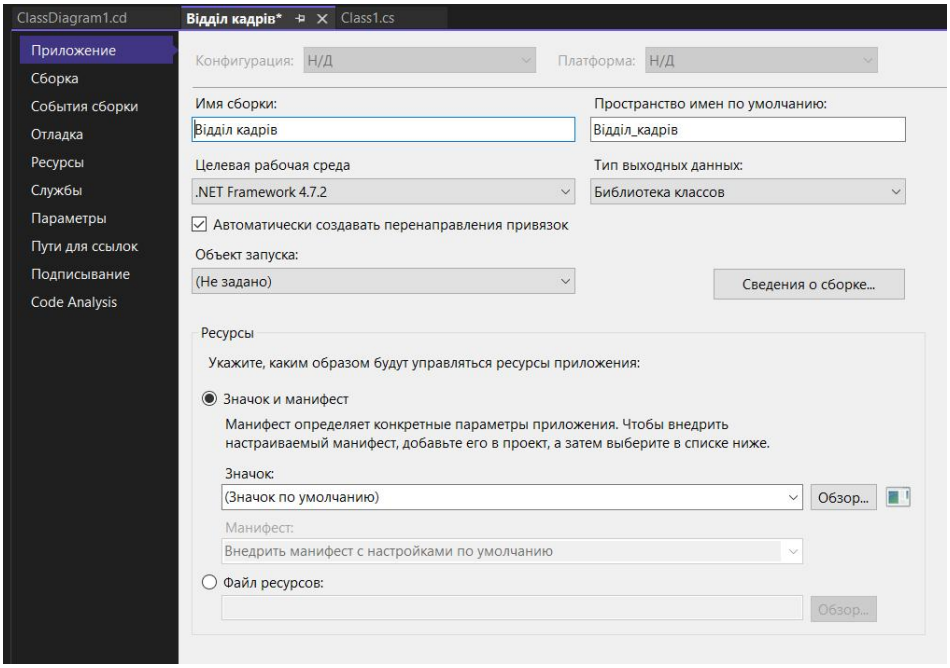

Рис. 6.6 – Додаємо назву

- Виберіть потрібний елемент з панелі інструментів (наприклад, клас, інтерфейс, асоціацію);
- Намалюйте елемент на діаграмі за допомогою миші;
- Додайте властивості та методи до класів, якщо це необхідно.

4. Створення зв'язків між елементами:
   - Виберіть інструмент зв'язку (наприклад, агрегація, композиція, успадкування);
   - Намалюйте зв'язок між двома елементами на діаграмі;
5. Додайте коментарі до діаграми (необов'язково):

– Виберіть інструмент "Коментар";
– Намалюйте коментар на діаграмі;
– Напишіть текст коментаря;

6. Збережіть діаграму.

– В меню «Файл» виберіть «Зберегти»/ «Зберегти як» і введіть ім'я файлу;
– Діаграма буде збережена у вашому проекті у вигляді файлу з розширенням ".cd" (Class Diagram) або ".ad" (Activity Diagram);

Тепер для моделювання проекту можна використовувати UML-схему в Visual Studio. Зауважте, що можна також використовувати інші типи діаграм UML, наприклад діаграми послідовності, стану або компоненти.

Після додавання нового елемента, ми переходимо до створення класів мовою програмування C# (рис. 6.7).

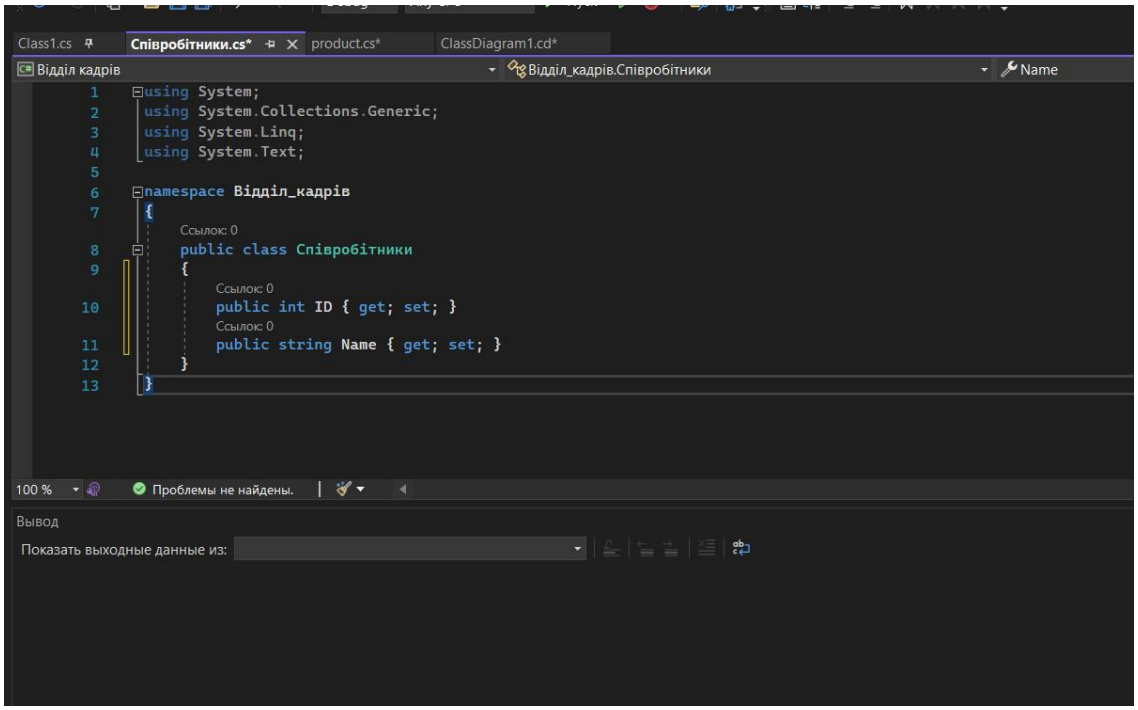

Рис. 6.7 –Створення класів у Visual Studio

Коли ми створили певний клас, то можливо здійснити перегляд початкового вигляду класу з властивостями (рис. 6.8).

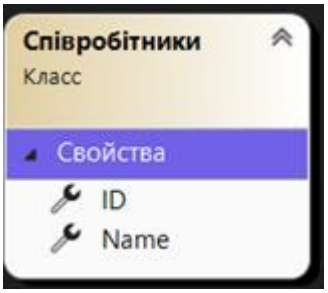

Рис. 6.8 – Початковий вигляд класу

Потім для цього класу ми додаємо ще більше властивостей, які призначені саме для цього класу (рис. 6.9)

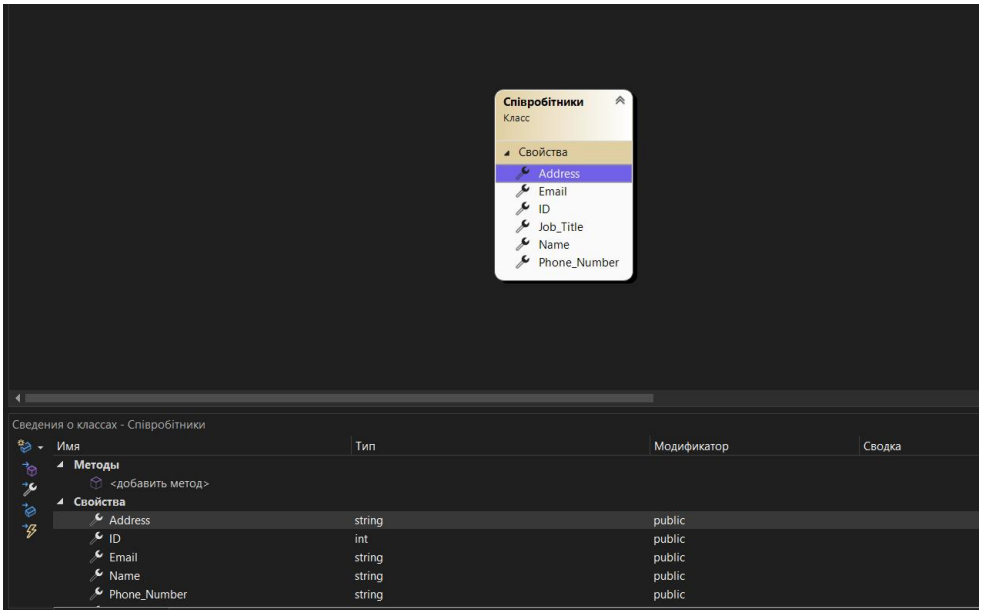

Рис. 6.9 –Створення класу «Співробітники» проекту«ІС «Відділ кадрів»

Так повинен виглядати готовий вигляд класу«Співробітники» проекту. Після цього ми вже можемо створювати інші класи і додавати кожному властивості, які притаманні саме цьому класу (рис. 6.10).

Рис. 6.10 – Створені класи проекту «ІС «Відділ кадрів»

Після того, як виконано розробка та створення всіх класів проекту, можливо виконувати асоціації між ними (рис. 6.11).

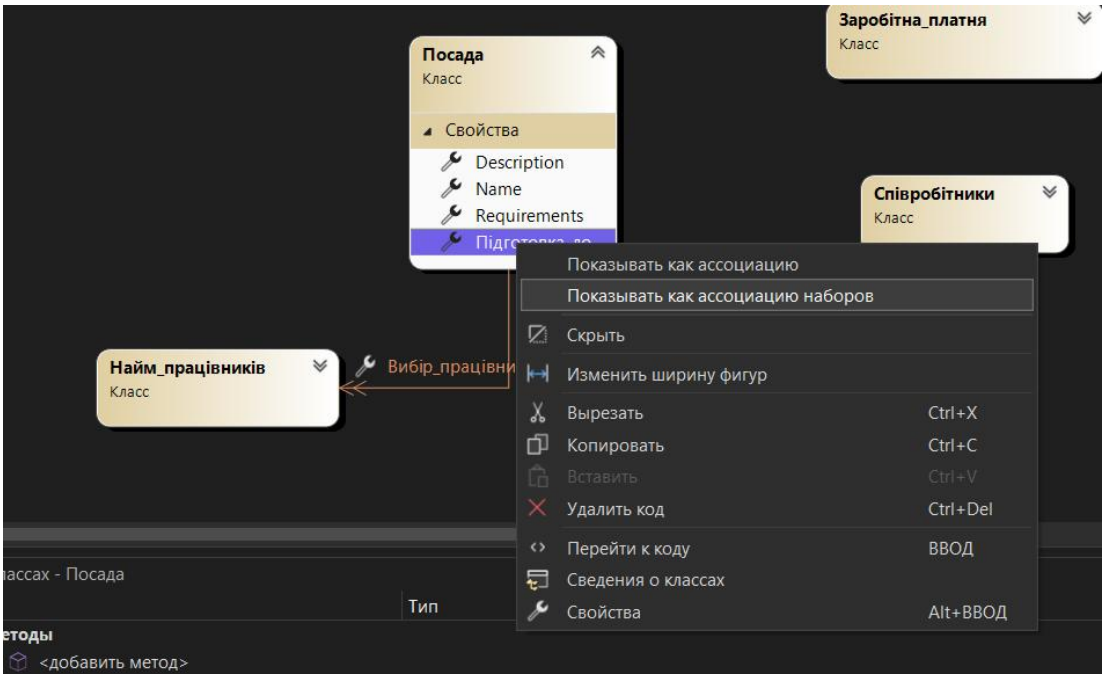

Рис. 6.12 – Створення асоціацій між класами проекту

Коли ми створили асоціацію між всіма класами, то наша UML-діаграма має наступний вигляд (рис. 6.13).

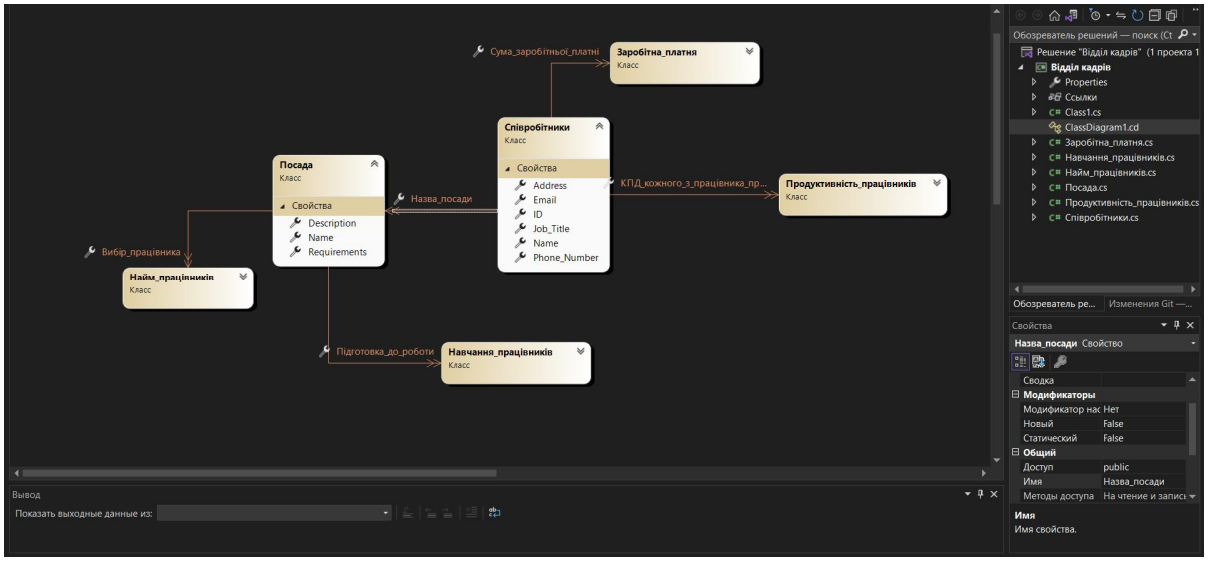

Рис. 6.13 – UML-діаграма класів проекту

UML-діаграма класів проекту має наступний перелік класів:

- ***Employee*** – Клас, який відображає дані працівника. Містить інформацію про його особисті дані, займану посаду, заробітну плату та інші дані.
- ***Department*** – Клас, який відображає дані про відділи підприємства. Містить інформацію про назву відділу, його голову, структуру та інші характеристики.

- ***JobPosition*** – Клас, який відображає дані про посаду робітника підприємства. Містить інформацію про назву посади, її опис, вимоги та інші характеристики.
- ***Payroll*** – Клас, який відображає дані про заробітну плату робітників. Містить інформацію про заробітну плату працівника, дату виплати та інші характеристики.
- ***Recruitment*** – Клас, який відображає процес найму працівників. Містить інформацію про відкриті вакансії, кандидатів на посаду, результати співбесід та інші характеристики.
- ***Training*** – Клас, який відображає процес навчання працівників. Містить інформацію про доступні курси, навчальні матеріали, прогрес навчання та інші характеристики.
- ***Performance*** – Клас, який відображає процес оцінки продуктивності працівників. Містить інформацію про критерії оцінки, результати оцінювання та інші характеристики.

Звіт

Звіт з лабораторної роботи повинен складатися з Документу «Протокол лабораторної роботи», що містить:

- Назву роботи, мету;
- Хід виконання роботи (графічні результати виконання роботи);
- Відповіді на контрольні питання;
- Висновки.

Захист звіту з лабораторної роботи полягає в пред'явленні викладачеві отриманих результатів, відповідях на питання викладача.

Контрольні питання

1. Які інструменти має Team Foundation Server для управління командними проектами?
2. Які папки генеруються при створенні командного проекту?
3. Для чого використовуються у проекті області та ітерації?
4. Поясніть процес створення коду розробником командного проекту в середовищі VisualStudio.
5. За допомогою яких клієнтських інструментів можна проводити планування командного проекту?

Лабораторна робота №7

Управління робочим процесом робочих елементів. Реалізація методів Scrum з застосуванням Azure Boards

Мета роботи: Отримати практичні навички проведення оцінки складності елементів робіт та встановити пріоритети робочим елементам, включених в поточний спринт. Розподілити завдання спринту між членами команди, виконати планування спринту.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Управління життєвим циклом програмного продукту допомагає розробникам цілеспрямовано домагатися створення якісного ПЗ, уникати втрат часу на переробку, повторне проектування та перепрограмування ПЗ.

Якість програмного продукту визначається за кількома критеріями. Якісний програмний продукт повинен відповідати функціональним і нефункціональним вимогам, відповідно до яких він створювався, мати цінність для бізнесу, відповідати очікуванням користувачів. В життєвому циклі управління додатками якість повинна відслідковуватися на всіх етапах життєвого циклу ПЗ. Вона починає формуватися з визначення необхідних вимог. При завданні вимог необхідно вказувати бажану функціональність і способи перевірки її досягнення.

Важливим аспектом створення якісного ПЗ є забезпечення функціональних вимог, таких як зручність в експлуатації, надійність, продуктивність, захищеність, зручність супроводу. Надійність ПЗ визначає здатність без збоїв виконувати задані функції в заданих умовах і протягом заданого відрізку часу. Продуктивність характеризується часом виконання заданих транзакцій або тривалих операцій. Захищеність визначає ступінь безпеки системи від пошкоджень, втрати, несанкціонованого доступу та злочинної діяльності. Зручність супроводу визначає легкість, з якою обслуговується продукт в плані простоти виправлення дефектів, внесення коректив для відповідності новим вимогам, управління зміненої середовищем.

Для планування проекту з розробки програмного забезпечення та відстеження дефектів ПЗ за допомогою методології Scrum команди

використовують типи робочих елементів «Елемент невиконаної роботи» (PBI) та «Помилка». Щоб отримати уявлення про портфелі функцій, сценаріїв або інтерфейсу користувача, власники продуктів і керівники програм можуть зіставляти PBI і помилки з функціями. Коли команди працюють у спринтах, вона визначає завдання, які автоматично пов'язуються з PBI та помилками.

Засоби Sprints включають відфільтровану невиконану роботу на основі шляху ітерації та аналогічно відфільтрованої області завдань. Ці засоби корисні для реалізації методик Scrum. За допомогою Scrum можна планувати спринти, оновлювати область завдань та відстежувати спринт.

Методи Scrum використовують шляхи ітерації, також звані спринтами, для планування роботи, яку команда виконуватиме протягом певного періоду часу та певної частоти. Щоб розпочати роботу, для вашої команди зумовлено кілька спринтів.

Невиконана робота зі спринту та дошкам завдань надають відфільтроване уявлення робочих елементів, призначених командою по певному шляху ітерації або спринту. Спринти визначаються для проекту, потім вибираються командами. З невиконаної роботи можна зіставити роботу за допомогою ітерації за допомогою перетягування, а потім переглянути в окремому спринті невиконаної роботи.

Кожен спринт, вибраний для вашої команди, надає доступ до невиконаної роботи, дошки завдань та інших інструментів Agile для планування та відстеження роботи.

На початку кожного спринту потрібно спланувати роботу, яку може виконати ваша команда. До трьох засобів Agile, що підтримують цю роботу, відносяться невиконана робота зі спринту, планування ємності та гістограма ємності. Невиконана робота зі спринту містить відфільтровану підмножину елементів невиконаної роботи, шлях ітерації яких відповідає поточному спринту.

Тестувальники можуть створювати та запускати тестові випадки, а також створювати помилки для відстеження дефектів коду за допомогою веб-порталу або Microsoft Test Manager. Перешкоди використовуються для відстеження проблем, що блокують. Задаючи ємність команди, команда точно знає загальну кількість робочих годин чи днів, які команда має на кожному спринту. За допомогою цього засобу можна задати ємність та

вихідні дні для окремих учасників команди. І, зручно, можливо встановити свята чи спільні вихідні дні, прийняті усією командою.

Завдання ємності для кожного учасника команди, що працює під час спринту, призводить до відображення індикатора ємності для цієї людини.

Власник продукту може визначити пріоритети невиконаної роботи з продуктів на основі бізнес-цінностей, зусиль та відносної залежності від інших елементів невиконаної роботи. Список невиконаних робіт по продукту змінюється зі зміною бізнес-вимог. Як правило, команди докладно описують лише елементи з найвищим пріоритетом або елементи, призначені для поточного та наступного спринту.

Тестування програмного забезпечення

Тестування програмного продукту дозволяє протягом усього життєвого циклу ПЗ гарантувати, що програмні проекти відповідають заданим параметрам якості. Головна мета тестування - визначити відхилення в реалізації функціональних вимог, виявити помилки у виконанні програм і виправити їх якомога раніше в процесі виконання проекту.

Протягом усього життєвого циклу розробки ПО застосовуються різні типи тестування для гарантії того, що проміжні версії відповідають заданим показникам якості. При цьому застосовуються автоматичні і ручні тести.

Для розробника програмного забезпечення в Visual Studio надана можливість створювати модульні і навантажувальні тести, а також тести для користувача інтерфейсу.

У Visual Studio є наступні шаблони тестових проектів:

- проект модульного тесту, який дозволяє створювати модульні тести в процесі розробки;
- проект з веб-тестами продуктивності і навантажувальними тестами;
- проект з закодованими тестами для користувача інтерфейсу.

Інструментарієм тестувальника в Visual Studio є Microsoft Test Manager (MTM). MTM призначений для управління життєвим циклом тестування програмного забезпечення, включаючи планування, тестування та моніторинг. MTM інтегрований з Team Foundation Server. За допомогою Microsoft Test Manager тестувальники готують плани тестування, управляють тестуванням. При створенні плану тестування в нього додаються набори

тестів, тестові випадки і конфігурації, необхідні для тестування. Конфігурації використовуються для встановлення середовища, в якій будуть виконуватися набори тестів. Microsoft Test Manager дозволяє виконувати ручні та автоматичні тести, а також дослідницькі тести. Результати тестування зберігаються в базі даних, що дозволяє готувати різні аналітичні звіти. Помилки, виявлені в процесі тестування, фіксуються, документуються і передаються розробникам для їх усунення. При внесенні змін в код програмної системи виникає необхідність у регресійному тестуванні, причому MTM автоматично формує план регресійного тестування, виявляючи які тести повинні бути повторно виконані. Модульне тестування призначено для перевірки правильності функціонування методів класів ПЗ. Модульні тести пишуться і виконуються розробниками в процесі написання коду. Модульне тестування застосовується як для перевірки якості коду програми, так і для перевірки об'єктів баз даних.

ПРАКТИЧНА ЧАСТИНА

Завдання

Ознайомитися з теоретичним матеріалом. Для проекту, згідно з варіантом завдання провести призначення 4–6 спринтів, провести оцінку складності елементів робіт. Встановити час виконання спринтів, пріоритети та призначити виконавців. Для робочих елементів, включених в поточний спринт, визначити завдання. Розподілити завдання спринту між членами команди.

Приклад виконання

Розглянемо приклад виконання завдання цієї лабораторної роботи при створенні ПП «Інформаційна система «Відділ кадрів підприємства», а саме створимо спринти, визначимо виконавців, оцінимо складність, визначимо пріоритети, робочі елементи, завдання. Розподілимо завдання проекту між членами команди розробників.

Для виконання цієї лабораторної роботи створюємо спринти, що дозволить налаштувати та описати роботи, які повинна виконати команда. Для роботи з спринтами потрібно обрати вкладку зліва “Taskboard”, а потім вже вкладку “Sprints”(рис. 7.1).

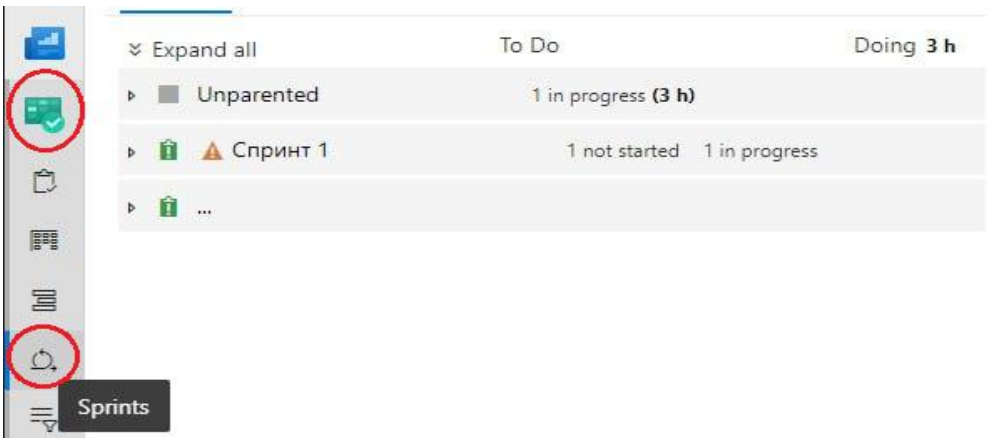

Рис. 7.1 – Переходимо в меня “Sprints”

Коли перешли до вкладки “Sprints, то у вікні відображено всі наявні у обраному проекті спринти. Обравши один з спринтів з переліку, розглянемо етапи налаштування саме обраного спринту. В нас з’являється маленьке віконце спринту, де можливо його налаштувати чи щось додати (рис. 7.2).

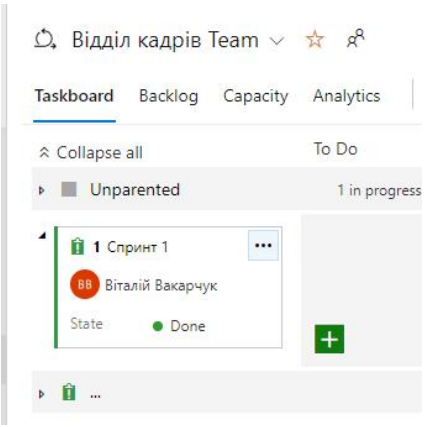

Рис. 7.2 – Вікно, що відображає обраний спринт

Для призначення відповідального за цей спринт потрібно обрати учасника команди з переліку, який відображено у вікні (рис. 7.3).

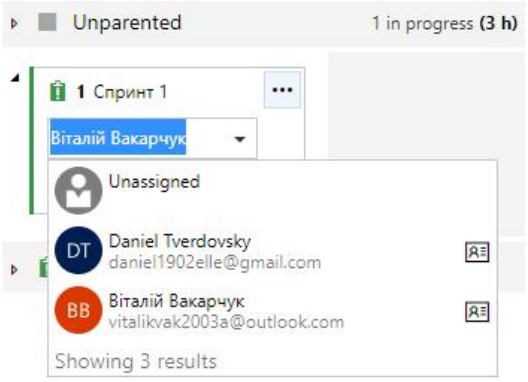

Рис. 7.3 – Обираємо відповідального за спрінт

Після цього ми натискаємо на три крапки (…) і відкривається вікно налаштувань спрінта, де ми обираємо розділ «опис спринта» (рис. 7.4).

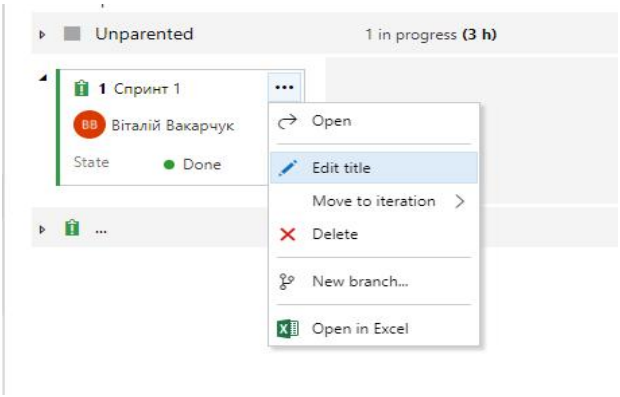

Рис. 7.4 – Налаштування спринта

Користувачу відкривається діалогове вікно, в якому можливо надати стислий опис спринту (рис. 7.5).

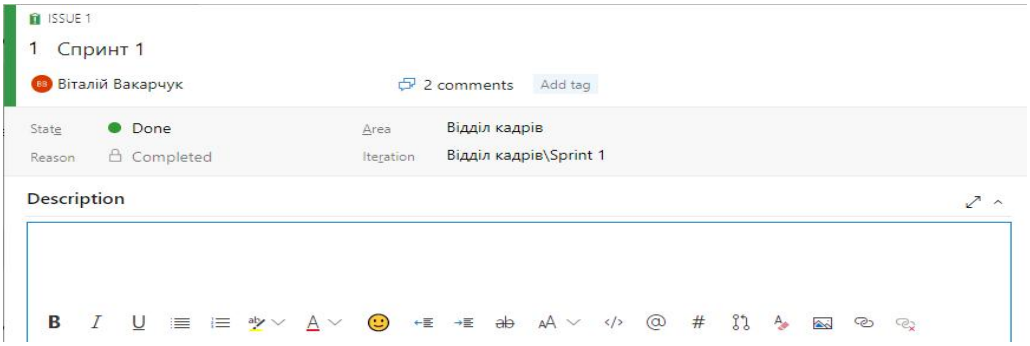

Рис.7.5. – Вікно для опису спринта

При здійсненні стислого опису спринта обов’язково зазначають яке саме завдання повинно бути виконано. Опис цього завдання повинен бути зрозумілий на сам перед для виконавця (рис. 7.6).

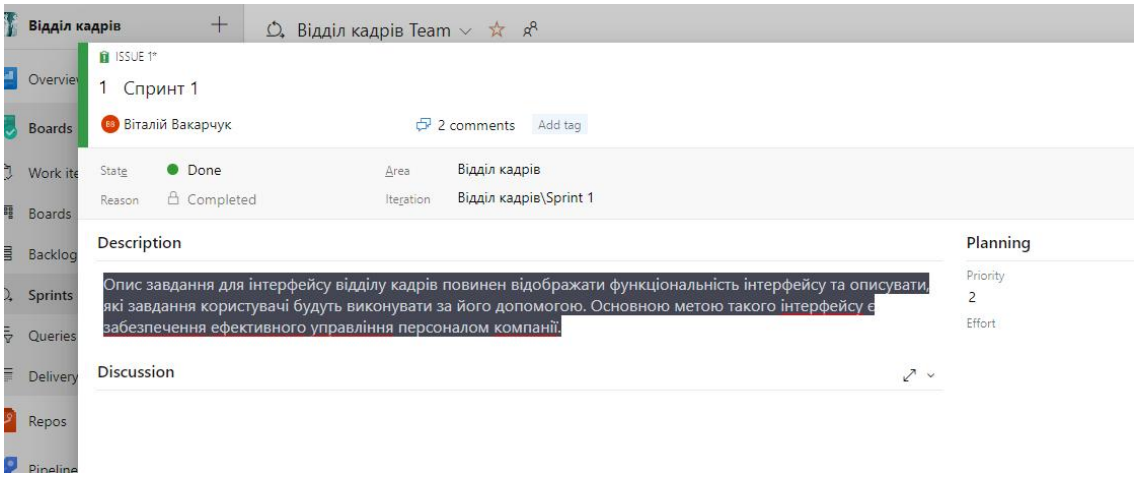

Рис. 7.6 – Визначення завдання спринта

Для спринту можливо надати пріоритет виконання у наступному вікні (рис. 7.7).

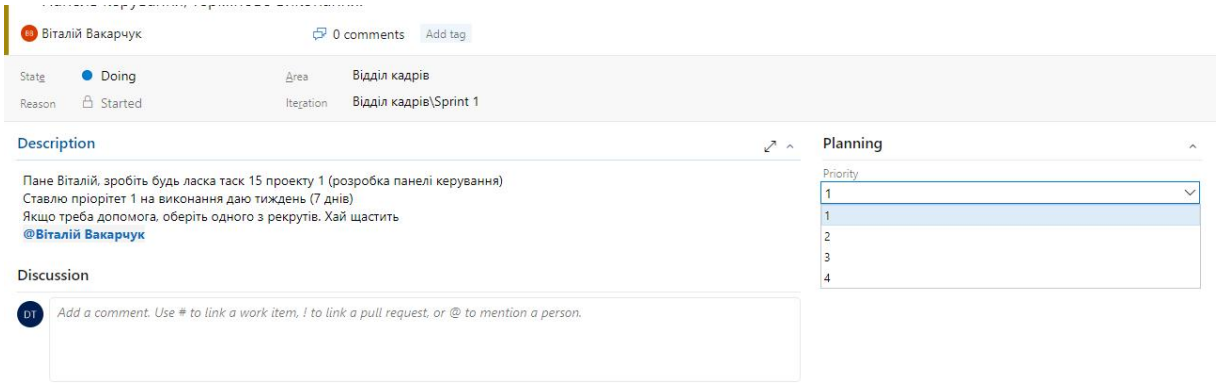

Рис. 7.7 – Додавання пріоритету спринту

Під час створення спринту користувачу обов’язково потрібно задати час виконання роботи, щоб потім мати можливість відстежувати терміни виконання (рис. 7.8).

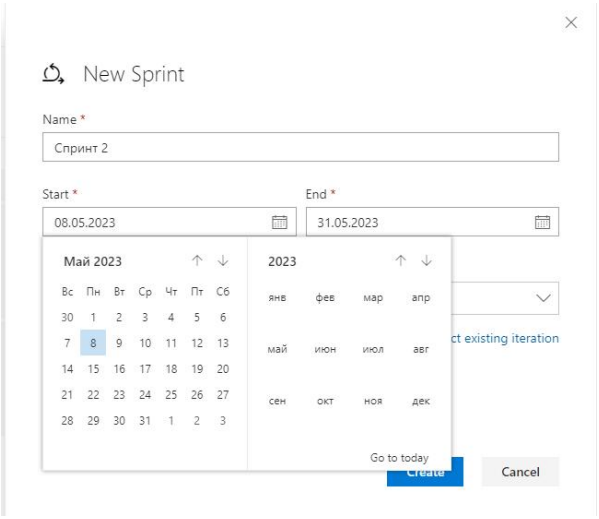

Рис. 7.8 – Визначення часу виконання роботи

Але якщо під час роботи пріорітети зміняться чи плани на проект, то можливо додати в описі зміну часу (рис.7.9).

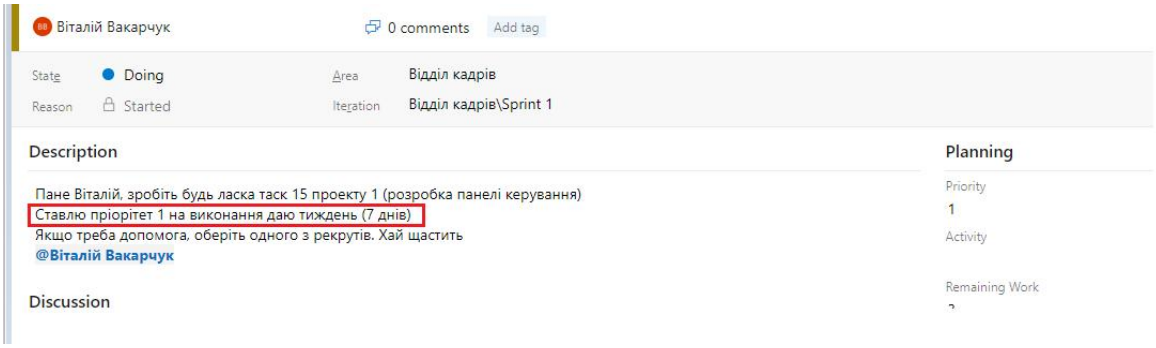

Рис. 7.9 – Зміна часу в спринті проекту

Після виконання всіх потрібних налаштувань спринта, повертаючись до дошки всіх спринтів бачимо відображення наступного вікна (рис.7.10). В цьому вікні відображені спринти проекту та їх виконавці.

Рис. 7.10 – Вікно відображення всіх спринтів проекту

Якщо в проекті передбачено велика кількість спринтів, то можливо використати пошукову функцію, яка дозволить користувачу швидко находити і відстежувати виконання потрібного спринту (рис. 7.11).

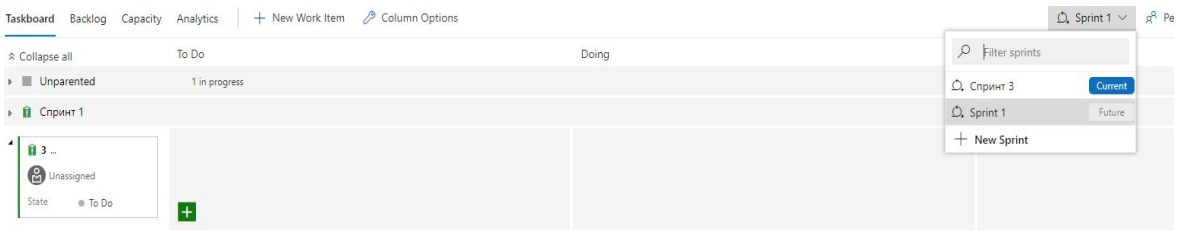

Рис. 7.11 – Застосування функції пошуку спринтів

Якщо користувач виконав спринт, то статус спринту буде змінено на “Виконаний”, а також візуально у вікні спринт буде мати зелену позначку (рис. 7.12 – 7.13).

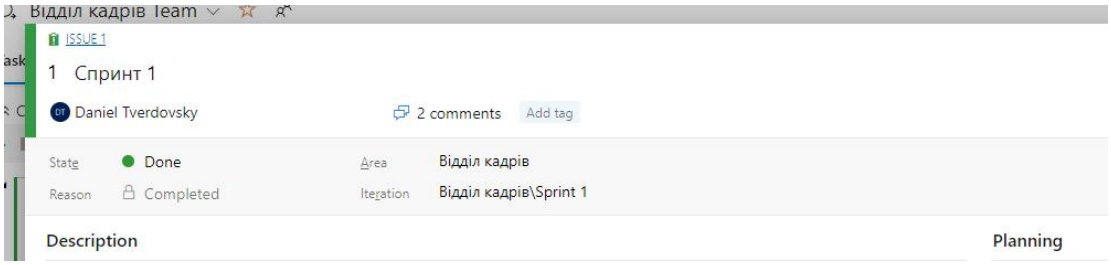

Рис. 7.12 – Перегляд виконаного спринту

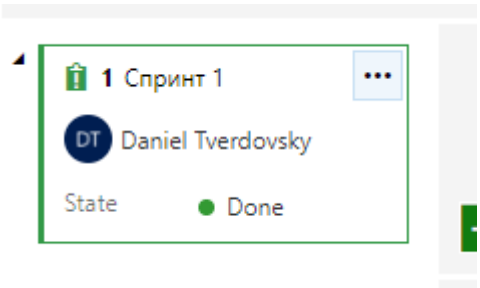

Рис. 7.13 – Відображення виконаного спринту на панелі спрінтів

Звіт

Звіт з лабораторної роботи повинен складатися з документу «Протокол лабораторної роботи», що містить:

- Назву роботи, мету;
- Хід виконання роботи (графічні результати виконання роботи);
- Відповіді на контрольні питання;
- Висновки.

Захист звіту з лабораторної роботи полягає в пред'явленні викладачеві отриманих результатів, відповідях на питання викладача.

Контрольні питання

1. При плануванні проекту які типи робочих елементів використовуються?
2. За якими критеріями визначається якість програмного продукту?
3. Хто з команди має можливість призначати пріоритети невиконаним роботам?
4. Які методи Scrum для вашої команди визначені в Azure Boards?
5. За якими критеріями визначається надійність та продуктивність ПЗ?
6. Що визначається в спринті?

Лабораторна робота №8

Розробка та макетування інтерфейсів користувача програмного продукту

Мета роботи: отримати практичні навички розробки представлення та макетування інтерфейсу користувачів ПЗ, створити розкадрування для інтерфейсу додатку, що розробляється.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Зручним інструментом макетування користувача інтерфейсу реалізації вимог користувача є PowerPoint, який інтегрований з Visual Studio. Для Елементів «задел роботи продукту» є можливість створити розкадрування подання екранних форм або веб-сторінок додатку програмного продукту. Розкадрування призначена для попереднього обговорення основних рішень для користувача інтерфейсу.

Існує багато варіантів розробки документації для створеного програмного продукту та підготовки інсталяційного пакету, але розглянемо два найбільш популярних серед розробників варіанта. По-перше, можливо використання Visual Studio Code через Azure DevOps, по-друге, можливо використання інструментів Figma.

Щоб створити інтерфейс і розмістити його в Azure DevOps, засобами Visual Studio Code потрібно виконати наступні дії:

- створіть інтерфейс за допомогою відомих вам технологій. Наприклад, ви можете створити інтерфейс за допомогою HTML, CSS та JavaScript;
- визначте, як ви будете зберігати код інтерфейсу та керувати ним. Якщо ви ще не зареєстровані в Azure DevOps, зареєструйтеся та створіть проект, де ви будете зберігати свій код. У проекті створіть репозиторій, де буде зберігатися код інтерфейсу;
- завантажте код інтерфейсу в репозиторій. Для цього ви можете використовувати Git cli або інтерфейс Git, інтегрований в код Visual Studio;
- налаштуйте процес створення та розгортання вашого інтерфейсу. Для цього можна використовувати різні інструменти в Azure DevOps, наприклад Azure Pipelines. Налаштуйте збірку та розгортання таким

чином, щоб ваш інтерфейс автоматично будувався та розгортався щоразу, коли код змінюється.

Після успішного створення та розгортання ваш інтерфейс буде доступний за адресою, яку ви вказали в процесі налаштування. Ви можете використовувати цю адресу для тестування свого інтерфейсу та доступу до нього.

Також можливо здійснення розробки інтерфейсу ПЗ засобами та інструментами сервісу Figma. Figma – це хмарний багатоплатформовий сервіс для дизайнерів інтерфейсів і web-розробників, з яким можна працювати безпосередньо в браузері. З точки зору функціоналу Figma – це зручний графічний редактор, в якому можна створювати: прототипи web-сайтів і додатків; окремі елементи інтерфейсу: іконки, кнопки, форми і багато іншого; векторні зображення та ілюстрації, інше.

Крім безкоштовного тарифу і можливості використовувати Figma он-лайн, ця платформа для дизайнерів має цілу низку додаткових переваг:

- розрахований на багато користувачів режим редагування;
- зручне зберігання файлів у власному хмарному сховищі;
- коментування безпосередньо в макеті, що дозволяє уникнути пересилань файлів і довгих листувань в месенджерах;
- прототипування;
- багатозадачність;
- режим презентації за посиланням на проект, зручна система роботи з символами, велика бібліотека компонентів.

Для створення інтерфейсу ПЗ відкриємо у вікні додаток Figma, який і буде основним нашим інструментом. Спочатку треба зареєструватися в сервісі Figma, чи просто зайти на вашу сторінку, після чого вже можемо створити проект (рис. 8.1).

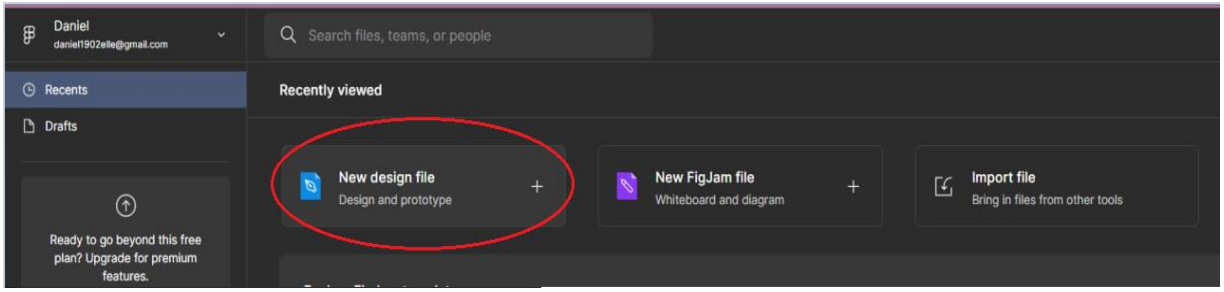

Рис. 8.1 – Головна панель сервісу Figma

Після того як ми обрали створити проект, то нам одразу відкривається вікно з робочим простором на якому ми вже зможемо створювати дизайн інтерфейсу (рис. 8.2).

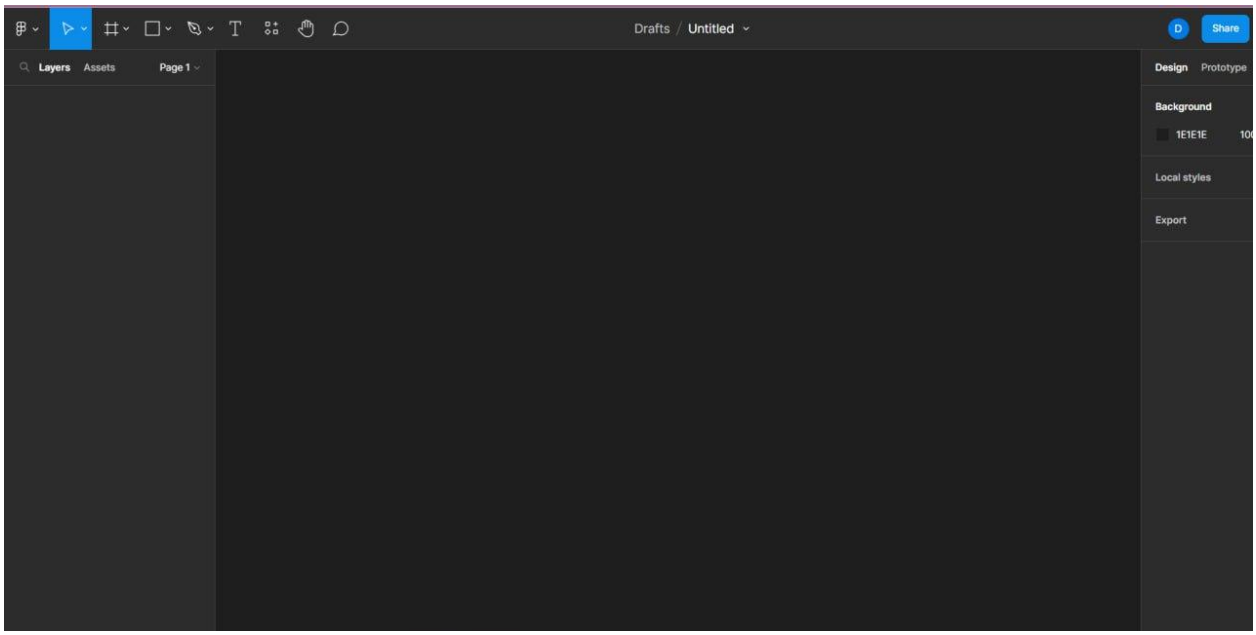

Рис. 8.2 – Робочий простір Figma

ПРАКТИЧНА ЧАСТИНА

Завдання

Ознайомитися з теоретичним матеріалом. Для проекту, згідно з варіантом завдання, провести інструментом макетування сервісу Figma, розробку інтерфейсу користувача, розкадрування подання екранних форм ПП або веб-сторінок додатку. Розкадрування призначене для попереднього обговорення основних рішень для користувачів інтерфейсу.

Приклад виконання

Проведемо розробку та макетування інтерфейсу користувача інструментами сервісу Figma згідно з розглянутим у якості прикладу ПЗ «Інформаційна система «Відділ кадрів підприємства». Результати інтерфейсу проекту «ІС «Відділ кадрів підприємства» імпортували в Azure (рис. 8.3).

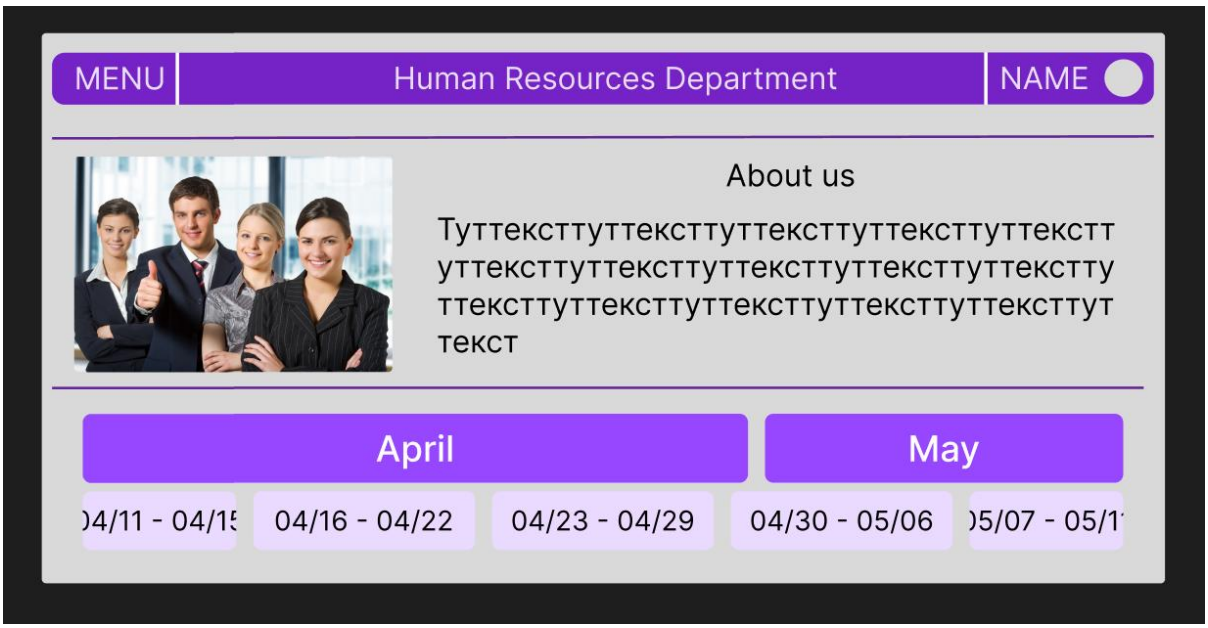

Рис. 8.3 – Початковий вигляд інтерфейсу ІС «Відділ кадрів підприємства»

Згідно з даними, які повинно вносити та зберігати в інформаційній системі та визначеними у технічному завданні функціями детально опрацьовуємо всі необхідні елементи інтерфейсу, а саме сторінок або форм. . Окрім інтерфейсу користувача було розроблено макет інтерфейсу для адміністратора (панель адміністратора) ІС «Відділ кадрів підприємства» (рис 8.4).

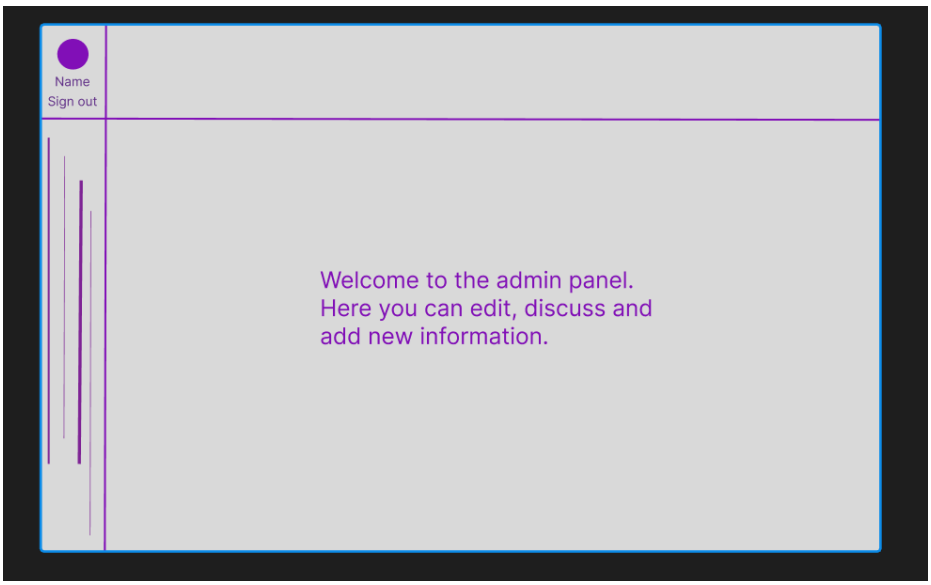

Рис. 8.4 – Макет інтерфейсу для адміністратора (панель адміністратора) ІС «Відділ кадрів підприємства»

Окрім панелі адміністратора було виконано макети інтерфейсу для вкладок (панелей) “Позиція”, “Зарплатня”, "Працівники” (рис. 8.5).

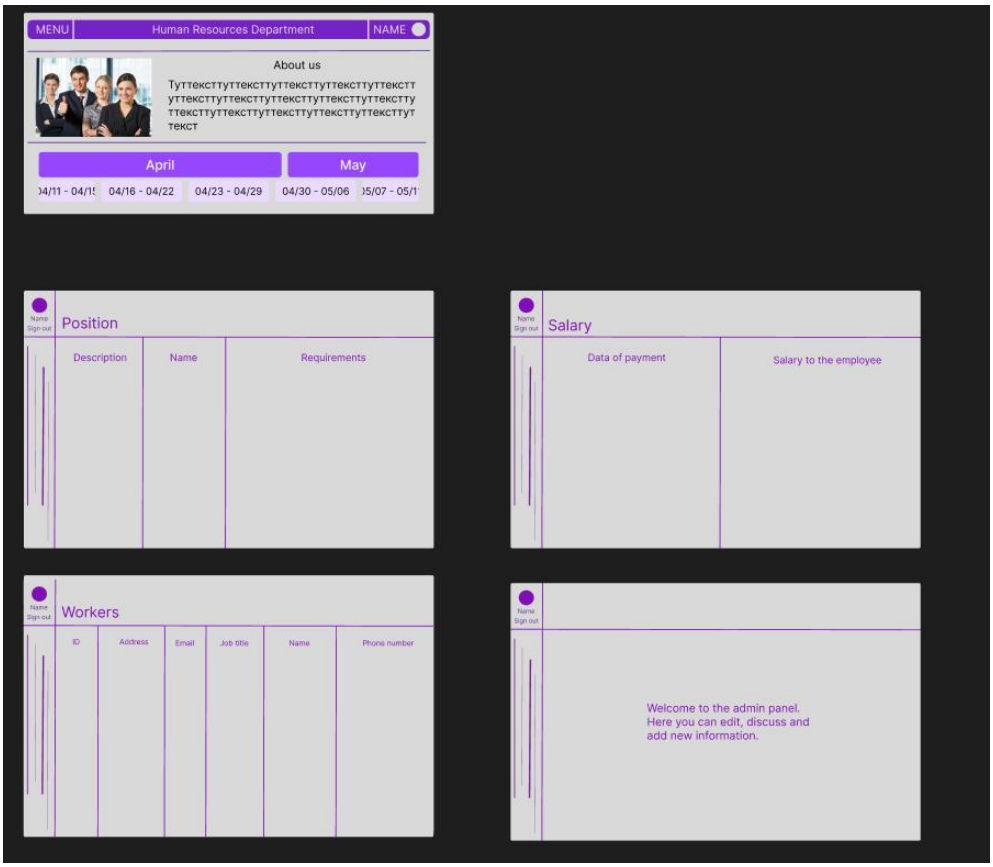

Рис. 8.5 – Розроблені макети інтерфейсів ІС «Відділ кадрів підприємства»

Для того, щоб імпортувати розроблені макети до проекту в Azure потрібно натиснути кнопку “Поділитись”(рис. 8.6).

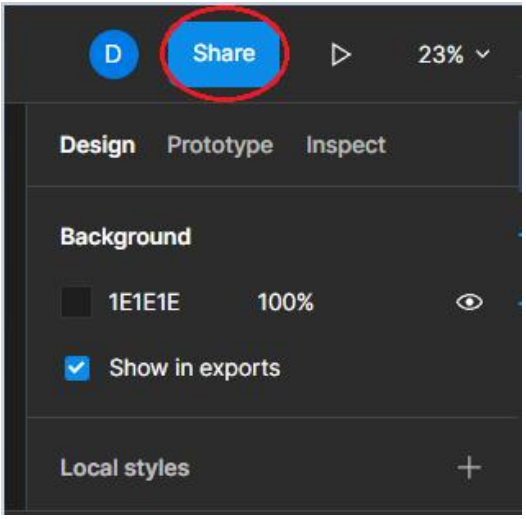

Рис. 8.6 – Кнопка “Поділитись”

Існує багато способів поділитись створеними макетами, але оберемо найзручніший спосіб – копіювання посилання (рис. 8.7).

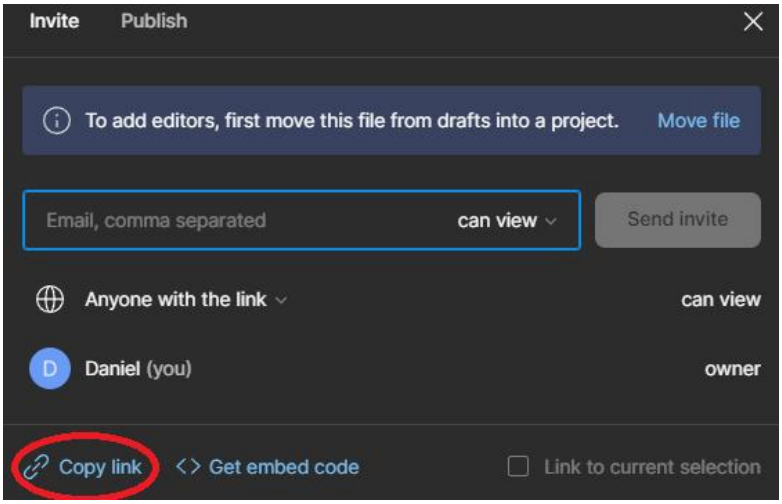

Рис. 8.7 – Обираємо спосіб імпортування макетів – копіювання посилання

Коли ми скопіювали посилання, повертаємося до Azure DevOps, щоб додати посилання. Для цього нам потрібно обрати вкладку “Огляд”, де обираємо “Панелі приладів” (рис. 8.8).

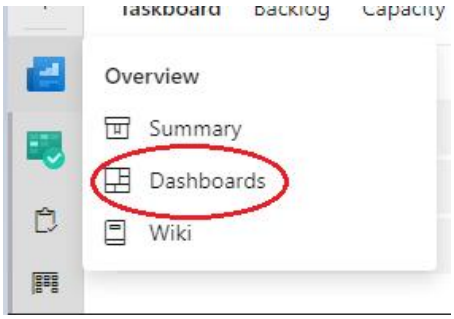

Рис. 8.8 – Обираємо “Панелі приладів”

На наступному кроці відкривається вікно з запрошенням додати необхідний відмет (рис. 8.9).

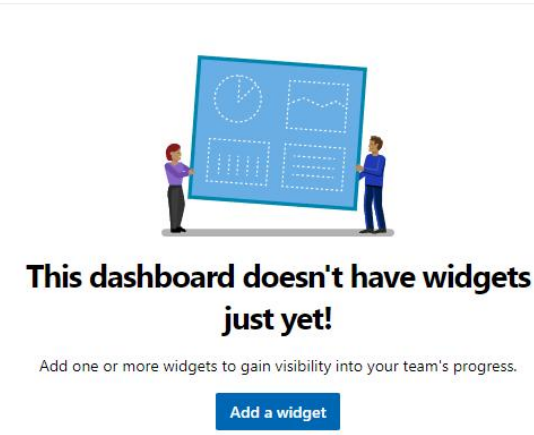

Рис. 8.9 – Діалогове вікно панелі приладів

Після того, як обираємо “Додати віджет”, відкривається вікно, де ми обираємо пункт “Вбудований веб-додаток” (рис. 8.10).

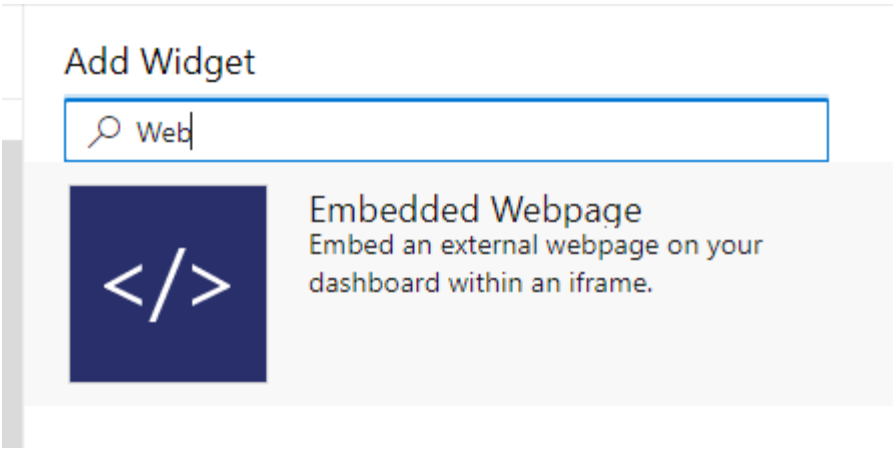

Рис. 8.10 – “Вбудований веб-додаток”

На наступному етапі потрапляємо до діалогового вікна налаштування віджету (рис. 8.11).

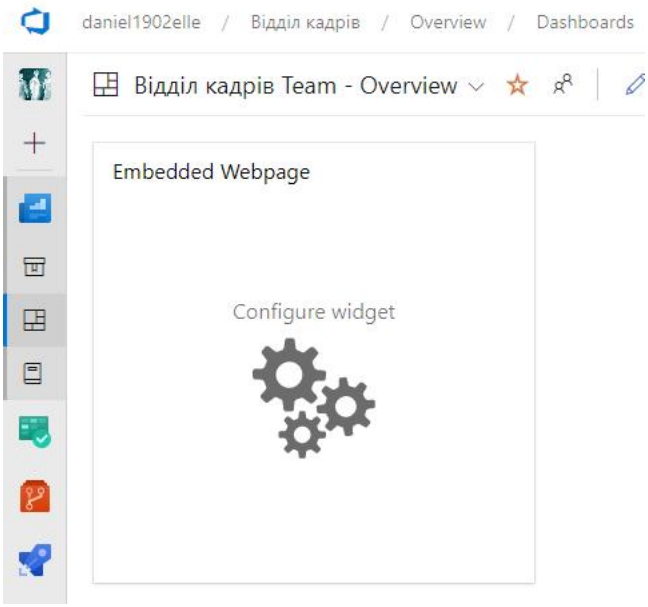

Рис. 8.11 – Вікно налаштування віджетів

Обравши налаштування віджетів, наступним запропоновано користувачу здійснити налаштування, а саме вставляємо наше посилання, встановлюємо висоту та ширину панелі для інтерфейсу (рис. 8.12).

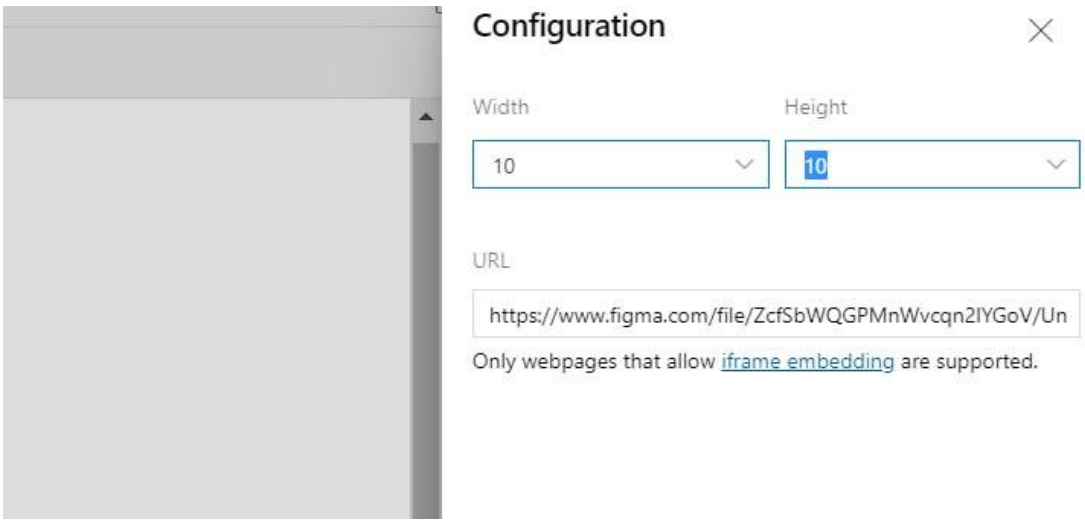

Рис. 8.12 – Налаштування віджета

Після того, як ми вставили посилання, налаштування розмір панелі в робочої області Azure з'являється розроблений макет інтерфейсу ІС «Відділ кадрів підприємства» (рис. 8.13).

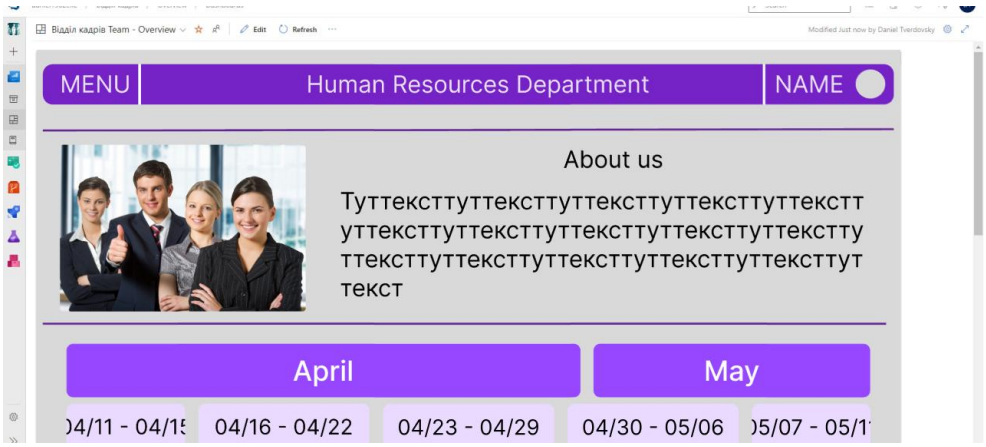

Рис. 8.13 – Макет інтерфейсу ІС «Відділ кадрів підприємства» в панелі Azure

Звіт

Звіт з лабораторної роботи повинен складатися з документу «Протокол лабораторної роботи», що містить:

- Назву роботи, мету;
- Хід виконання роботи (графічні результати виконання роботи);
- Відповіді на контрольні питання;
- Висновки.

Захист звіту з лабораторної роботи полягає в пред'явленні викладачеві отриманих результатів, відповідях на питання викладача.

Контрольні питання

1. Які інструменти надає сервіс Figma для дизайнерів інтерфейсів?
2. Хто обговорює деталі макетів інтерфейсів?
3. Які робочі елементи визначені в Team Foundation Server?
4. Які шаблони проектів маються на TFS ?
5. Які можливості надає система контролю версій TFS?
6. Які можливості має розробник для взаємодії з ключовими службами Team Foundation Server?

ЛІТЕРАТУРА

Основна

1. Гнатовська Г.А. Технологія створення програмних продуктів. Конспект лекцій, ОДЕКУ (електронний варіант) 2016. – 98 с.
2. Дегтярьова Л.М., Гроза П.М., Сомов С.В. Навчальний посібник з дисципліни «Технології розробки програмного забезпечення» – Полтава: ПолтНТУ, 2017. – 218с.

Додаткова

3. Карпенко М. Ю., Манакова Н. О., Гавриленко І. О. Технології створення програмних продуктів та інформаційних систем: навч. посібник. Харків. нац. ун-т міськ. госп-ва ім. О. М. Бекетова. – Харків : ХНУМГ ім. О. М. Бекетова, 2017. – 93 с.
4. О.В. Алексенко. Технології програмування та створення програмних продуктів. Конспект лекцій. – Суми : СумДУ, 2013. – 133 с.