Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
кафедра комп'ютерних систем та мереж
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

магістр
(назва освітнього ступеня)

на тему: Методи і засоби проектування платформ "Low code/No code" в комп'ютерних системах

Виконав(ла): студент(ка) VI курсу, групи СІм-61
спеціальності
(шифр і назва спеціальності)

(підпис) Яворська Х.В (прізвище та ініціали)

Керівник (підпис) Яцишин В. В. (прізвище та ініціали)

Нормоконтроль (підпис) (прізвище та ініціали)

Завідувач кафедри (підпис) (прізвище та ініціали)

Рецензент (підпис) (прізвище та ініціали)

Тернопіль
2021

Міністерство освіти і науки України

Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних систем та мережі
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Осухівська Г.М.
(підпис) (прізвище та ініціали)
« » 20___ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня магістр
(назва освітнього ступеня)

за спеціальністю 123 Комп'ютерна інженерія
(шифр і назва спеціальності)

студенту Яворській Христині Володимирівні
(прізвище, ім'я, по батькові)

1. Тема роботи методи і засоби проектування платформи „Low code / No code" в комп'ютерних системах

Керівник роботи Якушин Василь Володимирович к.т.н. доц.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «28» жовтня 2021 року № 4/7-916

2. Термін подання студентом завершеної роботи

3. Вихідні дані до роботи

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. 1. Аналіз сучасного стану досліджень при проектуванні No code / Lo code платформ. 2. Формалізація об'єктів платформи 3. Розробка та тестування середовища. 4. Охорона праці та безпека в надзвичайних ситуаціях. Висновки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1 Актуальність задачі, мета дослідження і завдання дослідження
2 об'єкт дослідження, предмет дослідження, наукова новизна одержаних результатів 3. практичне значення одержаних результатів, публікації
4 порівняння візуальної мови програмування зі звичною
3. візуальна мова програмування. Огляд роботи програми
6. Головні елементи платформи, прототипи блоків
7 Загальний вигляд програми та приклади роботи
8. Висновки

\

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	завдання видав	завдання прийняв
Охорона праці та безпека в надзвичайних ситуаціях	Осуховська І.М. зав. каф. КС Стадник І.Я. проф. каф. ОХ		

7. Дата видачі завдання

КАЛЕНДАРНИЙ ПЛАН

№ з.п.	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз сучасного стану дослідження при проектуванні платформ "Low code/No code"	01.11.21	Виконано
2	Формалізація платформ Low code/No code	07.11.21	Виконано
3	Розробка та тестування платформи Low code/No code в комп'ютерних системах	14.11.21	Виконано
4	Охорона праці та безпека в надзвичайних ситуаціях	01.12.21	Виконано
5	Оформлення пояснювальної записки	05.12.21	Виконано
6	Оформлення графічного матеріалу	08.12.21	Виконано
7	Попередній захист кваліфікаційної роботи магістра	15.12.21	Виконано
8	Захист кваліфікаційної роботи	23.12.21	

Студент _____ (підпис) Яворська Х.В. (прізвище та ініціали)

Керівник роботи _____ (підпис) Яцишин В.В. (прізвище та ініціали)

АНОТАЦІЯ

Методи і засоби проектування платформ “Low code/No code” в комп’ютерних системах // Дипломна робота магістра // Яворська Христина Володимирівна // ТНТУ, Комп’ютерна інженерія, група СІм-61 // Тернопіль, 2021 // с. – 73, рис. – 41, аркушів А1 –8 , додат. –1 , бібліогр. – 16.

Ключові слова: LOW CODE, NO CODE, ВІЗУАЛЬНЕ ПРОГРАМУВАННЯ, СЕРЕДОВИЩЕ, ПЛАТФОРМА.

Метою проекту є дослідження методів і засобів розробки середовища, яке допоможе швидше та доступніше створювати продукти..

У першому розділі проведено аналіз середовищ LC/NC розробки та їх класифікацію. Також досліджено технології і засоби розробки проектування платформи і проаналізовано аналіз особливостей розробки програм.

У другому розділі запропоновано формалізацію об’єктів та процесів. Розроблено прототипи блоків а також описано арифметичні операції. Розроблено алгоримт роботи оператору if else і циклу for.

У третьому розділі було спроектовано та реалізовано Low code/No code платформу на основі визначених функціональних вимог, побудовано алгоритми роботи додатку і забезпечено його верифікацію шляхом різних видів тестування ПЗ.

У четвертому розділі було розглянуте питання з охорони праці, а саме умов праці та розробка заходів при роботі користувача, а також проаналізовано питання організації оповіщення і зв’язку у надзвичайних ситуаціях техногенного та природного характеру..

ABSTRACT

Methods and tools for designing "low code / No code" platforms in computer systems /Master thesis / Yavorska Khrystyna / TNTU, Computer engineering, group CIm-61 // Ternopil, 2021// p. - 73, fig. – 41, Sheets A1 – 8, Add – 1, Ref. – 16.

Keywords: LOW CODE, NO CODE, VISUAL PROGRAMMING, ENVIRONMENT, PLATFORM.

The aim of the project is to study methods and tools for developing an environment that will help create products faster and more accessible.

The first section analyzes the LC / NC development environments and their classification. The technologies and means of platform design development are also studied and the analysis of software development features is analyzed.

The second section proposes the formalization of objects and processes. Prototypes of blocks are developed and arithmetic operations are described. An algorithm for the operation of the if else operator and the for loop has been developed.

In the third section, the Low code / No code platform was designed and implemented on the basis of certain functional requirements, algorithms for the application were built and verified by various types of software testing.

The fourth section considered the issue of labor protection, namely working conditions and the development of measures for the user's work, as well as analyzed the organization of notification and communication in emergencies of man-made and natural nature.

ЗМІСТ

ПЕРЕЛІК ОСНОВНИХ ПОЗНАЧЕНЬ І СКОРОЧЕНЬ

NC	No code
LC	Low code
ПЗ	Програмне забезпечення
VS	Misual studio
WPF	Windows Presentation Foundation
ПК	Персональний комп’ютер
ОП	Охорона праці

ВСТУП

Актуальність теми. Low-Code і No-Code стають одними з найбільш популярних технологій, які розиваються у 2021 році. За оцінками Gartner, уже налічується понад 300 рішень на ринку, які так чи інакше позиціонують себе в цій групі. За прогнозами Forbes, ринок Low / No-Code досягне обсягу $ 187 млрд до 2030 року, а більше 65% робіт, пов'язаних з розробкою додатків, буде здійснюватися з використанням Low/No Code рішень.

На стогодні спостерігається тренд щодо зростання і впровадження інформаційних технологій у традиційні види бізнес-діяльності. Організаціям потрібно мати можливість швидко проводити експерименти і масштабувати бізнес-процеси з врахуванням зміни їх контексту, оперативно запускати нові цифрові сервіси та переводити бізнес в онлайн. Бізнес-потреби значно перевищують можливості IT, і це посилюється кризою на ринку розробників та інженерів. Поширення віддаленої роботи і зростання потреб в цифрових проектах призводять до потужної конкуренції роботодавців за IT-фахівців, що також породжує різке зростання зарплат.

З приходом нових поколінь змінюються не тільки споживачі, але й кандидати і співробітники. Тренд на широкопрофільних фахівців стає всебільш помітним і сильнішим. Швидкий розвиток технологій, ускладнення технолоогій реалізації IT- продуктів, велика кількість фреймворків і рішень призвело до появи великої кількості вузько спеціалізованих фахівців, складних команд і багатоланкової структури їх управління. Зараз все більше починають цінуватися люди з широкими компетенціями, які можуть одночасно володіти бізнес-контекстом і мати можливість імплементувати / автоматизувати процеси самостійно.

Актуальність задач розробки платформ LC/NC зумовлена масовим розпорошенням організацій, яке відбулося протягом останнього року, що простимулювало розвиток LC та NC. З початку кризи COVID-19 кількість керівників, які назвали платформи розробки LC I NC своїми найважливішими

інвестиціями в автоматизацію, зросла майже втричі — з 10% до 26%. Крім того, KPMG виявила, що 100% підприємств, які впровадили платформу розробки LC і NC, отримали рентабельність інвестицій завдяки цим ініціативам.

Актуальність розробки і впровадження середовищ Low code/No code обгрунтована зменшенням часу реалізації програмного забезпечення у комп'ютерних системах, а також зниженням порогу входу спеціалістів без фахової освіти в IT. Це дозволяє підвищити ефективність управління бізнес-процесами та забезпечити конкурентоспроможність організацій на ідповідному ринку.

Мета кваліфікаційної роботи: дослідження методів і засобів проектування платформ "Low code/No code" в комп'ютерних системах.

Завдання дослідження:

- аналіз публікацій і практик розробки платформ LC/NC ;
- формалізація об'єктів предметної області при створенні платформи;
- проектування архітектури та розробка алгоритму роботи платформи у комп'ютерних системах;
- тестування розробленого програмного забезпечення у комп'ютерних системах.

Об'єкт дослідження: процес програмної імплементації платформ Low Code/ No Code.

Предмет дослідження: моделі, методи і програмні засоби реалізації платформ Low Code/ No Code.

Методи дослідження. Щоб вирішити поставлену задачу було використано наступні методи: аналіз – для області застосування та можливі засоби для вирішення проблеми; проектування та програмування – для закладення: гнучкої, маштабованої системи архітектури, її реалізація з використанням всіх правил програмування; тестування – для перевірки коректності роботи системи в різних ситуаціях.

Наукова новизна одержаних результатів. Наукова новизна одержаних результатів:

- запропоновано та формалізовано процес створення платформи Low Code/No Code на основі класичних підходів до проектування блок-схем, що дає змогу знизити поріг входу фахівців при прозробці комп'ютерних систем, а також збільшити кількість інженерів, які без спеціалізованої освіти можуть розробляти програмні продукти;
- набули подальшого розвитку методи і засоби розробки програмних додатків шляхом атоматизації процесів та спрощення роботи з візуальними інструментами, що дало змогу підвищити ефективність розробки комп'ютерних систем.

Практичне значення одержаних результатів. Розроблено платформу LC/NC засобами: C#, Kanban, WPF для забезпечення взаємодії обєктів між собою та запуску програми.

Даний підхід дає змогу людям без спеціалізованої освіти створювати власні програмні продукти та покращувати наявні.

Публікації. Результати дослідження апробовано на X міжнародній науково - технічній конференції молодих учених і студентів «Аналіз особливостей візуальних мов програмування» (24-25 листопада 2021 р.) Тернопільського національного технічного університету імені Івана Пулюя та на IX науково-технічній конференції Тернопільського національного технічного університету імені Івана Пулюя «Відмінності Low-code/No-code розробки» (8-9 грудня 2021 року) у вигляді тез конференцій.

Структура роботи. Робота складається з пояснювальної записки та графічної частини. Пояснювальна записка складається із вступу, 4 розділів, висновків, списку використаних джерел та додатку (-ів). Обсяг роботи: пояснювальна записка – 73 арк. формату А4, графічна частина – 8 аркушів формату А1.

РОЗДІЛ 1
АНАЛІЗ СУЧАСНОГО СТАНУ ДОСЛІДЖЕНЬ ПРИ ПРОЕКТУВАННІ NO CODE/LO CODE ПЛАТФОРМИ

1.1. Класифікація NC/LC платформ

Сучасна епоха проектування та реалізації комп'ютерних інформаційних систем характеризується, у більшості своїх випадків, застосуванням мов програмування з визначеними синтаксичними конструкціями у вигляді тексту. Це передбачає написання рядків коду для виконання конкретного завдання у вигляді строго визначеної послідовності команд, наприклад у вигляді коду C або C++. Альтернативою таким мовам програмування є візуальні мови, в основі яких лежить «програмування графічними блоками», яке підтримує концепцію «Drag&Drop». Перевагою використання таких мов програмування є зниження порогу входу розробників програмного забезпечення комп'ютерних систем, зменшення витрат часу для програмування системи, а також зручність і простота модифікації визначених алгоритмів.

Будь-яка мова, яка використовує графічне представлення об'єктів програмування, а під візуальним інтерфейсом яких виконується програмний код, що написаний за допомогою синтаксичних конструкцій, представляє собою візуальну мову програмування.

Розглянемо деякі платформи low-code/no-code розробки:

1) Unreal Engine 4 Blueprints зображений на рис 1.1.

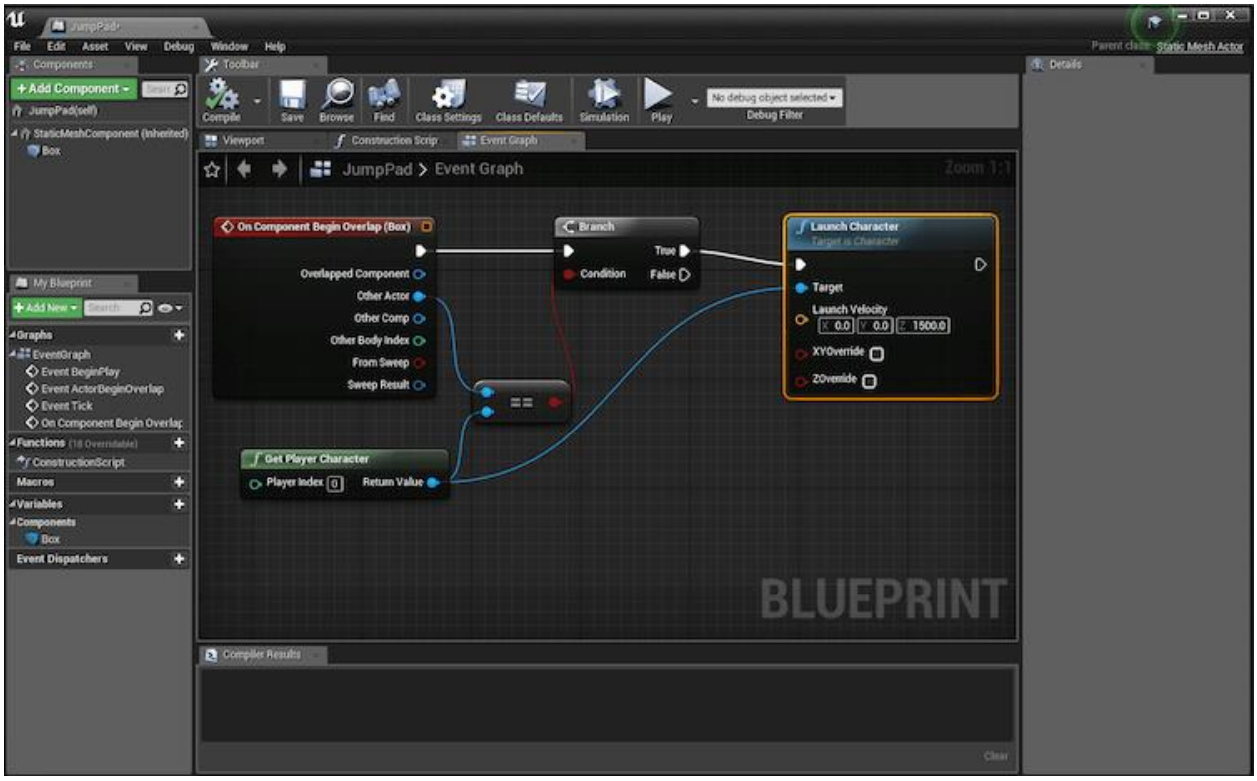

Рис. 1.1. Unreal Engine 4 Blueprints

Blueprints — це візуальна система сценаріїв у Unreal Engine 4, яка є швидким способом розпочати створення прототипу вашої гри. Замість того, щоб писати код рядок за рядком, ви робите все візуально: перетягуєте вузли, встановлюєте їхні властивості в інтерфейсі користувача та перетягуєте дроти для підключення.

Окрім того, що Blueprints є швидким інструментом створення прототипів, він також полегшує для тих, хто не програміст, занурення та розпочати створення сценаріїв.

2) Unity visual scripting зображено на рис. 1.2.

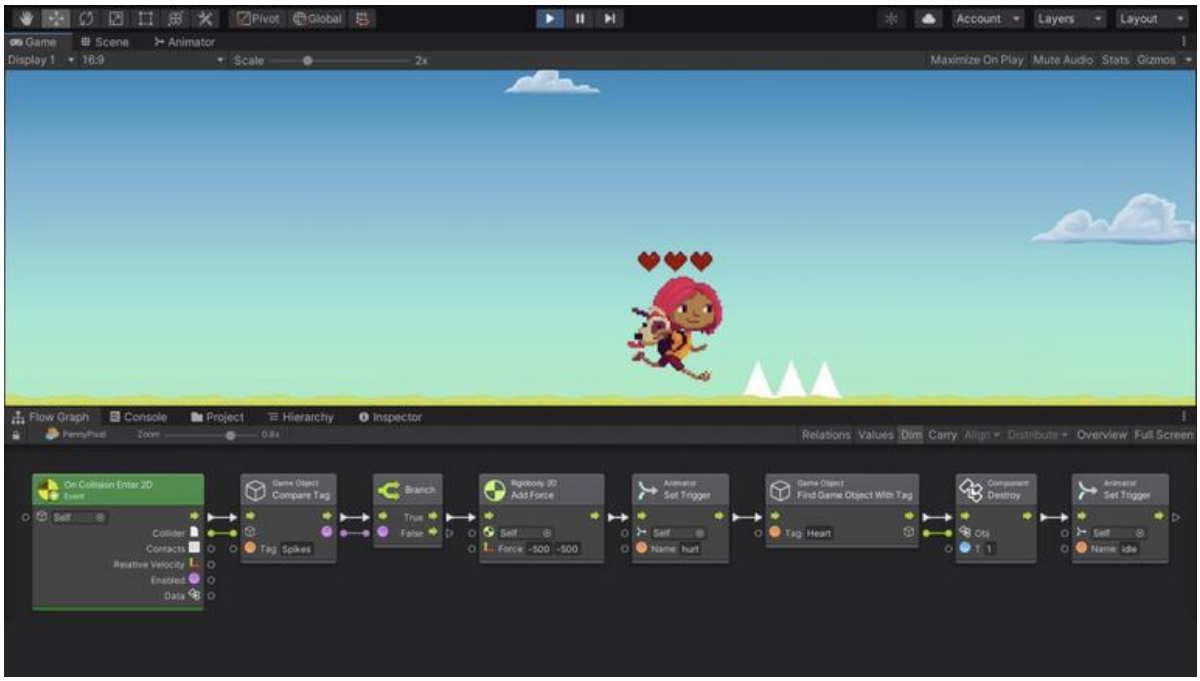

Рис. 1.2. Unity visual scripting

Візуальні сценарії в Unity допомагають членам команди створювати логіку сценаріїв за допомогою візуальних графіків із перетягуванням замість того, щоб писати код з нуля. Це також забезпечує більш безперебійну співпрацю між програмістами, художниками та дизайнерами для швидшого створення прототипів та ітерацій.

Візуальні сценарії є чудовим рішенням як для дизайнерів, так і для художників, які хочуть перевірити ідеї, внести зміни або зберегти більш прямий контроль над своєю роботою в Unity. люди без спеціалізованої освіти також можуть використовувати графіки вузлів, створені більшою кількістю членів технічної команди.

3) програмний продукт «Glide» зображено на рис. 1.3.

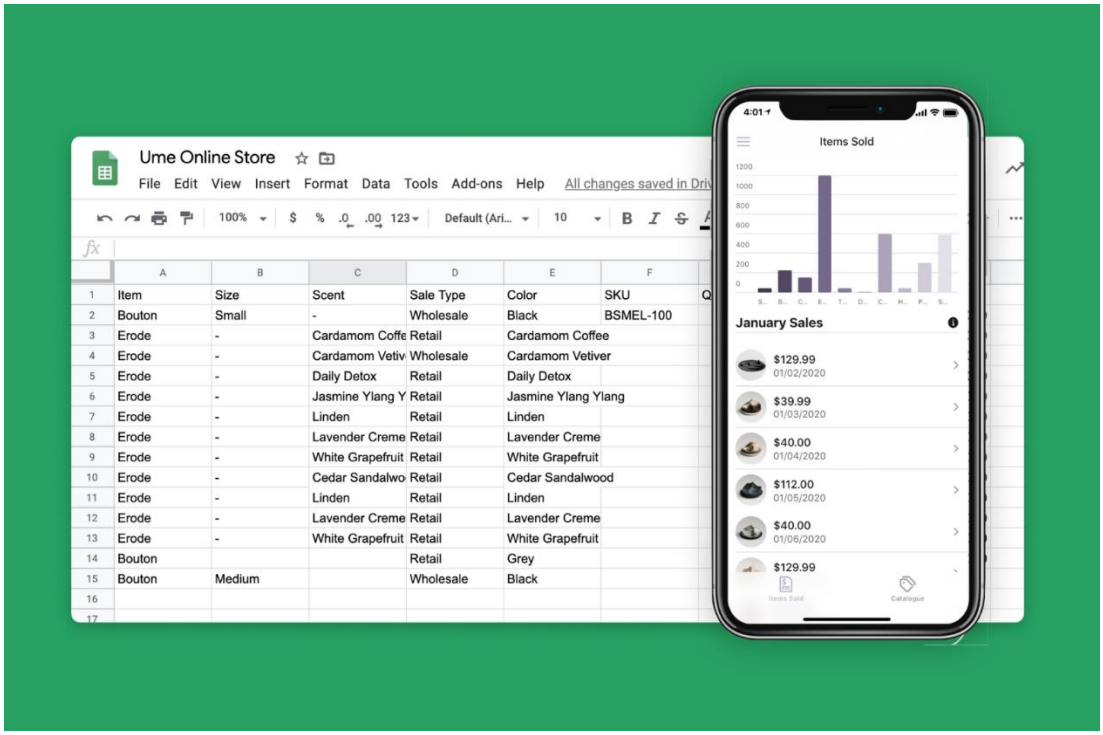

Рис. 1.3. Програмний продукт «Glide»

Glide - це інструмент, який, буквально за 5 хвилин, дозволяє створювати програми, які базуються на таблицях Google. Зокрема, йдеться про мобільні програми. При цьому, доки користувачеві цього засобу не потрібні певні можливості, застосовувати його він може безкоштовно. І досвід програмування для роботи з Glide не потрібний. Якщо вам чи комусь із ваших знайомих потрібна програма, його можна швидко створити за допомогою Glide. Слід зазначити, що на сайті проекту є якісні шаблони та посібники.

4) Платформа «Webflow» зображено на рис. 1.4.

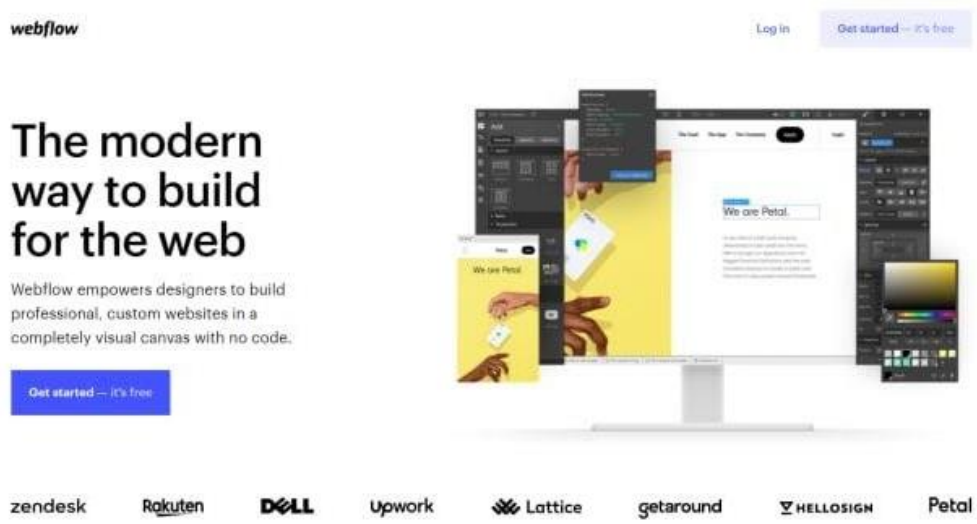

Рис. 1.4. Платформа «Webflow»

Webflow – це платформа, яка дозволяє створювати, тестувати та публікувати сайти без необхідності писати код (або HTML-розмітку). Створення хорошого, високоякісного сайту - це, традиційно, завдання, на виконання якого потрібно багато часу. Більше того, в минулі часи навіть для створення персонального веб-проекту, досить сучасного та функціонального, були потрібні чималі знання. Тепер, завдяки таким інструментам, як Webflow, створювати привабливі та функціональні сайти стало набагато легше, ніж раніше. По суті, головне завдання того, хто користується Webflow, полягає в тому, щоб розібратися в тому, що саме йому потрібно і застосувати відповідні механізми платформи для досягнення бажаного.

5) Платформа «Adalo» зображено на рис. 1.5.

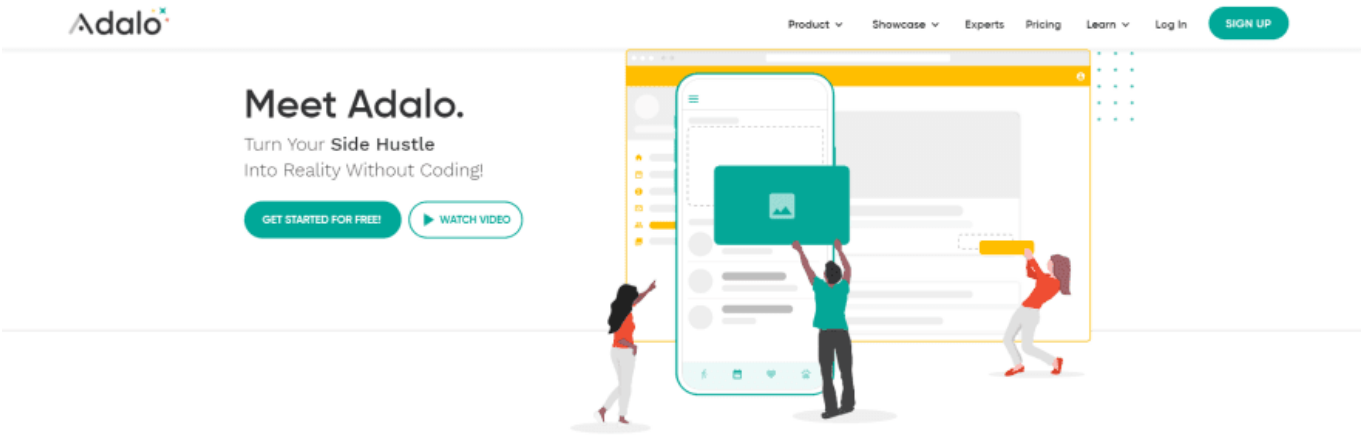

Рис. 1.5. Платформа «Adalo»

Adalo – це платформа, яка дозволяє дуже швидко створювати інтерактивні програми. Скористатися їй зможе навіть той, хто не знає жодної мови програмування. Хоча платформа Adalo схожа на Glide, у ній акцент робиться на візуальну розробку додатків у стилі «drag-and-drop», що дозволяє її користувачам відразу ж бачити те, що вони створюють.

Підведемо підсумки по наведених платформах, а саме можна зрозуміти, що Low / No-Code - це можливість збирати системи з кубиків, готових блоків і абстракцій, які можуть являти собою як готові операції взаємодії з зовнішніми сервісами і API, так і елементи UI інтерфейсу, а також візуальне уявлення потоків завдань і можливість створювати логіку без написання коду або з мінімальним написанням коду, використовуючи скриптові мови.

1.2. Технології розробки

Low-Code і No-Code стають одними з найбільш обговорюваних тем в 2021 році. За оцінками Gartner, налічується понад 300 рішень на ринку, так чи інакше позиціонують себе в цій групі. За прогнозами Forbes, ринок Low / No-Code досягне обсягу $ 187 млрд до 2030 року, а більше 65% робіт, пов'язаних з розробкою додатків, буде здійснюватися з використанням Low / No Code рішень. Дійсно, на сьогодні до цього є всі передумови. Зростання потреб бізнесу в IT

зростає, організаціям потрібно мати можливість швидко проводити експерименти і масштабувати свій бізнес під швидко мінливий контекст, оперативно запускати нові цифрові сервіси та переводити бізнес в онлайн. Бізнес-потреби значно перевищують можливості IT, і це посилюється кризою на ринку розробників та інженерів. Поширення віддаленої роботи і зростання потреб в цифрових проектах призводять до сильної конкуренції роботодавців за IT-фахівців, що також породжує різке зростання зарплат.

З приходом нових поколінь змінюються не тільки споживачі, але й кандидати, і співробітники. Тренд на широкопрофільних фахівців стає помітний все сильніше. Швидкий розвиток технологій, ускладнення топологій IT- 13 продуктів, велика кількість фреймворків і рішень призвело до появи великої кількості вузько заточених фахівців, складних команд і багатоланкової структурі їх управління. Зараз все більше починають цінуватися люди з широкими компетенціями, які можуть одночасно володіти бізнес-контекстом і мати можливість імплементувати / автоматизувати процеси самостійно.

Важливими аспектами при виборі технологій та засобів розробки були доступність і складність розробки а також якість і підтримка програмного продукту. Саме тому були обрані такі засоби для реалізації платформи:

1) мова програмування C# 8.0;
2) Visual Studio Professional 2019;
3) систему контролю версій Git;
4) Windows Presentation Foundation.

C# - є однією із найпопулярніших мов програмування вже протягом багатьох років. На сьогоднішній момент мова програмування C# одна з найпотужніших мов, що швидко розвиваються і затребуваних в IT-галузі. Зараз на ньому пишуться різні програми: від невеликих десктопних програм до великих веб-порталів і веб-сервісів, що обслуговують щодня мільйони користувачів.

Git — це безкоштовна розподілена система контролю версій з відкритим кодом, призначена для швидкого та ефективного керування всіма проектами від малих до великих.

Visual Studio – повноцінне середовище розробки з багатим функціоналом. Підходить для великих проектів (web, enterprise). Повна підтримка технологій. Зручний супровід та масштабування коду. Зручна командна робота.

WPF — використовується для створення програмних продуктів з інтерфейсом користувача для операційних систем Windows.

Для вчасного виконання та можливості відслідкування етапів розробки магістерської роботи використовувала метод розробки «Канбан» [4] (рис. 1.6.)

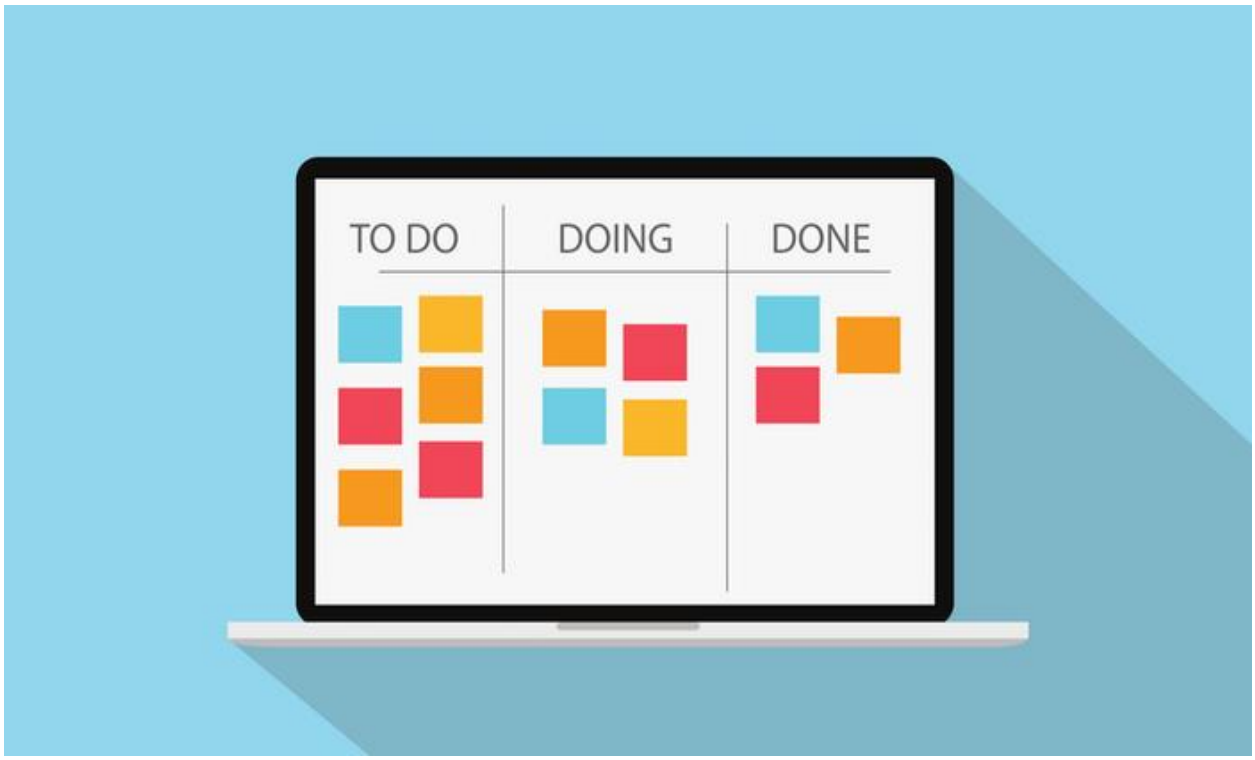

Рис. 1.6. Приклад організації «Канбан»

Проаналізувавши поставлені задачі та переглянувши наявні програмні засоби були обрано вибрані інструменти для створення якісної платформи LC/NC.

1.3. Аналіз особливостей NC/LC середовищ

Low code — це візуальний підхід до розробки програмного забезпечення, який оптимізує весь процес розробки для прискорення доставки. Завдяки LC ви можете абстрагувати й автоматизувати кожен крок життєвого циклу програми, щоб спростити розгортання різноманітних рішень. На рис. 1.7 можна переглянути як будуть виглядати програм написані LC.

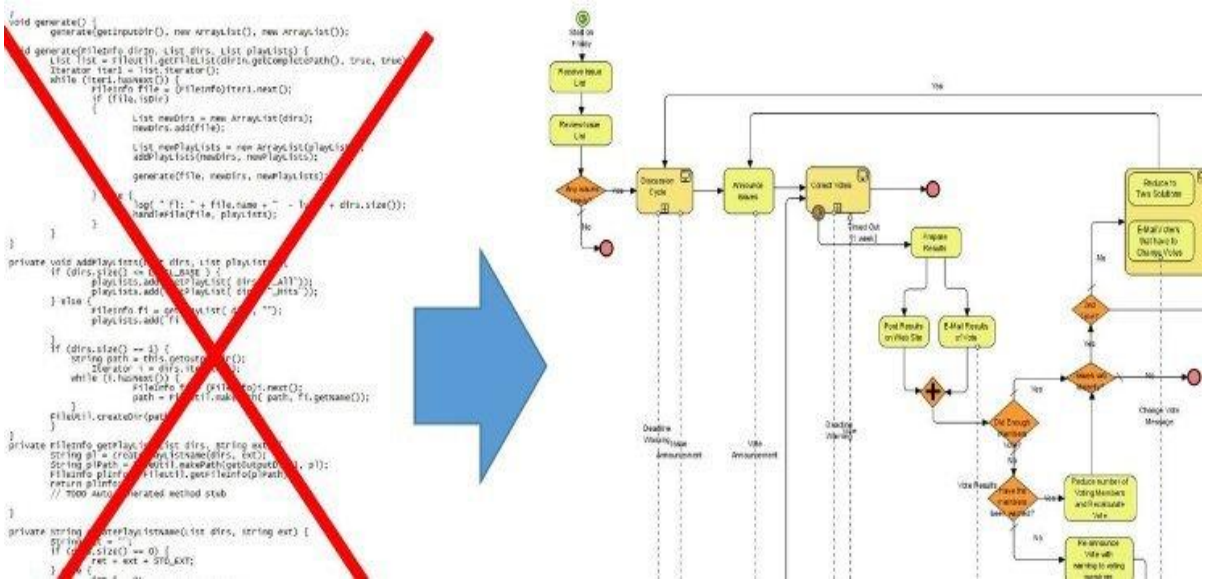

Рис. 1.7. Low code програма

Масове розпорошення організацій, яке відбулося протягом останнього року, прискорило рух із LC та NC, свідчить опитування KPMG. З початку кризи COVID-19 кількість керівників, які назвали платформи розробки LC I NC своїми найважливішими інвестиціями в автоматизацію, зросла майже втричі — з 10% до 26%. Крім того, KPMG виявила, що 100% підприємств, які впровадили платформу розробки LC I NC, отримали рентабельність інвестицій завдяки цим ініціативам.

Протягом наступних двох років, за прогнозами Gartner, більше половини середніх і великих підприємств застосують платформи програм з NC. Опитування 324 організацій, проведене Unisphere Research/Information Today, Inc., показало, що принаймні 76% вже мають принаймні частину програм,

розроблених за межами традиційних IT-відділів або постачальників послуг. Вони розробляють необхідні заявки за кілька тижнів, і лише 17% повідомляють, що терміни виконання перевищують три місяці. Опитування показало, що розробники, які не належать до IT, походять із різноманітного досвіду, але, здебільшого, є досвідченими користувачами та розробниками, які працюють у ділових відділах, які створюють додатки. Опитування також показало, що проблеми розробки з низьким рівнем коду та без коду включають безпеку даних і проблеми з навчанням правильним методам програмування та обробкою даних.

Функції NC/LC середовищ:

1) візуальне моделювання;

2) drag-and-drop: функція для перетягування елементів на робочу область;

3) модульність;

4) кросплатформність.

Терміни NL і LC часто згадуються разом, оскільки вони схожі. Як платформи з низьким кодом, так і без коду використовують візуальні інтерфейси, щоб дозволити користувачам розробляти власні бізнес-рішення без великих знань у кодуванні.

Основна відмінність між платформами з низьким кодом і без коду, як випливає з їх назви, полягає в тому, що платформи з низьким кодом все ще можуть включати кодування в певних випадках, тоді як жодні платформи коду не потребують абсолютно ніякого кодування.

Це, по суті, означає, що жодна технологія коду не розроблена спеціально для розробників без додаткової освіти, тоді як технологія з низьким кодом обслуговує як розробників без додаткової освіти, так і професійних розробників. Оскільки платформи з низьким кодом все ще можуть включати кодування, вони здатні створювати більші та складніші додатки, ніж платформи без коду зазвичай. Для кращої гнучкості та контролю над циклом розробки, передові компанії розгортають платформи, які поєднують як технології з низьким кодом, так і без коду.

Використання платформи NC/LC має безліч переваг:

– швидкість: за допомогою низького коду ви можете створювати додатки для кількох платформ одночасно і показувати зацікавленим сторонам приклади роботи за кілька днів або навіть годин;

– більше ресурсів: якщо ви працюєте над великим проектом, вам більше не доведеться чекати, поки розробники з професійною освітою закінчать ще один тривалий проект, а це означає, що все виконується швидше і за меншу вартість;

– низький ризик/висока рентабельність інвестицій: завдяки низькому коду надійні процеси безпеки, інтеграція даних і міжплатформна підтримка вже вбудовані та їх можна легко налаштувати, що означає менше ризику та більше часу для зосередження на своєму бізнесі;

– розгортання одним клацанням миші: із низьким кодом достатньо одного клацання, щоб надіслати вашу програму в робочий стан. День запуску більше не тягне за собою нерви;

Порівняльний вигляд додатків LC/NC та традиційний код зображено на рис. 1.8

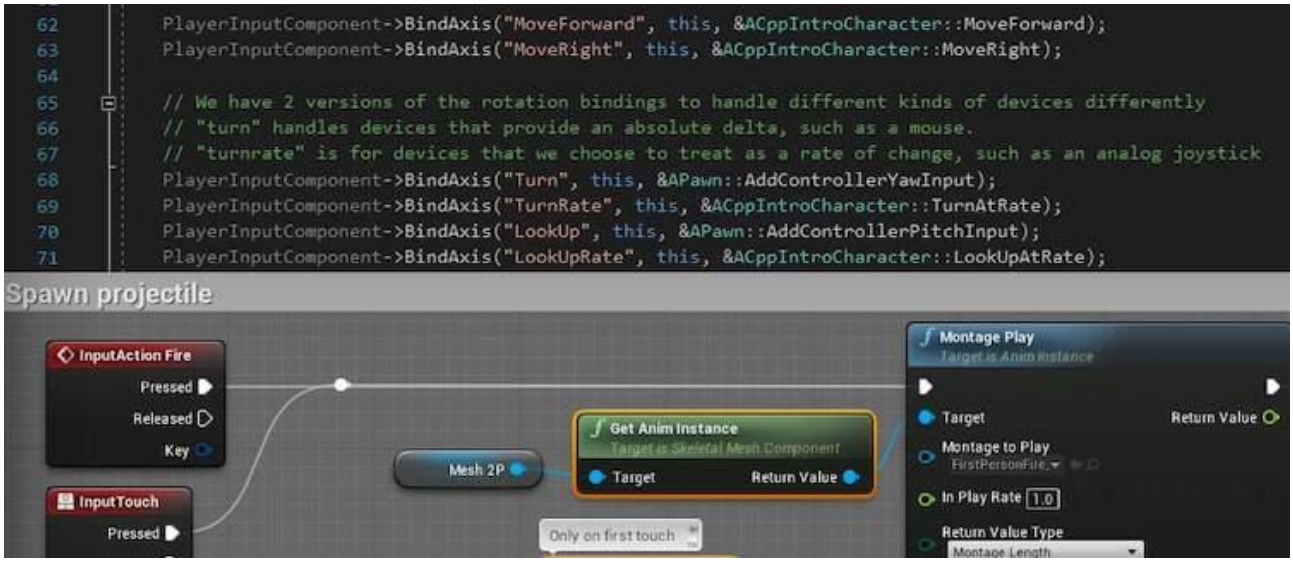

Рис. 1.8. Порівняння LC/NC та традиційного коду

У світі, де комп’ютери є частиною повсякденного життя більшості людей, корисно навчитися виконувати тривалі та/або складні завдання. Візуальні мови програмування спрямовані на те, щоб люди, які програмують комп’ютери, мали

найкращий досвід, обходячи деякі проблеми синтаксису та перекладу з ментального плану до виконуваної програми. Однак текстові мови виходять на перше місце, коли мова йде про мови програмування, які найчастіше використовуються. Візуальні мови програмування пропонують швидке введення в світ програмування, яке багатьма способами зменшує труднощі в цій області, роблячи програмування ігровим і творчим. З іншого боку, межі мови досягаються швидко і вимагають від учнів перейти на текстову мову. Крім того, візуальна мова програмування не допомогла студентам навчитися планувати та усувати неполадки у своїх програмах. При переході до текстової мови програмування, де планування та усунення несправностей необхідні етапи процесу, це може стати проблемним.

1.4. Висновки до розділу

На основі проведеного аналізу отримано наступні результати:

1) проведено аналіз існуючих програмних продуктів, у результаті якого встановлено типи продуктів і аналіз особливостей LC/NC середовищ;

2) на основі дослідження технологій розробки середовищ визначено оптимальні технології програмного забезпечення для їх реалізації;

3) проведено аналіз особливостей LC/NC, у результаті якого визначено функціональні вимоги до них.

РОЗДІЛ 2
ФОРМАЛІЗАЦІЯ ОБ’ЄКТІВ ПЛАТФОРМИ

2.1. Особливості навчання користувачів

Профіль звичайного користувача комп’ютера суттєво змінився за останнє десятиліття. Розробка додатків більше не є виключною сферою комп’ютерного вченого. Початківці користувачі починають розробляти власні програми. Існує гостра потреба полегшити процес розробки програмного забезпечення. Візуальне програмування може запропонувати рішення цієї проблеми. Крім того, оскільки можливості комп’ютерної графіки продовжують вдосконалюватися, логічним кроком є скористатися перевагами цієї передової технології. Візуальне програмування може стати партнером зростаючих апаратних можливостей.

Вважається, що люди проходять п'ять етапів, коли оволодівають новим навиком. Всі люди починають свій шлях як новачки і в кінцевому підсумку можуть досягти рівня експерта, як правило, з великою практикою.

П'ять етапів набуття навичок(рис. 2.1):

1. новачок: новачок вивчає об'єктивні факти, особливості та правила для визначення дій на основі фактів і правил, яких вони засвоїли. Новачок не адаптує те, що він робить до контексту;

2. досвідчений початківець: досвідчені початківці почали розпізнавати й обробляти ситуації, які не охоплені наведеними фактами, функціями та правилами. Просунуті новачки чутливі до контексту, але не зовсім розуміють, що вони роблять;

3. еомпетентний спцеіаліст: Рівень компетентності досягається, коли набувач навичок розглядає всю ситуацію і свідомо обирає організований план для досягнення мети;

4. спеціаліст: вправність досягається, коли набувач навичок більше не повинен свідомо міркувати через усі кроки для визначення плану для досягнення мети;

5. експерт: «Експерт, як правило, знає, що робити, ґрунтуючись на зрілому й напрацьованому розумінні».

Рис. 2.1. Етапи набуття компетенції

На придбання навичок впливає ряд факторів, одним з яких є особистість. Люди навчаються по-різному і часто віддають перевагу одному методу навчання перед іншим. Ці методи навчання зазвичай називають стилем навчання учня. Ці стилі описують, як люди навчаються, дивлячись на різні виміри навчання, такі як сприйняття, введення, організація, обробка та розуміння. Зазвичай людина часто віддає перевагу одному стилю у вимірі над іншим, і прикладом може бути особа, якавіддає перевагу глобальному стилю під час розуміння предмета, але хоче, щоб вхід був візуальним, а процес — активним.

Опишемо різні стилі навчання:

– «Учні, які відчувають», яким подобається працювати з фактами та реальними об’єктами і люблять бачити зв’язок із «реальним світом» з тим, що вони вивчають. Навпаки, «інтуїтивні учні» віддають перевагу роботі з абстрактним теоретичним матеріалом і зазвичай працюють швидше, ніж ті, що чують;

– «Візуальні учні» найкраще запам’ятовують те, що вони бачать. Діаграми та символи – це методи введення, які працюють для візуального учня;

– «Слухові учні», з іншого боку, добре запам’ятовують те, що вони чують, і найкраще те, що вони чують, а потім говорять;

– «Учні-кінестетики» найкраще запам’ятовують те, що вони на смак, дотик і запах;

– «Учні індуктивного навчання» та «дедуктивні учні» є майже протилежними, коли індуктивні учні беруть деталі, як-от спостереження, і переходять до узагальнення, тоді як дедуктивні учні беруть принципи та загальні концепції та переходять до наслідків і застосувань;

– «Активні учні» — це «виконавці» в тому сенсі, що вони швидше навчаються, експериментуючи, в той час як «учні, що рефлексують», скоріше спостерігають за процесами навчання;

– «Учні, які навчаються послідовно», хочуть отримати новий вхід у, якщо можливо, хронологічному та/або логічному порядку, по черзі.

Більшість людей переважно візуальні, активні та сенсорні учні.

Навчаючи новачків програмуванню, вчитель завжди ризикує зосередити увагу студентів на синтаксичних питаннях, а не на обчислювальній семантичній потужності мови програмування. Такий синтаксис фокусування не дозволяє студентам-початківцям зрозуміти роль мови програмування як інструменту для вирішення проблем.

Опишемо як слід застосувати, педагогіку щоб покращити розуміння програмування новачками:

1. вивчайте синтаксис і семантику однієї функції мови за раз;

2. навчіться поєднувати цю мовну функцію з відомими навичками проектування для розробки програм для вирішення проблем (це розширює навички проектування студентів і включає шаблони та процедурні навички, такі як планування, тестування та переформулювання) ;

3. розвивати загальні навички розв’язування задач.

Можна стверджувати, що найбільшою перевагою VPL під час викладання та навчання може бути те, що він за своєю суттю допомагає учням у вирішенні проблем, не викликаючи синтаксичних помилок. VPL перемикає фокус з синтаксису на семантику, а також дозволяє користувачам спробувати своє рішення, не будучи повним.

Щоб більш чітко зрозуміти, які можливості система повинна обробляти, щоб її віднести до мови візуального програмування, почнеться більш детальне обговорення візуалізації.

Візуалізація використовується для покращення розуміння комп'ютерних програм. Візуалізація ділиться наа три основні гілки: візуалізація програми, візуалізація алгоритму та візуалізація даних- рис. 2.2

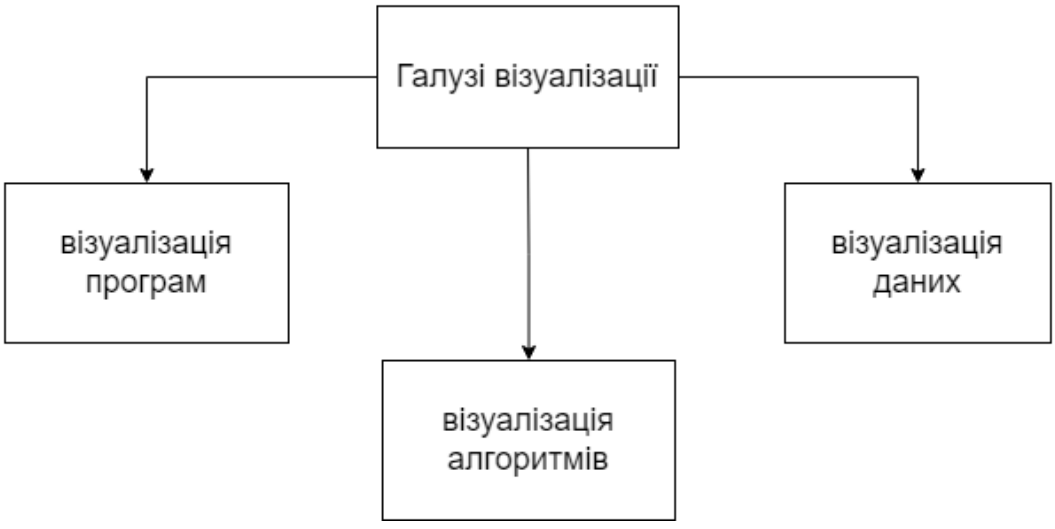

Рис. 2.2. Галузі візуалізація

У візуалізації програми графіка використовується для ілюстрації деяких аспектів програми після її написання і може бути як статичною, так і динамічною

візуалізацією програми. Методи статичної візуалізації програми включають створення блок-схем і красивий друк (вставка пробілів і порожніх рядків, відступи та коментарі для покращення читабельності програми). Виконання програми ілюструється або анімацією, або виділенням програмного коду, коли реалізована динамічна візуалізація програми.

Системи візуалізації алгоритмів створюють анімацію алгоритмів, щоб показати «фундаментальні операції програми, які втілюють як перетворення, так і доступ до даних і потік керування».

Візуалізацію даних також можна розділити на:

- статична візуалізація даних створює статичні графічні відображення для структур даних. Цей метод полегшує «налагодження, представляючи структури даних програмістам так, як вони малювали їх вручну на папері»;
- динамічна візуалізація даних графічно відображає стан змінних, масивів, списків, дерев та інших структур даних під час виконання програми.

По суті, візуалізація надає засіб краще зрозуміти, як працює програма після її кодування. Це прямо контрастує з візуальним програмуванням, коли програма фактично розроблена шляхом маніпулювання графічними представленнями або комбінацією піктограм і текстової інформації.

Візуальне програмування ділиться на дві ключові галузі:

- системи графічної взаємодії;
- системи візуальної мови.

Цей поділ заснований на тому, як графіка використовується для побудови програми. Системи, в яких користувач керує або дає інструкції системі для створення програми, класифікуються як системи графічної взаємодії. Візуальні мовні системи складаються із систем, у яких для визначення програми використовуються значки, символи, діаграми чи форми.

У системах графічної взаємодії послідовність дій користувача має життєво важливе значення, оскільки система «вчиться» на введених користувачами. Ця категорія є більш поширеною і, можливо, більш влучною програмою на прикладі. У більшості систем від користувача вимагається вказати все про

програму, і система може запам’ятати приклади для подальшого використання. І навпаки, деякі системи намагаються зробити висновок про загальну структуру програми після того, як користувач надав ряд прикладів, які працюють за допомогою алгоритму. Їх часто називають автоматичним програмуванням, що, як правило, було сферою досліджень штучного інтелекту.

Друга галузь візуального програмування називається візуальними мовними системами. У рамках цієї класифікації є системи, що використовують значки, символи, діаграми та форми для визначення програми.

Просторове розташування символів визначає програму. Це відрізняє візуальні мови від систем графічної взаємодії (програмування за прикладом), оскільки в системах графічної взаємодії важлива взаємодія користувача з системою, а у візуальних мовах — розташування символів на екрані.

Візуальні мови складаються з набору графічних символів, які складаються у «візуальні речення із заданим синтаксисом і семантикою». Потім візуальні речення необхідно просторово проаналізувати, щоб визначити основну синтаксичну структуру. Потім необхідно провести семантичний аналіз, щоб визначити значення візуальних речень (просторова інтерпретація). Синтаксичний та семантичний аналіз візуальної мови мало чим відрізняється від традиційного мовного підходу. Обидва типи мов необхідно проаналізувати, щоб визначити синтаксис і значення, істотна відмінність полягає в тому, що візуальні мови використовують графічні символи, а не текстові вирази традиційних мов. поділ візуальних мов на три основні категорії на основі графічної абстракції, яка використовується для створення програми:

- блок-схеми;
- значки;
- форми/таблиці.

Категорія, блок-схеми, включає візуальні мови, які надають різні типи діаграм, графіків і діаграм для побудови програм. Блок-схеми в основному складаються з потоків управління або діаграм потоків даних.

Друга категорія мов візуального програмування, піктограм, складається з візуальних мов, що використовують графічні символи або піктограми та їх взаємозв’язки для формування звичайних речень. Як було зазначено раніше, просторовий розбір та інтерпретація використовуються для забезпечення синтаксичного та семантичного аналізу відповідно.

Не існує загальноприйнятого стандарту для визначення значка. Головний критерій оформлення значка – це чітке відображення абстракції.

Останньою категорією візуальних мов є ті мови, які використовують таблиці та форми. Користувач створює програму, використовуючи таблиці або заповнюючи форми. Поширене використання цієї категорії включає розробку запитів до реляційних баз даних за допомогою таблиць і розробку офісно-інформаційних систем за допомогою форм.

Ретельна оцінка візуальної мови є складним завданням. Оскільки парадигма була нещодавно представлена, або, точніше, нещодавно відкрита широким населенням, було опубліковано мало досліджень на тему оцінки візуальної мови. Єдиним винятком є тривимірна структура Шу для характеристики та порівняння візуальних мов, як показано на рис. 2.3 . Трьома критеріями для порівняння Шу є візуальний обхват, обсяг і рівень мови. Візуальний об’єм — це міра використання мовою графічних об’єктів для конструкцій програмування. Обсяг — це показник того, чи застосовна візуальна мова лише в дуже обмеженій області чи корисна в різноманітних додатках. Нарешті, рівень мови показує, чи кваліфікується візуальна мова як мова високого чи низького рівня.

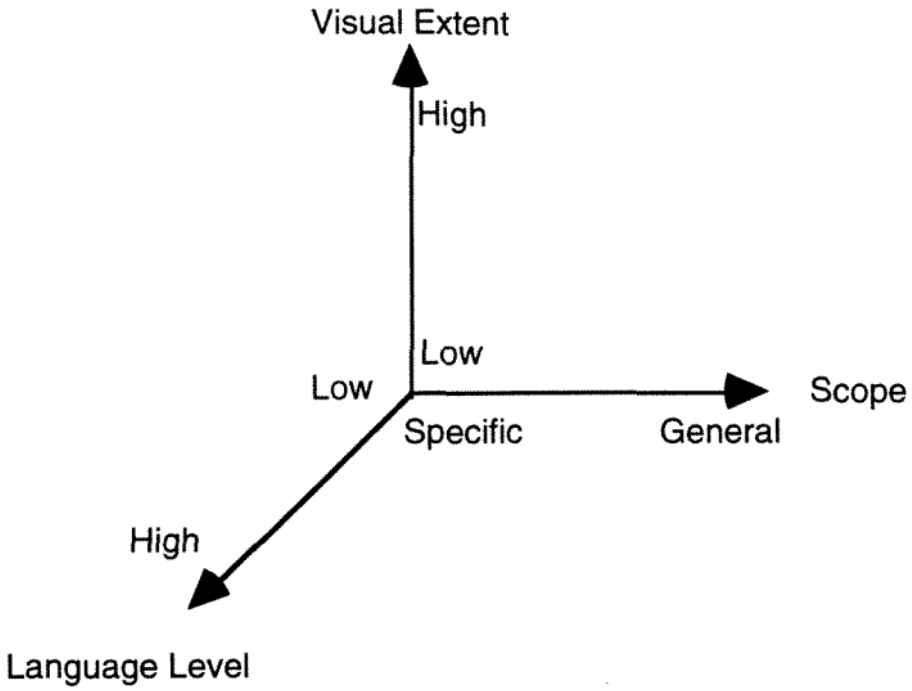

Рис. 2.3 Тривимірна структура Шу

2.2. Формалізація об’єктів платформи low code/no code

Оскільки для проектування платформи необхідно створити велику кількість обєктів, які повинні взаємодіяти між собою тому для розробки середовища необхідно описати їхні властивості.

Головними елементами платформи є:

- цикл for;
- арифметині операції (додавання, віднімання, множення та ділення);
- операції порівняння;
- оператор if.

Також кожна програма повинна містити два обєкти, а саме: блок початку та закінчення програми.

Особливістю блоку початку є те, що це єдиний блок у якого немає точки входу для передачі інформації та є тільки один вихід для передачі інформаціїї. Прототип блоку зображений на рис. 2.4

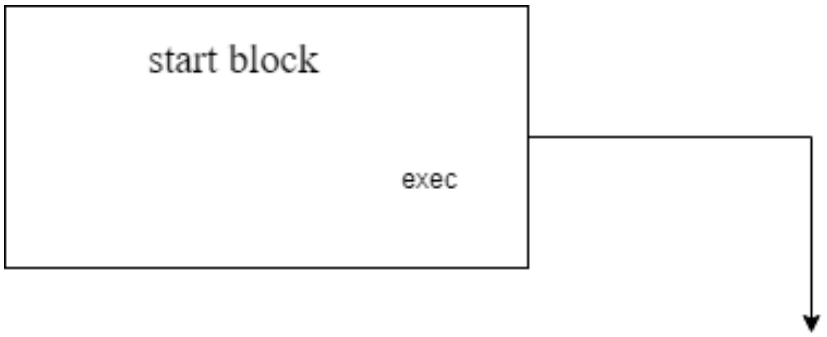

Рис. 2.4. Прототип блоку початку

Блок закінчення програми – це блок який приймає дані для завершення роботи і не передає їх дальше. Прототип блоку закінчення зображений на рис. 2.5

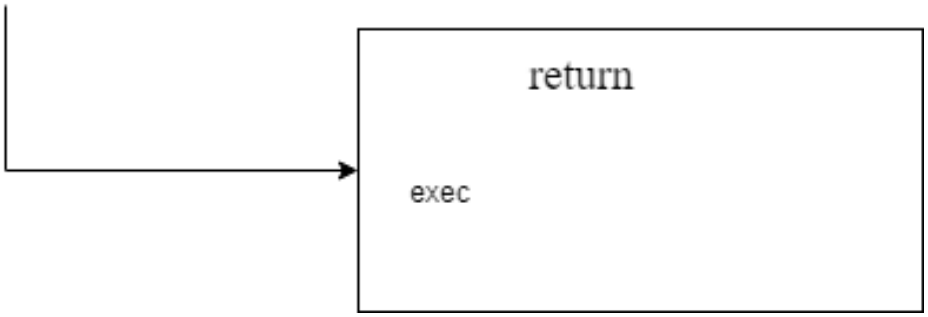

Рис. 2.5 Прототип блоку закінчення

2.2. Формалізація арифметичних операцій

Арифметичні операції складаються з 4 дій які поділенні на два блоки, а саме:

1) Прямі дії – додавання та віднімання.
2) Обернені дії – множення і ділення.

Розглянемо кожну дію більш детально. Розпочнемо з прямих дій

Додавання позначається знаком +. Та загальною формулою є:

$$c = a + b, \tag{2.1}$$

де с – сума;

a та b – доданки.

Віднімання позначається знаком –. Та загальною формулою є:

$$c = a - b, \tag{2.2}$$

де c – різниця;

a – зменшуване;

b – відємник.

Розглянемо обернені дії, а саме множення. Множення позначається знаком * і загальною формулою є:

$$c = a * b, \tag{2.3}$$

де c – добуток;

a та b – множники.

Ділення позначається знаком / і формулою є:

$$c = a/b, \tag{2.4}$$

де c – частка;

a – ділене;

b – дільник.

Арифметичні дії зображаються в програмі як блок який складається з:

- вхідних чисел;
- елемент опрацювання;
- вихідний результат.

Прототип блоку арифметиних дій зображений на рис. 2.6

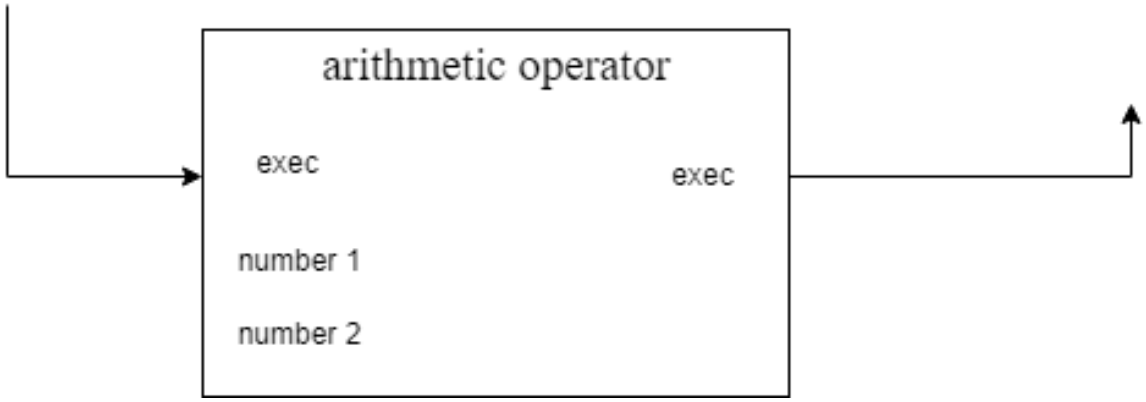

Рис. 2.6. Прототип блоку арифметичних дій

2.3. формалізація операцій порівняння

Операції порівняння представленні в 3 видах:

- більше числа;
- менше числа;
- рівне числу.

Операцій порівняння більше числа зображається знаком > . менше числа позначається знаком <, а рівне число знаком =

Прототип блоку операції порівняння зображений на рис. 2.7

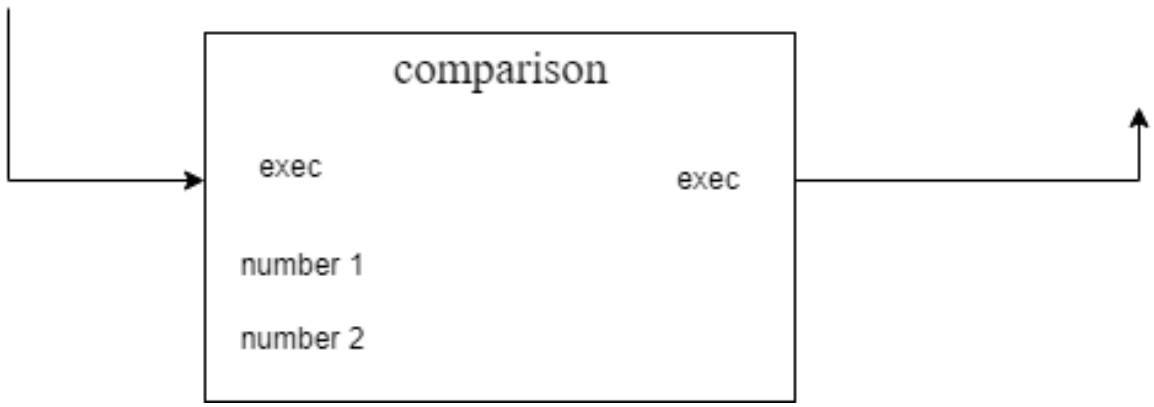

Рис. 2.7. Прототип операцій порівняння

2.4. формалізація оператору if else

Оператор if- це оператор порівняння після якого, якщо твердження істине виконується одна дія а якщо твердження неістине – виконується інша.

Оператор if не працює без блоку порівняння та істиною умови, без блоку умови коли операція не істина- оператор працює. Блок-схема роботи оператора if зображена на рис. 2.8

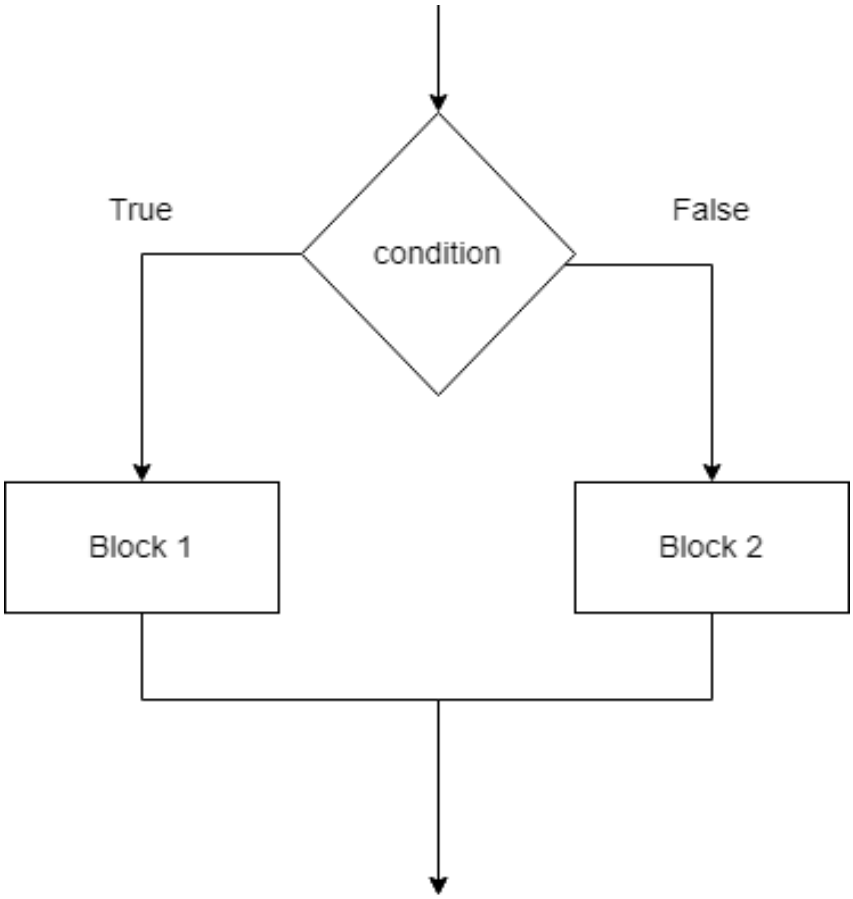

Рис. 2.8. Блок-схема оператора if

Блок оператора if складається з передачі даних у вигляді булевого значення. Та вихід для двох дій: істине та неістине значення. Прототип зображений на рис. 2.9

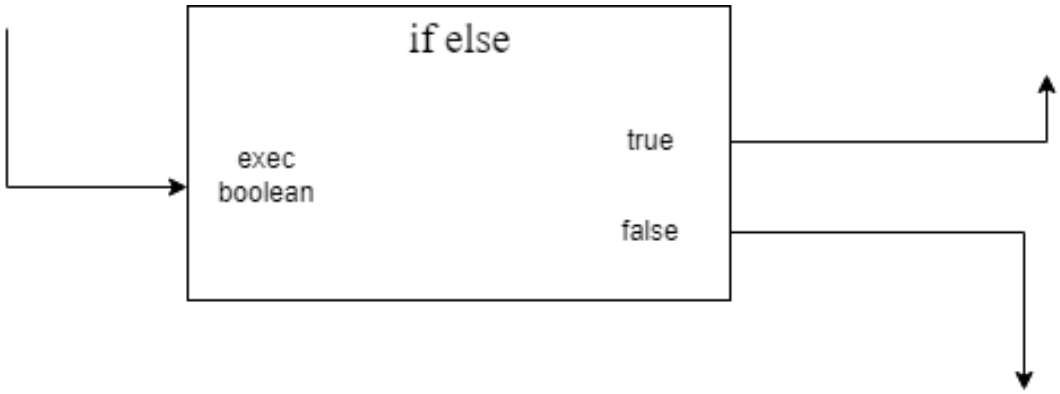

Рис. 2.9. Прототип блоку if else

Для реалізації блоку необхідно розробити алгоримти роботи програми (рис. 2.10)

```
public void TranslateIfElseNode(IfElseNode node)
{
    // Translate all the pure nodes this node depends on in
    // the correct order
    TranslateDependentPureNodes(node);

    string conditionVar = GetPinIncomingValue(node.ConditionPin);

    builder.AppendLine($"if ({conditionVar})");
    builder.AppendLine("{");

    if (node.TruePin.OutgoingPin != null)
    {
        WriteGotoOutputPinIfNecessary(node.TruePin, node.InputExecPins[0]);
    }
    else
    {
        builder.AppendLine("return;");
    }

    builder.AppendLine("}");

    builder.AppendLine("else");
    builder.AppendLine("{");

    if (node.FalsePin.OutgoingPin != null)
    {
        WriteGotoOutputPinIfNecessary(node.FalsePin, node.InputExecPins[0]);
    }
    else
    {
        builder.AppendLine("return;");
    }

    builder.AppendLine("}");
}
```

Рис. 2.10. Лістинг блоку if else

2.5. формалізація циклу for

Цикл for складається з початку, умови кроку та тіла циклу. Блок-схема алгоритму зображена на рис. 2.11

Рис. 2.11. Блок-схема циклу

У платформі цикл for складається з:

1) входу даних;
2) оператори кроку;
3) тіло циклу.

Прототип циклу зображений на рис. 2.12

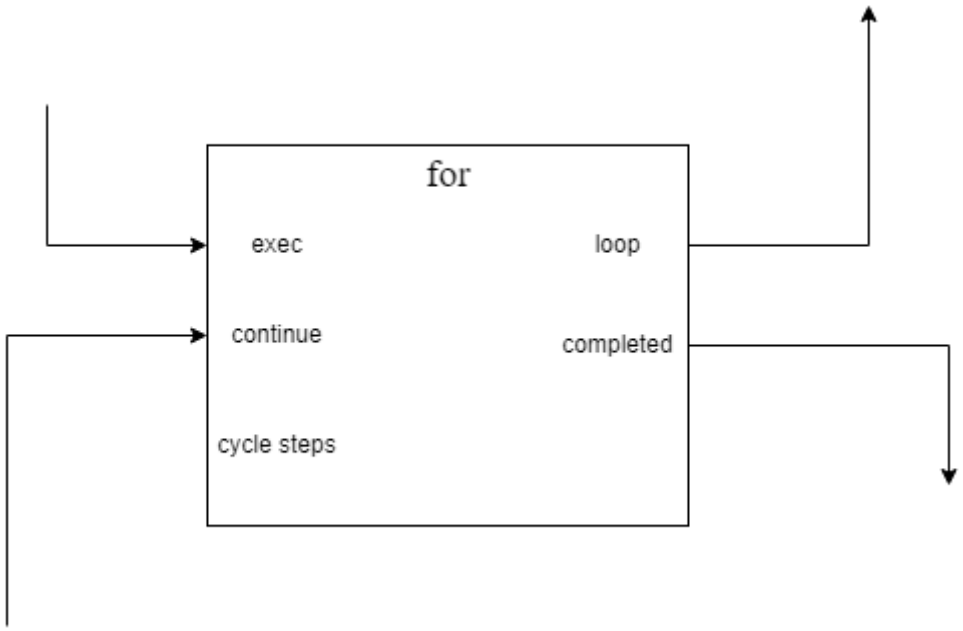

Рис. 2.12. Прототип циклу

Для коректної роботи блоку циклу for необхідно розробити алгоритм (рис. 2.13)

```
public void TranslateContinueForLoopNode(ForLoopNode node)
{
    // Translate all the pure nodes this node depends on in
    // the correct order
    TranslateDependentPureNodes(node);

    builder.AppendLine($"{GetOrCreatePinName(node.IndexPin)}++;");
    builder.AppendLine($"if ({GetOrCreatePinName(node.IndexPin)} < {GetPinIncomingValue(node.MaxIndexPin)})");
    builder.AppendLine("{");
    WritePushJumpStack(node.ContinuePin);
    WriteGotoOutputPinIfNecessary(node.LoopPin, node.ContinuePin);
    builder.AppendLine("}");

    WriteGotoOutputPinIfNecessary(node.CompletedPin, node.ContinuePin);
}
```

Рис. 2.13 Лістинг роботи циклу for

Однією з найбільш значущих переваг VPL є те, що він скорочує тривалість навчання та швидко перетворює початківців користувачів мови на досвідчених новачків. Ще одна перевага VPL полягає в тому, що він дає користувачеві позитивний відгук, приймаючи надмірність і зменшуючи збої, що може викликати позитивні емоції, які, у свою чергу, можуть підвищити пізнання.

Одна з найбільших слабких сторін VPL полягає в тому, що що межі мови швидко досягаються і що їх складно або неможливо обійти.

Незважаючи на те, що більша частина взаємодії з інтерфейсом була зручною та легко зрозумілою, користувачі все одно повинні були зрозуміти, як вони знайшли код і як його слід розмістити. Вони також повинні були зрозуміти деяку логіку програмування, наприклад, як комп’ютер аналізував код, щоб він працював і особливо усував неполадки. Вони також повинні були зрозуміти інші види знань, щоб працювати з програмою, наприклад математику. Той факт, що можна створити програму без плану, а також створити багато зайвого коду без збою програми, також є недоліком під час вивчення програмування, оскільки більш просунуті мови вимагають плану, структурованого коду та відсутність «вільних кінців» у код.

2.6. Висновки до розділу

На основі проведеного аналізу отримано наступні результати:

1) На основні характеристик об’єктів проведено математичне представлення їхніх властивостей, що дало змогу розробити протототи головних та обовязкових елементів платформи;
2) Розроблено алгоритми роботи циклу for та оператора if else;
3) Розглянуто особливості навчання користувачів. А саме етапи набуття навичок, стилі навикок;
4) Проаналізовані галузі візуалізації та візуального програмування.

РОЗДІЛ 3
РОЗРОБКА ТА ТЕСТУВАННЯ СЕРЕДОВИЩА

3.1. Визначення вимог

Візуальні мови складаються з набору графічних символів, які складаються у «візуальні речення із заданим синтаксисом і семантикою». Візуальні речення потім повинні бути просторово проаналізовані, щоб визначити основну синтаксичну структуру. Потім необхідно провести семантичний аналіз, щоб визначити значення візуальних речень (просторова інтерпретація). Синтаксичний та семантичний аналіз візуальної мови мало чим відрізняється від традиційного мовного підходу. Обидва типи мов необхідно проаналізувати, щоб визначити синтаксис і значення, істотна відмінність полягає в тому, що візуальні мови використовують графічні символи, а не текстові вирази традиційних мов.

Функціональна структура даного програмного середовища зображено на рис. 3.1.

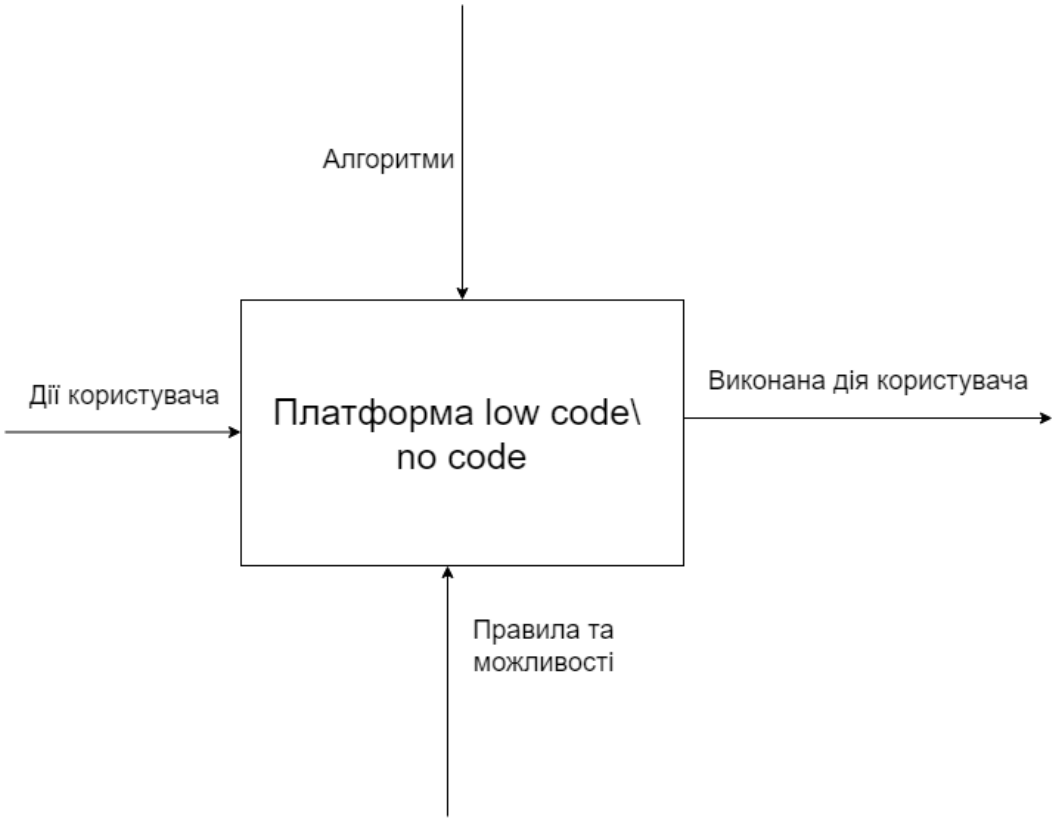

Рис. 3.1. Контекстна діаграма ІПП

Основні функції програми – це створення нових програмних продуктів. На основі вхідних даних та алгоритмів середовище створює програмний код та виконує дії задані користувачем. Для створення робочої програм користувач повинен коректно створити та зєднати всі блоки для виконання програми середовищем.

Даний програмний продукт повинен підтримувати наступні операційні системи: ПК з використанням операційних систем: Windows, Linux, MacOS.

Високих вимог до апаратної частини немає, тому зможе працювати на ЕОМ з мінімальними комплектаціями.

В будь-якому середовищі важливий графічний інтерфейс користувача. Також важливо передбачити адаптивність графічної оболонки. Оскільки користувачі будуть використовувати даний продукт на різному розширенні дисплеїв та різних операційних системах. А якщо інтерфейс не буде

розміщуватись правильно- користувачам не буде подобатись працювати в даному середовищі.

Після визначення функціоналу було розроблено макет користувацького інтерфейсу для головного меню (рис. 3.2).

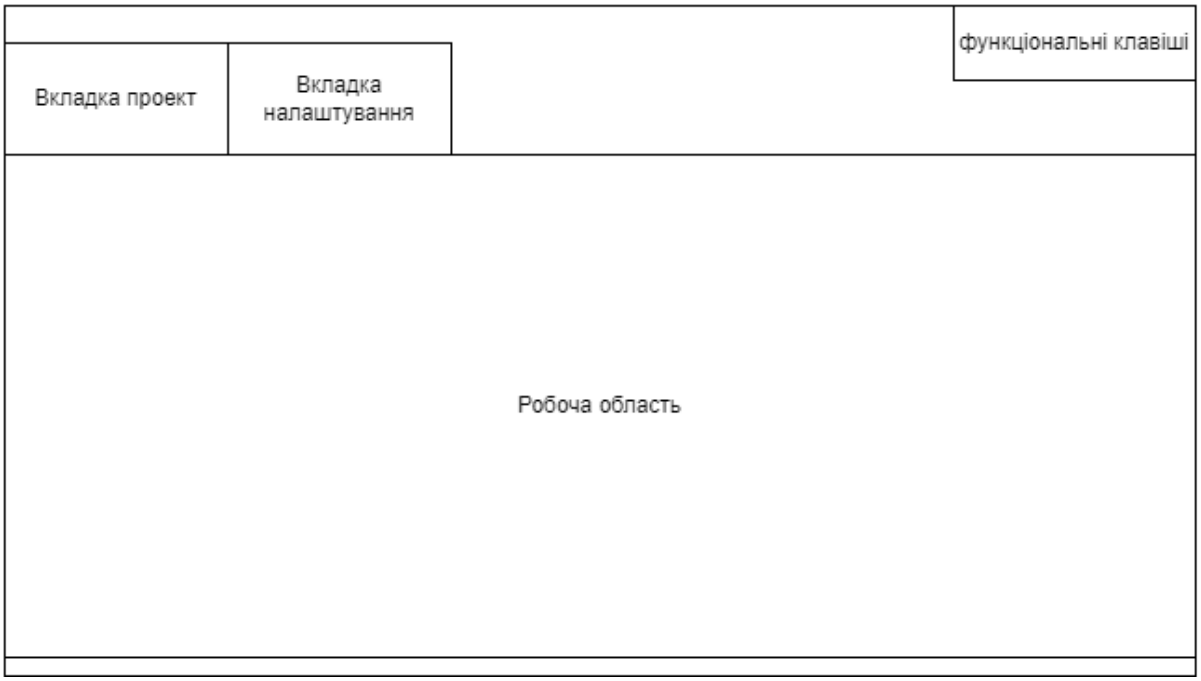

Рис 3.2. Макет головного меню

Після того як користувач обере проект – відбудеться завантаження класів проекту(рис 3.3)

Рис. 3.3. Макет класів в головному меню

Після того, як користувач вибрав конкретний клас, інтерфейс повинен оновитись (рис. 3.4).

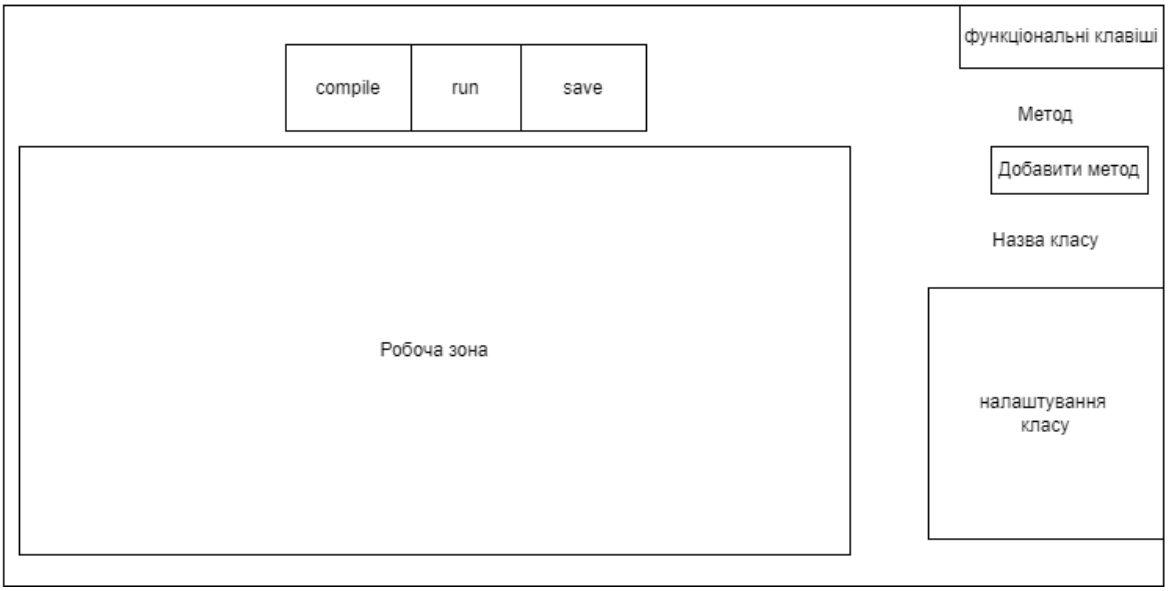

Рис. 3.4. Макети меню класа

3.2. Проектування архітектури

Для реалізації та якісної розробки середовище, необхідно правильно і грамотно спроектувати структуру алгоритмів роботи з програмним продуктом.

Вхідними даними є:

1) вхідні дані отримані від дій користувача;
2) дані отримані від попередньої ітерації обробки.

Функції програмного продукту, відносно задач можна розділити на такі категорії:

- алгоритм обробки вхідних даних користувача;
- допоміжні алгоритми;

Також на рис. 3.5. наведено діаграму прецедентів.

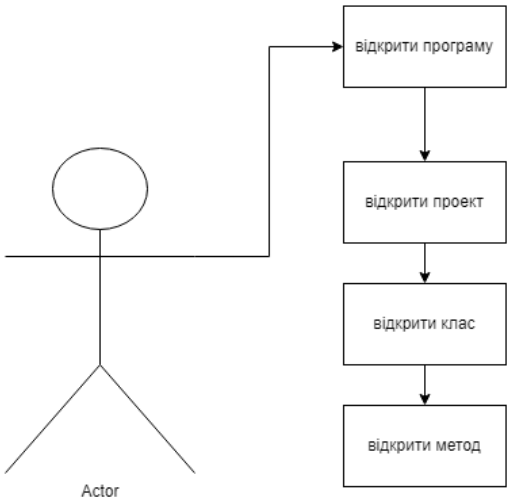

Рис. 3.5. Діаграма прецедентів

3.3. Тестування мульти-інтерфейсного середовища

Тестування програмного забезпечення — це процес збору інформації, дослідження для того, щоб можна було зробити висновок, що наявний

функціонал відповідає тому, що був закладений замовником і його якість відповідає встановленому рівню. Даний процес також виконує функцію виявлення помилок або дефектів, які виникають в процесі роботи програмного продукту.

Мета тестування – встановлення будь-яких невідповідностей в роботі програмного продукту відносно встановлених вимог.

Тестуваггя програмного забезпечення – це важливий крок у розробці продукту. Завдяки якому можна знайти та вирішити багато помилок. Також зрозуміти наскільки комфортний у використані програмний продукт та чи якісно він зроблений.

Розпочнемо тестування середовища розробки. Для цього необхідно запустити систему. Головне меню програми зображене на рис. 3.6.

Рис. 3.6. Головне меню програми

Після запуску програми необхідно обрати проект. Його можливо створити або вже запустити наявний(рис. 3.7.) а також зберегти. Вкладка зберегти стає активною коли клас вже обраний.

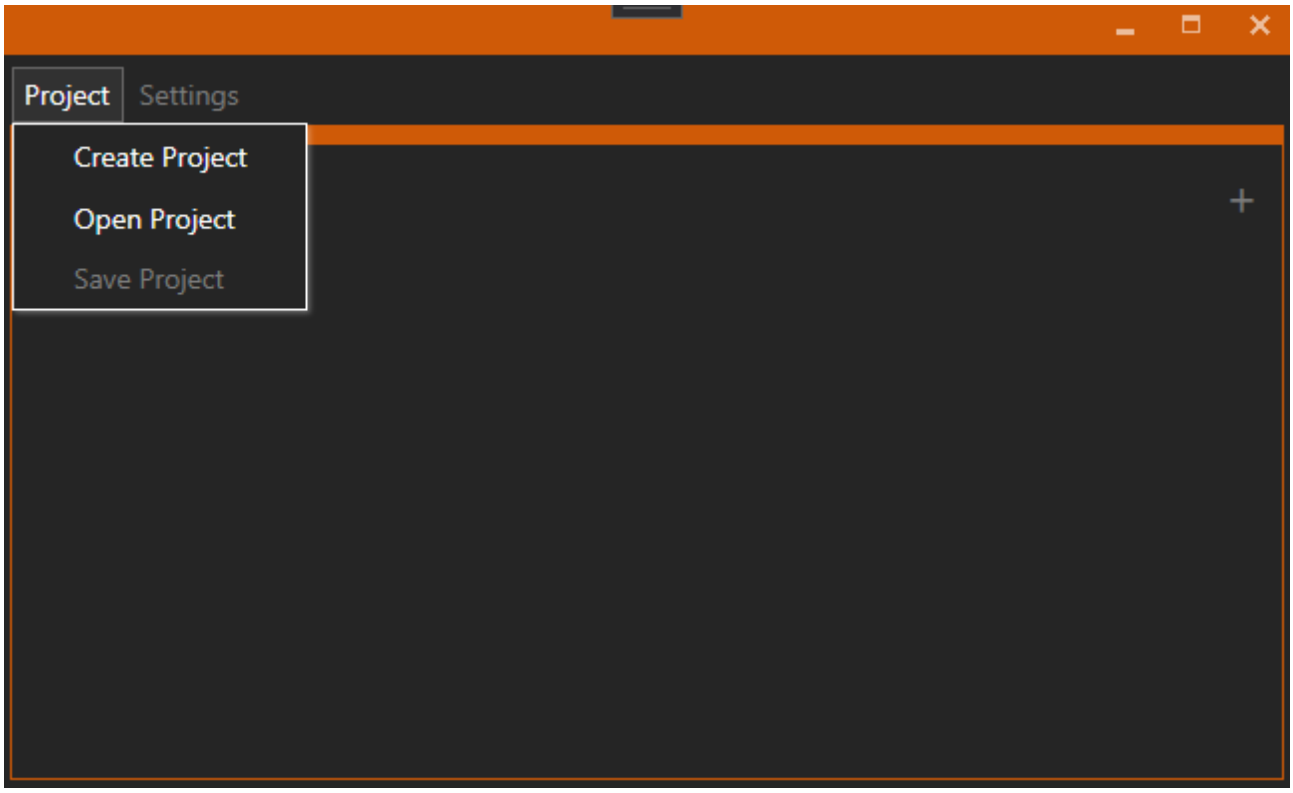

Рис. 3.7. Меню для роботи з проектом

Якщо проект вже створений потрібно обрати його та завантажити(рис. 3. 8., рис. 3.9)

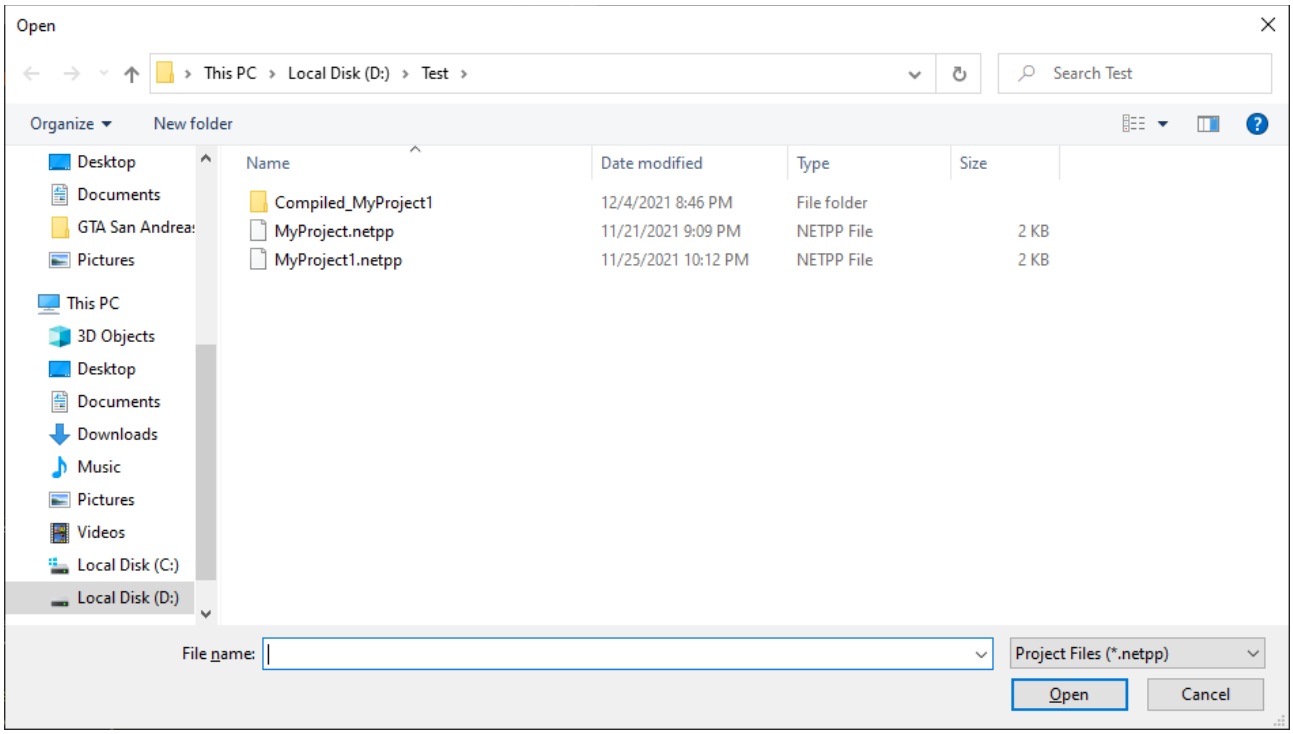

Рис. 3.8. Меню відкритя проекту

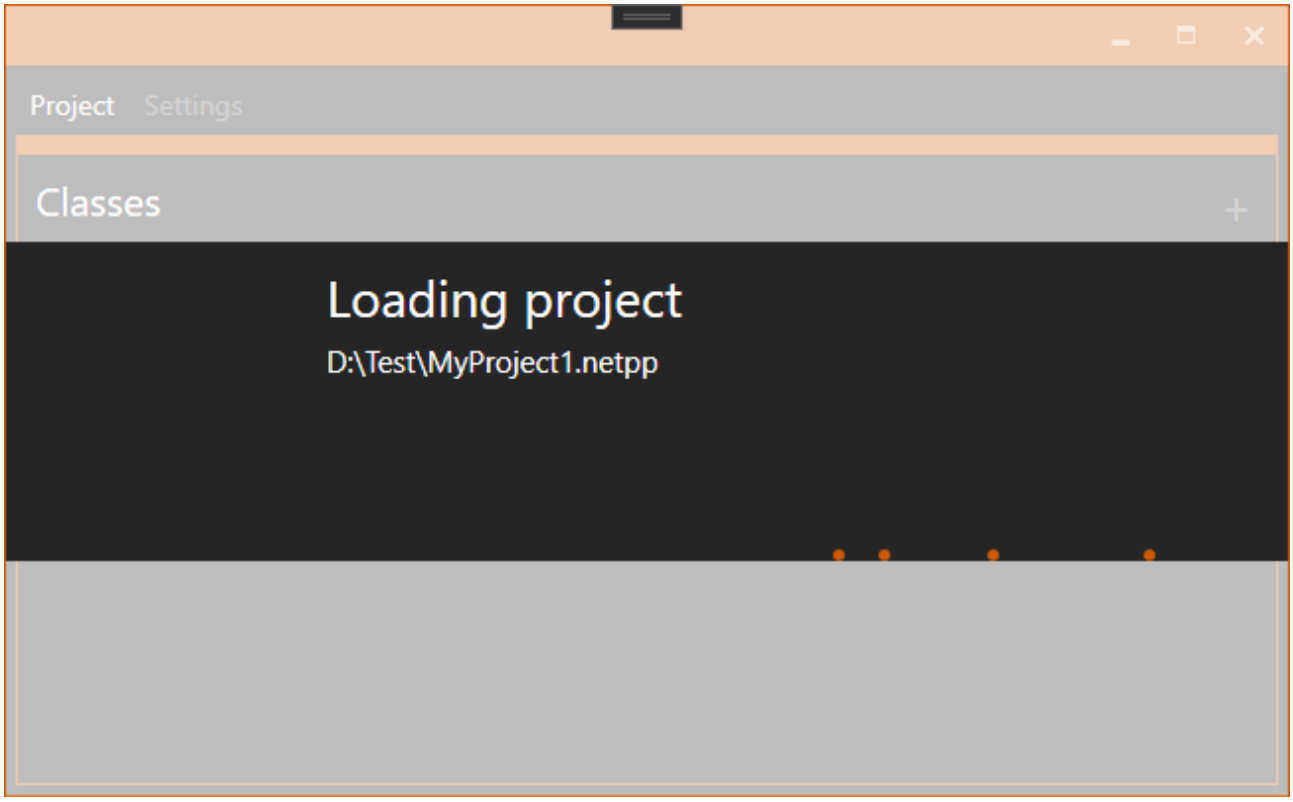

Рис. 3.9. Завантаженння проекту

Вигляд додатку після завантаження проекту з класом MyClass зображено на рис. 3.10.

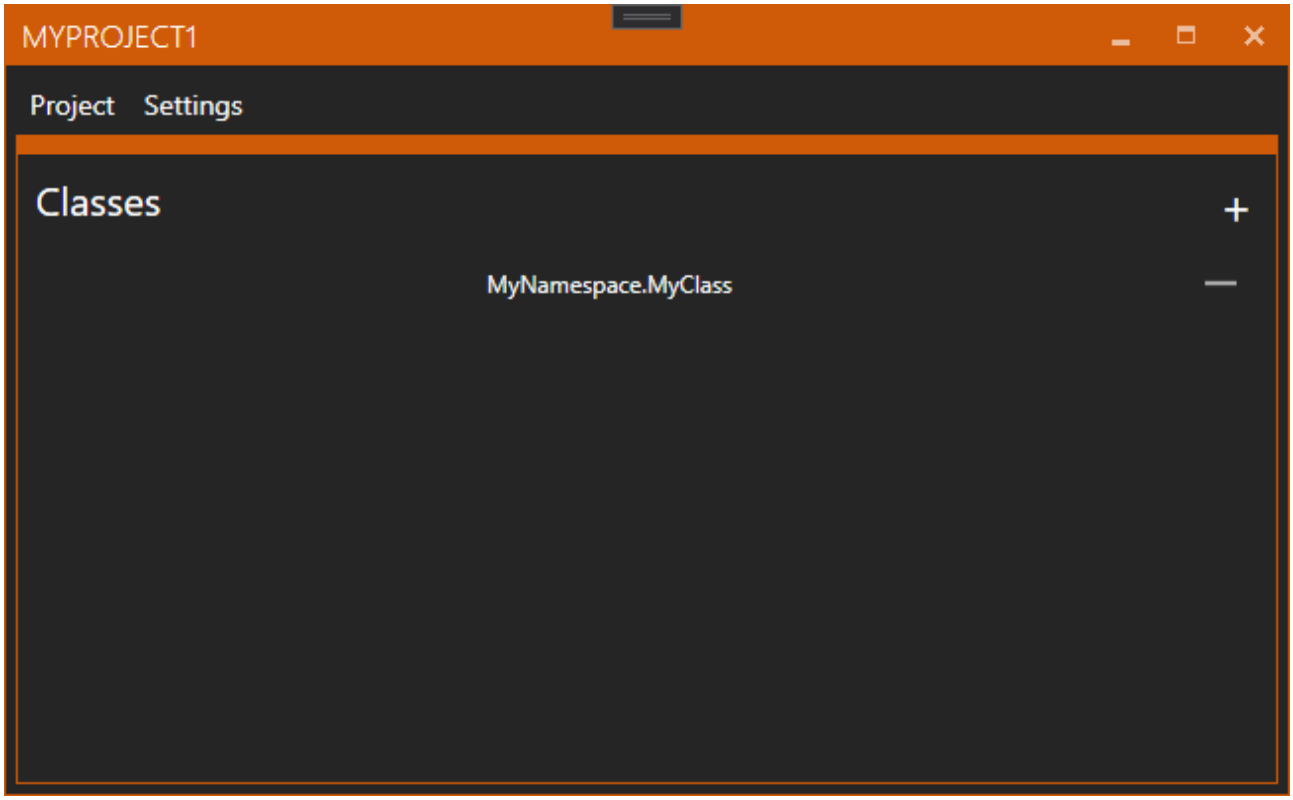

Рис. 3.10. Завантажений проект

Після завантаження проекту стає можливим створити новий клас або добавити вже існуючий(рис. 3.11.)

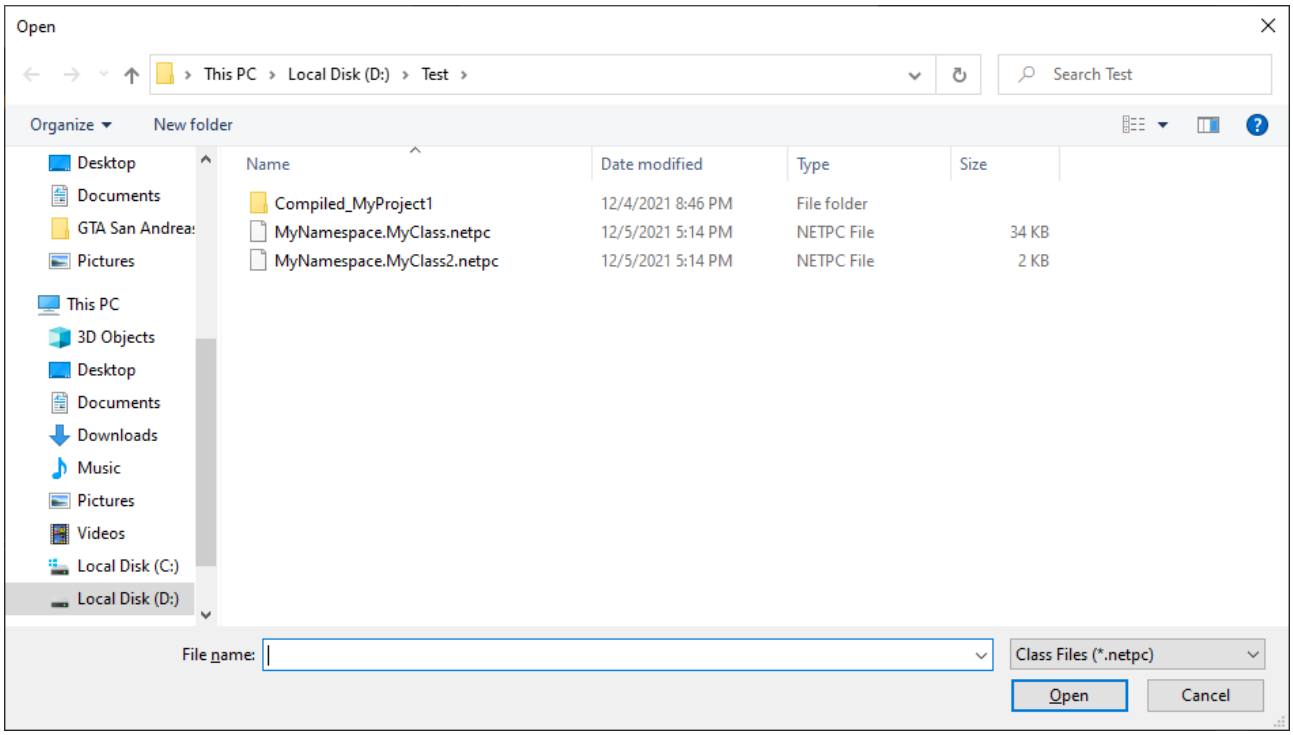

Рис. 3.11. Добавлення вже існуючого класу

Результат додавання класу зображений на рис. 3.12

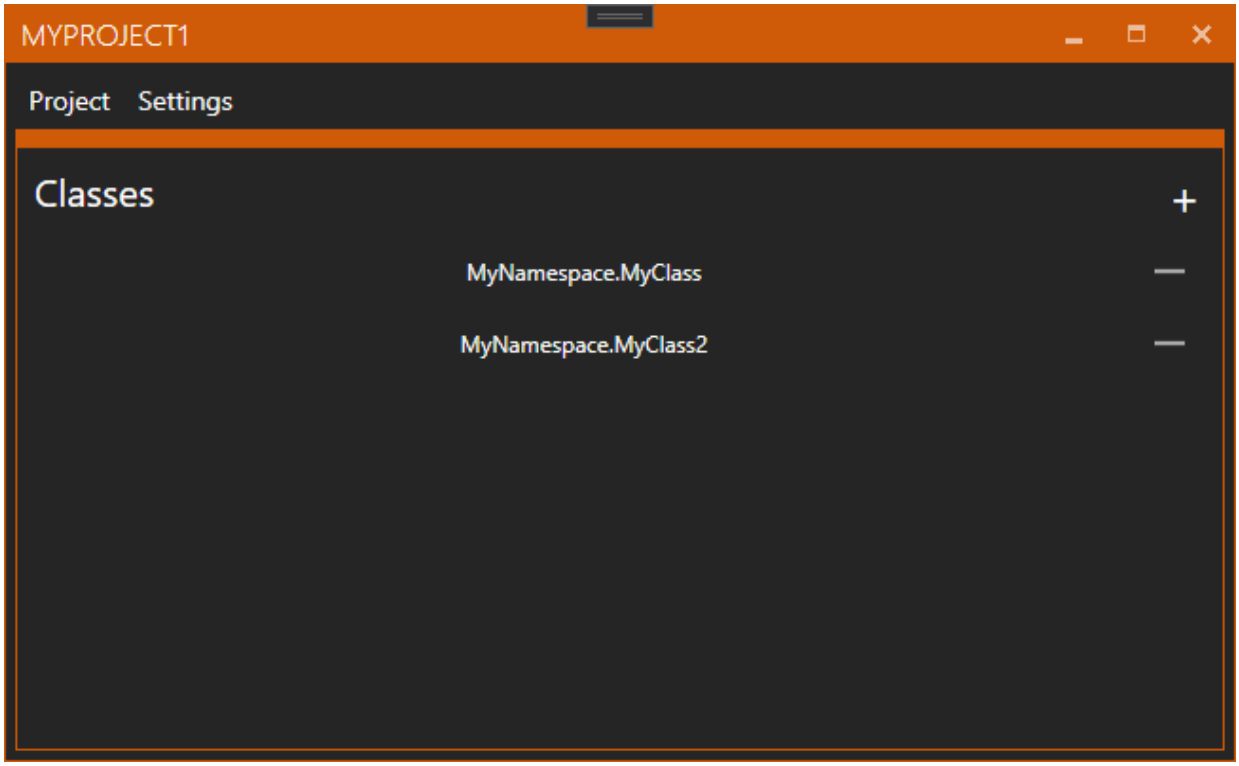

Рис. 3.12. Результат додавання класу

Також проект має свої налаштування які можна переглянути на рис. 3.13

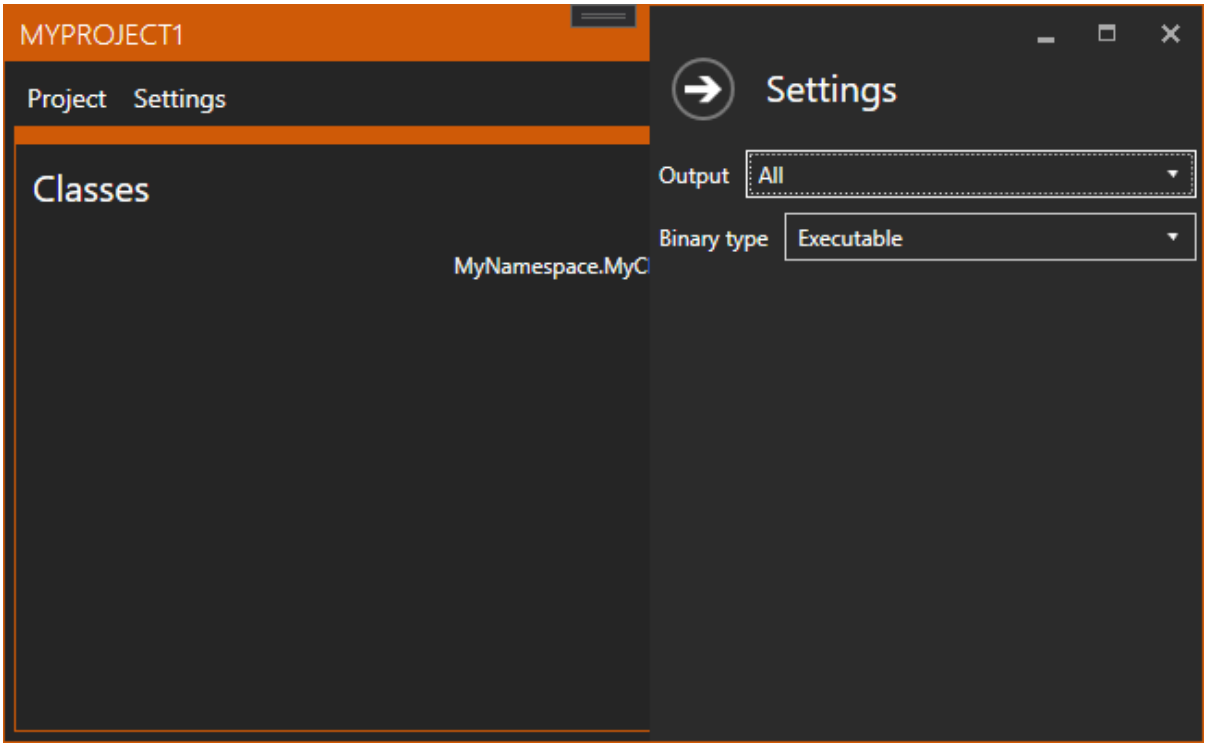

Рис. 3.13. Екран налаштувань проекту

Переходимо в меню класу(рис. 3.14.)

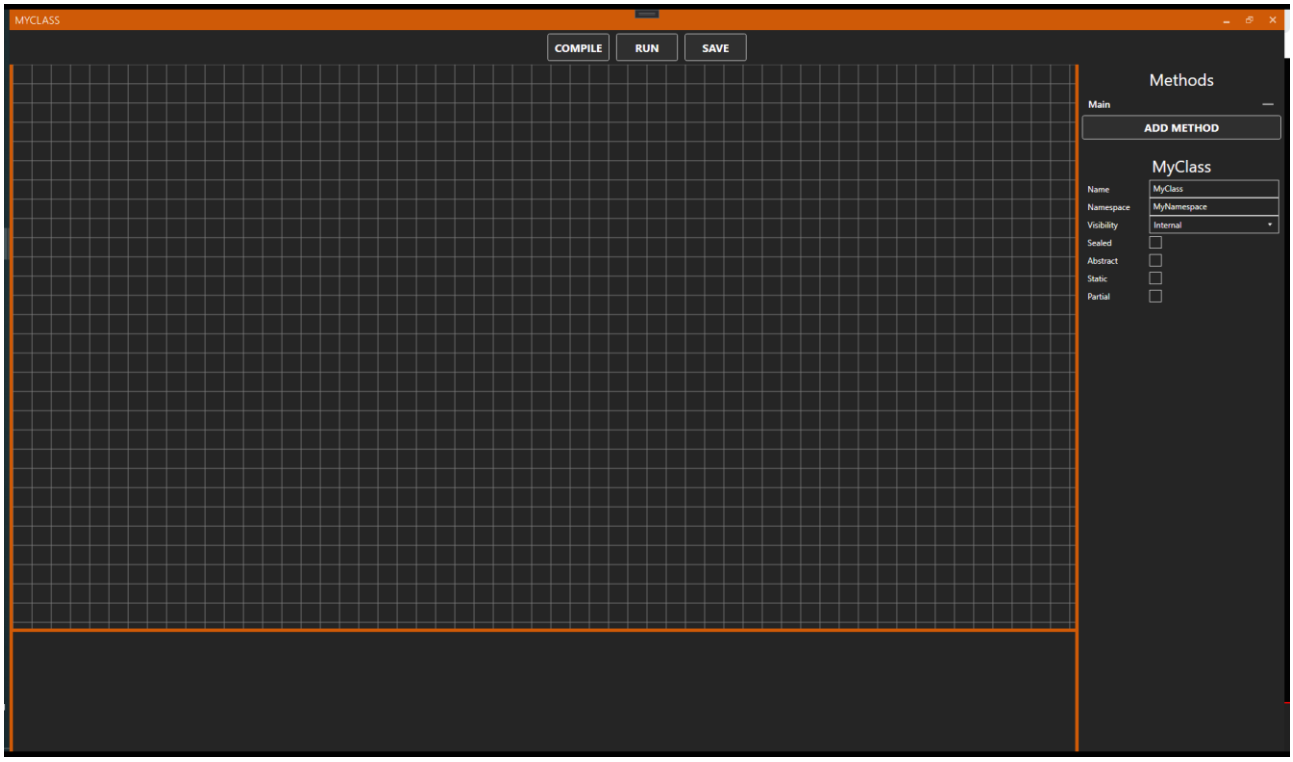

Рис. 3.14. Меню класу

В меню класу відбуваються основні дії. В ньому можна завантажити створені раніше або створити нові методи. Завантажимо вже створений метод main (рис. 3.15)

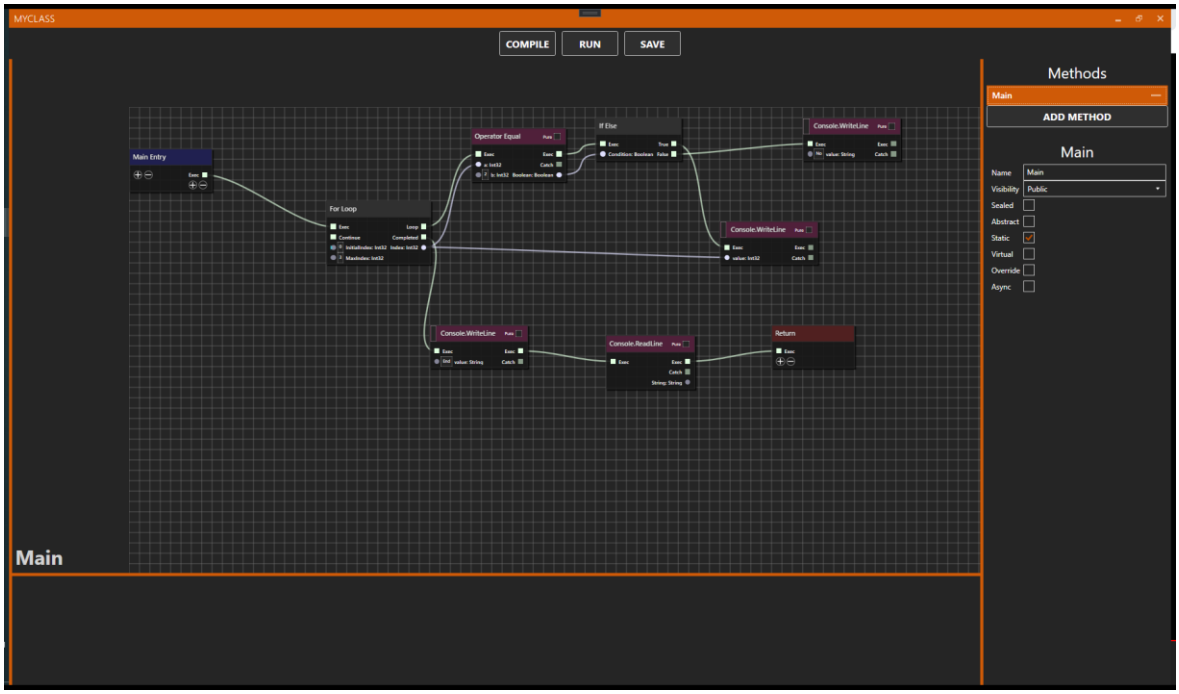

Рис. 3.15. Меню метода main

У кожного метода можна створити свої налаштування(рис. 3.16)

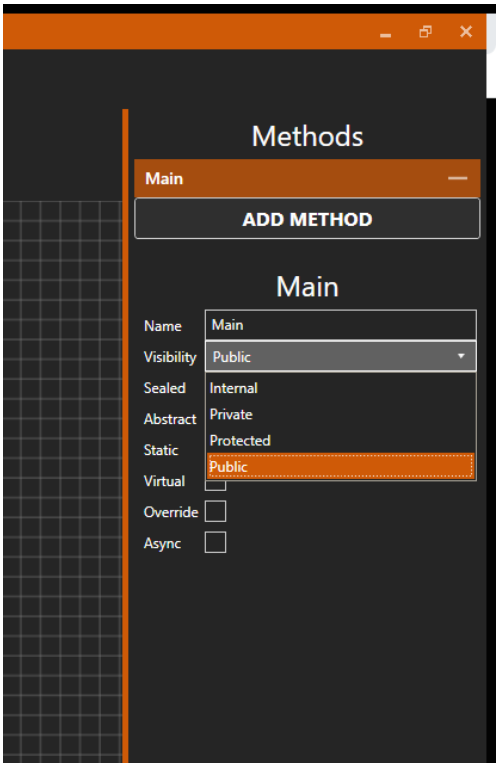

Рис. 3.16. Модифікатори класу

Щоб добавити оператор потрібно натиснути на праву клавішу мишки та вписати потрібний(рис. 3.17)

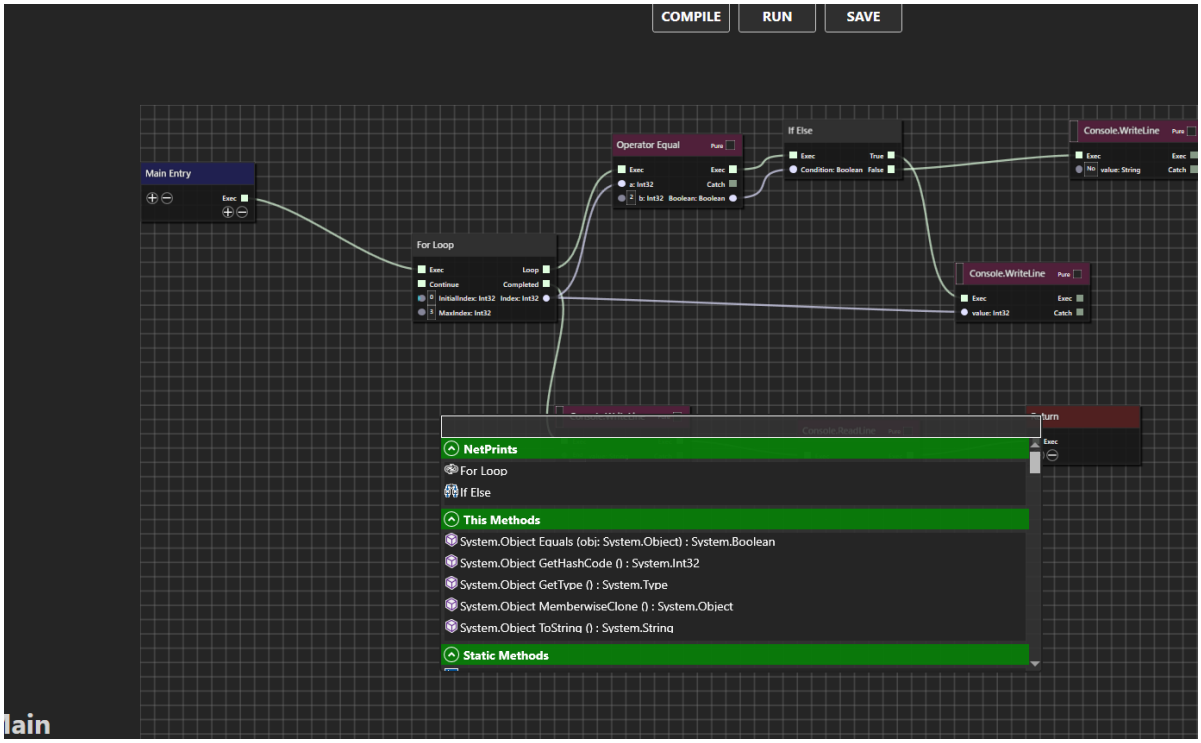

Рис. 3.17. Меню вибору оператора

Переглянемо загальний вигляд операторів арифметичних операцій(рис. 3.18)

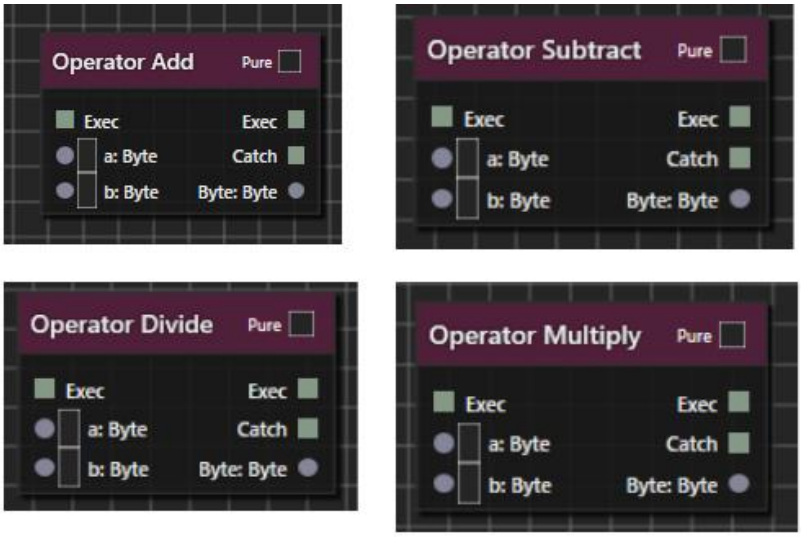

Рис. 3.18. Загальний вигляд операторів арифметичних операцій

Також переглянемо загальний вигляд оператора if та операторів порівняння(рис. 3.19)

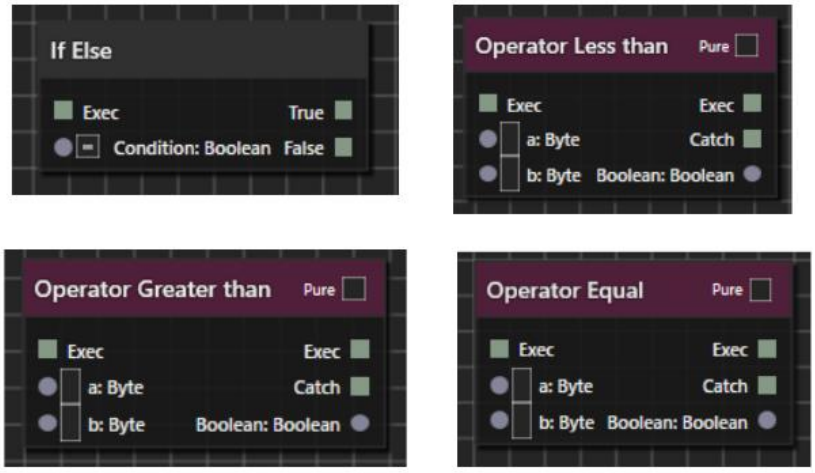

Рис. 3.19. Загальний вигляд оператора if та операторів порівняння

Також важливо переглянути загальний вигляд циклу for(рис. 3.20)

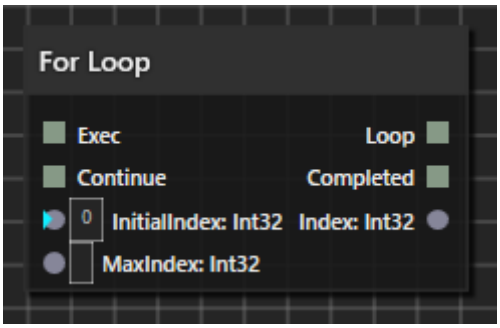

Рис. 3.20. Загальний вигляд циклу for

Перед запуском проекту потрібно його скомпілювати, щоб перевірити чи немає помилок. Успіша компіляція проекту зображена на рис. 3.21

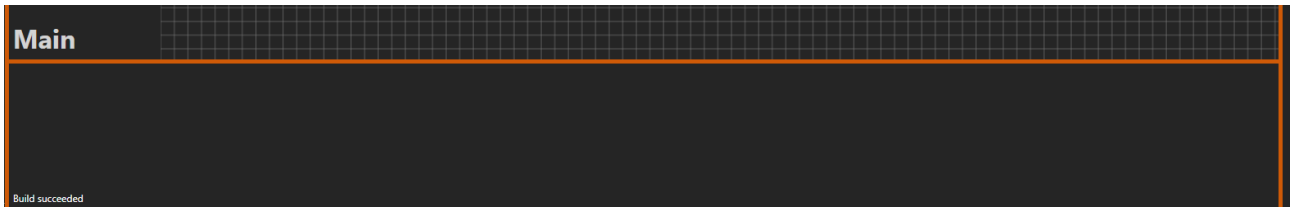

Рис. 3.21. Успішно скомпільований проект

При запуску проекту з помилками – проект не скомпліється і виведуться повідомлення про помилки(рис. 3.22)

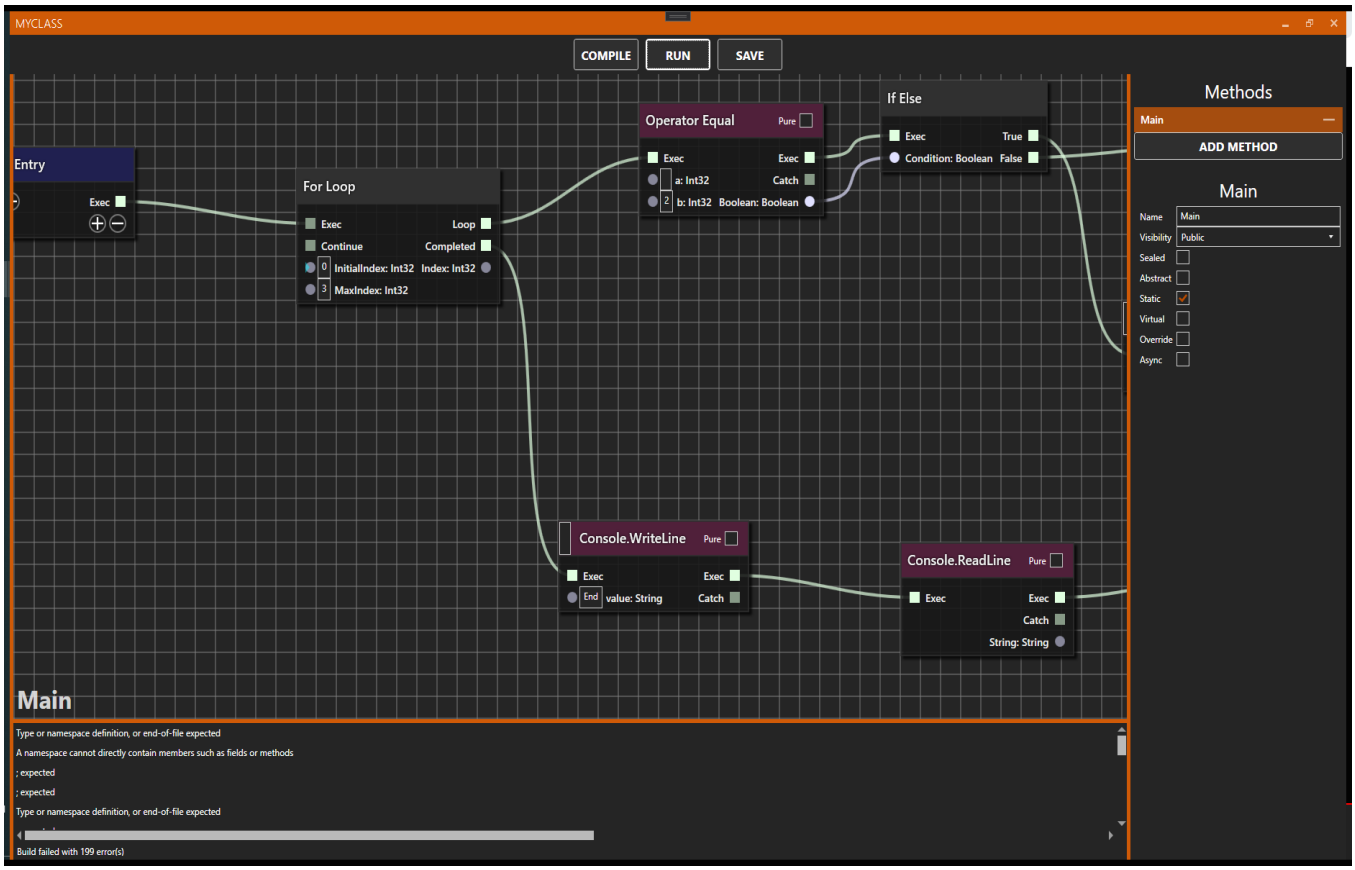

Рис. 3.22. Проект з помилками

Час запустити програму яка зображена на рис. 3.15. Результат виконання зображений на рис. 3.23

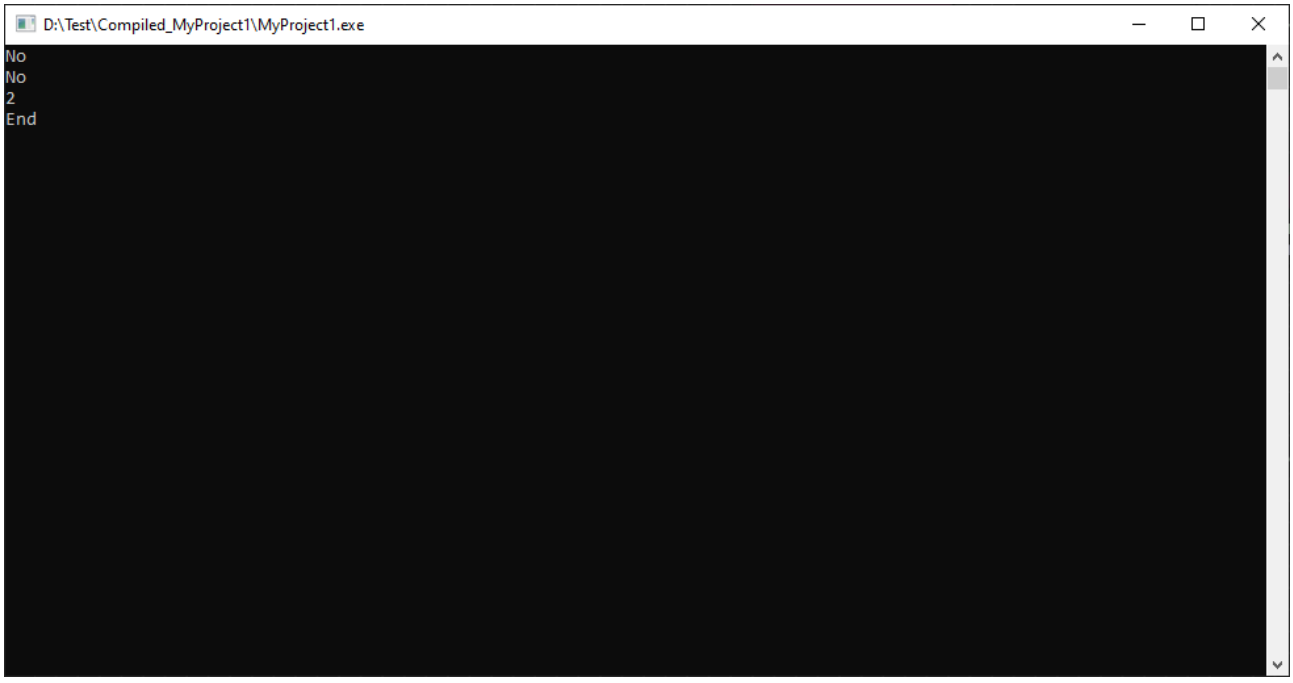

Рис. 3.23 Результат виконання проекту

Тестування середовища успішно завершено, помилок у пррограмі немає. Збоїв не зафіксовано.

Отже, дану система можна вважати такою яка не вимагає допрацювання та працює відповідно поставленій задачі. Ввсі етапи тестування успішно завершенні, помилок у виконанні не виникло.

Переваги даного продукту:

1) простий і зрозумілий інтерфейс;
2) можливість створити свій власний продукт;
3) стабільна робота додатку;
4) можливість швидкого створення програнмих додатків;
5) швидке навчання для роботи в середовищі.

3.4. Висновки до розділу

На основі проведеного аналізу отримано наступні результати:

1) визначено основні вимоги до середовища, за допомогою контексної діаграми і діаграми прецендентів;

2) створено макети інтерфейсів декількох важливих вкладок, на основі який реалізовано інтерфейси;

3) проведено ручне тестування програмного продукту.

РОЗДІЛ 4
ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1. Охорона праці

З розвитком науково-технічного прогресу важливу роль відіграє можливість безпечного виконання людьми своїх трудових обов'язків. У зв'язку з цим була створена і розвивається наука про безпеку праці.

Охорона праці - система законодавчих актів, соціально-економічних, організаційних, технічних, гігієнічних, лікувально профілактичних заходів, направлених на: забезпечення безпеки людини, збереження здоров'я та працездатності людини в процесі праці; розробку методів і засобів захисту шляхом зниження впливу шкідливих і небезпечних чинників до допустимих значень. Завдання ОП - звести до мінімуму вірогідність травмування або захворювання працюючого з одночасним забезпеченням комфорту для максимальної продуктивності праці [14].

Беручи до уваги можливі небезпеки під час розробки та подальшого використання платформ “Low code/No code” в комп’ютерних системах необхідно дотримуватися всіх вимог з охорони праці та техніки безпеки.

На робочому місці яке обладнане персональним комп’ютером потрібно дотримуватися «Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями», які затверджені наказом Міністерство соціальної політики України від 14.02.2018 за № 207. А саме:

— Робочі місця працівників з екранними пристроями мають бути спроектовані так і мати такі розміри, щоб працівники мали простір для зміни робочого положення та рухів.

— Для забезпечення безпеки та захисту здоров’я працівників усе випромінювання від екранних пристроїв має бути зведене до гранично допустимого рівня (вплив на людину факторів довкілля - шуму, вібрації, забруднювачів, температури тощо, який не спричиняє соматичних або психічних

розладів, а також змін стану здоров'я, працездатності, поведінки, що виходять за межі пристосувальних реакцій) з погляду безпеки та охорони здоров'я працівників.

— Організація робочого місця працівника з екранними пристроями має забезпечувати відповідність усіх елементів робочого місця та їх розташування ергономічним, антропологічним, психофізіологічним вимогам, а також характеру виконуваних робіт.

— Освітлення робочого місця працівника з екранними пристроями має створювати відповідний контраст між екраном і навколишнім середовищем (з урахуванням виду роботи) та відповідати вимогам ДСанПІН 3.3.2.007-98.

— Мікроклімат виробничих приміщень з робочими місцями працівників з екранними пристроями має підтримуватись на постійному рівні та відповідати вимогам Санітарних норм мікроклімату виробничих приміщень ДСН 3.3.6.042-99, затверджених постановою Головного державного санітарного лікаря України від 01 грудня 1999 року № 42 (далі - ДСН 3.3.6.042-99).

— Робочий стіл або робоча поверхня повинні бути достатнього розміру та мати поверхню з низькою відбивною здатністю, допускати гнучкість під час розміщення екрана, клавіатури, документів і відповідного устаткування.

— Робоче крісло має бути стійким і дозволяти працівнику з екранними пристроями легко рухатися та займати зручне положення. Сидіння має регулюватися по висоті, спинка сидіння - як по висоті, так і по нахилу. Слід передбачати підніжку для тих, кому це необхідно для зручності.

Звернемо увагу на основні санітарно-гігієнічні вимоги до умов праці в офісних приміщеннях де буде використовуватись платформа "Low code/No code" в комп'ютерних системах:

— площа приміщення повинна бути не менше 6,0 м² на 1 робоче місце; робочі місця повинні бути розташовані на відстані не менше ніж 1 м від стіни з вікном, і 1,4 м від звичайної стіни; відстань між бічними поверхнями комп'ютерів має бути не меншою за 1,2 м; відстань між тильною поверхнею одного комп'ютера та екраном іншого не повинна бути меншою 2,5м;

— відповідні робочі місця заборонено облаштовувати у підвальних або цокольних приміщеннях будинків. В обладнанні приміщень забороняється використання полімерних матеріалів (деревинно-стружкові плити, шпалери, що миються, рулонні синтетичні матеріали, шаруватий паперовий пластик тощо), що виділяють у повітря шкідливі хімічні речовини. Покриття підлоги повинно бути матовим, а поверхня – рівною, неслизькою, з антистатичними властивостями;

— особливу увагу необхідно приділити колірній гармонії офісних приміщень. Колір є засобом створення психологічного комфорту та підвищення продуктивності праці. Найбільш сприятливі для нервової системи світлі, пастельні тони – зеленувато-блакитний, ясно-сірий, золотавий. Яскраві, контрастні поєднання (синій і жовтогарячий, червоний і фіолетовий) викликають втому, роздратування;

— у приміщеннях, де здійснюється робота з комп'ютерами, щодня має проводитися вологе прибирання з метою недопущення запиленості підлоги та меблів. Крім того, має бути обладнана кімната психологічного розвантаження;

— конструкція робочого столу та крісла користувача персонального комп'ютера має забезпечити підтримання оптимальної робочої пози та забезпечувати оптимальне розміщення на робочій поверхні використовуваного обладнання (дисплея, клавіатури, принтера) і документів;

— приміщення для роботи з персональними комп'ютерами мають бути обладнані системами опалення, кондиціонування повітря, або припливно-витяжною вентиляцією. У приміщеннях на робочих місцях мають забезпечуватись оптимальні значення параметрів мікроклімату: температура повітря повинна становити 22–25°С, відносна вологість повітря — 40–60%, швидкість руху повітря — не більше 0,1 м/с. При недотриманні вказаних показників мікроклімату в офісних приміщеннях робочий день для робітників повинен бути скорочений мінімум на 10%;

— досить важливим є вимоги до освітлення приміщень,оскільки відомо, що тривала робота за комп'ютером та з документами при недостатньому рівні освітленості може призвести до значного перенапруження зору. Природне освітлення має забезпечувати коефіцієнт природної освітленості (КПО) не нижче ніж 1,5%. Для регулювання рівня освітлення природним світлом бажано застосовувати жалюзі. Робоче місце, обладнане ПК повинно бути розташоване так, щоб уникнути попадання в очі прямого сонячного світла. Штучне освітлення приміщення має бути обладнане системою загального рівномірного освітлення. Застосування світильників без розсіювачів та екрануючих сіток забороняється. Рівень освітленості на робочому столі в зоні розташування документів має бути в межах 300–500 лк;

— в офісних приміщеннях нормуються також еквівалентні рівні звуку (для програмістів – 50 дБА, а для операторів в залах обробки інформації на ПК та операторів комп'ютерного набору – 65 дБА) ;

— вимоги щодо рівня неіонізуючих електромагнітних випромінювань, електростатичних і магнітних полів, а також інтенсивність потоків інфрачервоного та ультрафіолетового випромінювань встановлюються відповідно до ДСанПіН 3.3.2.007-98 і ДСанПіН 3.3.6.096-2002.

Отже, небезпечні фактори погіршують умови праці надаючи шкідливу дію на організм людини. Наприклад, ті хто працюють в умовах тривалої шумової дії відчувають дратівливість, головні болі, запаморочення, зниження пам'яті, підвищену стомлюваність, пониження апетиту, болі у вухах і т.д. Такі порушення в роботі ряду органів і систем організму людини можуть викликати негативні зміни в емоційному стані аж до стресових.

4.2. Організація оповіщення і зв'язку у надзвичайних ситуаціях техногенного та природного характеру.

Правовою основою організації оповіщення населення області при загрозі чи виникненні надзвичайних ситуацій (НС) є Конституція України, Кодекс

Цивільного захисту України, Постанови Кабінету Міністрів "Про затвердження Положення про організацію оповіщення про загрозу виникнення або виникнення надзвичайних ситуацій та зв'язку у сфері цивільного захисту", "Положення про єдину державну систему цивільного захисту", накази центрального органу виконавчої влади з питань НС, відповідні розпорядження обласної державної адміністрації та інші акти.

Одним із основних завдань Цивільного захисту України, як державної системи органів управління, сил і засобів, які створені для організації і забезпечення захисту населення від наслідків надзвичайних ситуацій техногенного, екологічного, природного та воєнного характеру, є оповіщення населення про загрозу і виникнення надзвичайних ситуацій у мирний і воєнний часи та постійне інформування його про наявну обстановку..

Система централізованого оповіщення області представляє собою комплекс організаційно-технічних заходів, апаратури і технічних засобів оповіщення, засобів та каналів зв'язку, мереж проводового, радіо, телевізійного мовлення призначених для своєчасного доведення сигналів та інформації з питань цивільної оборони (цивільного захисту) до центральних і місцевих органів виконавчої влади, підприємств, установ, організацій і населення. Для зосередження уваги громадян перед передачею інформації вмикаються сирени, інші сигнальні засоби. Їх звук означає попереджувальний сигнал "УВАГА ВСІМ".

Взагалі система оповіщення складається із загальнодержавної, регіональних і спеціальних систем централізованого оповіщення; локальних та об'єктових систем оповіщення, систем циркулярного виклику. Ці системи забезпечують оповіщення і подальше інформування:

- чергових служб міністерств та інших центральних органів виконавчої влади по службових телефонах;
- чергових служб місцевих органів виконавчої влади;
- чергових аварійно-рятувальних служб.

Для виконання основних завдань оповіщення, які визначені керівними документами, а саме: забезпечення своєчасного проходження інформації між органами управління щодо ступенів готовності; оповіщення керівного складу, населення про загрозу радіоактивного, хімічного і бактеріологічного ураження, про загрозу і виникнення надзвичайних ситуацій у мирний і особливий період та постійне інформування його про наявну обстановку.

Система оповіщення працює за принципом відбору каналів з єдиної національної системи зв'язку. Апаратура оповіщення розташована на відповідних об'єктах органів управління, електрозв'язку, чергових відділах МВС, на радіо-теле-передавальних центрах та інших визначених підприємствах і установах.

Для оперативного доведення відповідної інформації до керівного складу по телефонам застосовуються стійки циркулярного виклику та апаратура автоматизованого багатоканального оповіщення.

Для передачі попереджувального сигналу "УВАГА ВСІМ" застосовуються електричні сирени централізованого і автономного включення, наявна кількість яких в основному забезпечує озвучення території де проживає населення області.

Інформація до населення доводиться через радіотрансляційні вузли, радіо-теле-передавальні центри по проводовому мовленню до якого підключено радіоточки і вуличні гучномовці, по визначеним радіо та телевізійним каналам.

На випадок виникнення надзвичайної ситуації безпосередньо на потенційно небезпечних підприємствах за їх рахунок створюються об'єктові системи оповіщення.

Локальні системи оповіщення створюються на потенційно небезпечних об'єктах, зона ураження від яких, у разі виникнення на них надзвичайної ситуації, досягає заселених територій або інших підприємств, установ, організацій. До їх складу входять абонентські радіоточки мережі радіомовлення та відомчих радіотрансляційних вузлів, вуличні гучномовці, пристрої запуску електросирен та самі електросирени, система централізованого виклику, магнітофони, магнітні стрічки із записаними текстами звернень.

Готовність систем оповіщення забезпечено шляхом:

− організація цілодовового чергування відповідних служб;

− налагодження телефонного зв'язку чергових служб потенційно небезпечних підприємств, зона ураження яких може поширюватися на заселені території або території інших підприємств, установ, організацій з оперативно-черговою службою пункту управління облдержадміністрації, чергових служб органів МВС в містах та районах області;

− завчасна підготовка персоналу чергових служб до дій у надзвичайних ситуаціях;

− впровадження автоматизованих систем оповіщення з використанням сучасних технологій;

− проведення якісного експлуатаційно-технічного обслуговування апаратури оповіщення та інших технічних засобів зв'язку та оповіщення.

Забороняється зняття та відключення телекомунікаційних мереж, абонентських ліній, через які здійснюється запуск електросирен, демонтувати вуличні гучномовці без погодження з відповідними органами управління з питань ЦЗН.

Висновки: у результаті проведеного аналізу організації оповіщення і зв’язку у надзвичайних ситуаціях техногенного та природного характеру виявлено та описано дії, які необхідно виконувати для оповіщення населення, необхідно дотримуватись визначених у нормативних документах правил. Також необхідно перевіряти готовність системи оповіщення: проводити підготовку персоналу, впроваджувати автоматизовані системи оповіщення, проводити експлуатаційно-технічне обслуговування.

ВИСНОВКИ

Основні результати полягають у наступному:

1. На основі аналізу електронних джерел проведено системний аналіз предметної області, у результаті якого визначено мету розробки системи, сформульовано та обґрунтовано задачі, функції системи та її структура, також окреслені вхідні та вихідні дані та визначені вимоги до продукту;

2. Визначено інструментальні засоби для вирішення поставлених задач, які зорієнтовані на використанням безкоштовних програмних засобів і дозволяють створювати просте у використанні та недороге програмне забезпечення, доступне широкому колу користувачів;

3. На основні фізичних характеристик об’єктів елементів візуального програмування проведено математичне представлення їхніх властивостей.

4. Запропоновано та формалізовано процес створення платформи Low Code/No Code на основі класичних підходів до проектування блок-схем, що дає змогу знизити поріг входу фахівців при прозробці комп’ютерних систем, а також збільшити кількість інженерів, які без спеціалізованої освіти можуть розробляти програмні продукти;

5. Набули подальшого розвитку методи і засоби розробки програмних додатків шляхом атоматизації процесів та спрощення роботи з візуальними інструментами, що дало змогу підвищити ефективність розробки комп’ютерних систем.

6. Проаналізовано існуючі операції представлені в мові програмування C#. Наведено приклади роботи простих різноманітних операції (арифметичних, порівння) в NC/LC додатках;

7. Спроектовано та реалізовано програмний продукт, який позиціонує себе, як додаток для розробки різноманітних програмних продуктів та інтеграції в існуючі або муйбутні продукти третіх компанії;

8. Створено макети інтерфейсу користувача на основі яких було реалізовано інтерфейс в програмному продукті та проведено ручне тестування програмного продукту.

9. Проведено аналіз вимог з охорони праці, а саме: робочих місць, приміщення в яких працюють співробітники. Також проаналізовано вимоги щодо організації оповіщення і зв’язку у надзвичайних ситуаціях техногенного та природного характеру. Проведено аналіз необхідності перевірки готовінсть системи оповіщення.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. No-code, zero-code, low-code программирование и сервисы. URL: https://intellect.icu/no-code-zero-code-low-code-programmirovanie-i-servisy-8181 (дата звернення: 05.10.2021).

2. Що таке low-code платформа. URL: https://www.terrasoft.ua/page/low-code (дата звернення: 06.10.2021).

3. Low-Code and No-Code: What's the Difference and When to Use What. URL:https://www.outsystems.com/blog/posts/low-code-vs-no-code/(дата звернення: 07.10.2021).

4. Методология Kanban: введение. URL: https://habr.com/ru/post/230725/ (дата звернення: 08.10.2021).

5. The Definitive Guide to Low-Code Development. URL: https://www.mendix.com/low-code-guide/ (дата звернення: 16.10.2021).

6. What is love code?. URL: https://www.creatio.com/page/low-code (дата звернення: 20.10.2021).

7. Low Code vs No Code Explained. URL: https://www.bmc.com/blogs/low-code-vs-no-code/# (дата звернення: 30.10.2021).

8. What is low-code and no-code? A guide to development platforms. URL: https://www.zdnet.com/article/special-report-what-is-low-code-no-code-a-guide-to-development-platforms/ (дата звернення: 01.11.2021).

9. What is visual language and how to create one for your business. URL https://rockcontent.com/blog/visual-language/ (дата звернення: 01.11.2021).

10. How to Build and Maintain a Visual Language. URL: https://www.shopify.com/partners/blog/visual-language(датазвернення: 05.11.2021).

11. Структурное/визуальное программирование. URL: https://habr.com/ru/post/532914/ (дата звернення: 06.11.2021).

12. Windows Presentation Foundation. URL: https://uk.wikipedia.org/wiki/Windows_Presentation_Foundation (дата звернення: 07.11.2021).

13. Почему визуальное программирование и D3NE могут быть Вам полезны. URL: https://habr.com/ru/post/341690/ (дата звернення: 08.11.2021).

14. Закон України «Про охорону праці». URL: http://zakon3.rada.gov.ua/laws/show/2694-12 (дата звернення: 15.11.2021).

15. Наказ України «Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями». URL: https://zakon.rada.gov.ua/laws/show/z0508-18#Text (дата звернення: 15.11.2021).

16. Геврик Є. О. Охорона праці: Навч. посіб. для студ. вищ. навч. закл.. 2.вид., випр. та доп. –К. : Ніка-Центр, 2005. 294с

17. Яцишин В.В., Яворська Х.В. Аналіз особливостей візуальних мов програмування. Матеріали X Міжнародної науково-практичної конференції молодих учених та студентів "Актуальні задачі сучасних технологій" (24-25 листопада 2021р.).Тернопільського національного технічного університету імені Івана Пулюя. Тернопіль: Тнту. 2021.С. 146

18. Яцишин В.В., Яворська Х.В. Відміності low-code/no-code розробки. Матеріали IX науково-технічної конференції "Інформаційні моделі, системи та технології" (8-9 грудня 2021р.).Тернопільського національного технічного університету імені Івана Пулюя. Тернопіль: Тнту. 2021.С. 144

Додаток А

Текст наукових публікацій дипломної роботи магістра

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Тернопільський національний технічний університет імені Івана Пулюя (Україна)
Університет імені П'єра і Марії Кюрі (Франція)
Маріборський університет (Словенія)
Технічний університет у Кошице (Словаччина)
Вільнюський технічний університет ім. Гедимінаса (Литва)
Білоруський національний технічний університет (Республіка Білорусь)
Міжнародний університет цивільної авіації (Марокко)
Наукове товариство ім. Т.Шевченка

АКТУАЛЬНІ ЗАДАЧІ СУЧАСНИХ ТЕХНОЛОГІЙ

Збірник
тез доповідей
Том І

X Міжнародної науково-практичної конференції молодих учених та студентів
24-25 листопада 2021 року

УКРАЇНА
ТЕРНОПІЛЬ – 2021

Матеріали X Міжнародної науково-практичної конференції молодих учених та студентів
«АКТУАЛЬНІ ЗАДАЧІ СУЧАСНИХ ТЕХНОЛОГІЙ» – Тернопіль 24-25 листопада 2021 року
УДК 004.4'236

В.В. Яцишин, канд. техн. наук, доцент, Х.В. Яворська
Тернопільський національний технічний університет імені Івана Пулюя

АНАЛІЗ ОСОБЛИВОСТЕЙ ВІЗУАЛЬНИХ МОВ ПРОГРАМУВАННЯ

V.V. Yatsyshyn PhD, Assoc. Prof., K.V. Yavorska

ANALYSIS OF VISUAL PROGRAMING LANGUAGES FEATURES

Сучасна епоха проектування та реалізації комп'ютерних інформаційних систем характеризується, у більшості своїх випадків, застосуванням мов програмування з визначеними синтаксичними конструкціями у вигляді тексту. Це передбачає написання рядків коду для виконання конкретного завдання у вигляді строго визначеної послідовності команд, наприклад у вигляді коду C або C++. Альтернативою таким мовам програмування є візуальні мови, в основі яких лежить «програмування графічними блоками», яке підтримує концепцію «Drag&Drop». Перевагою використання таких мов програмування є зниження порогу входу розробників програмного забезпечення комп'ютерних систем, зменшення витрат часу для програмування системи, а також зручність і простота модифікації визначених алгоритмів.

Будь-яка мова, яка використовує графічне представлення об'єктів програмування, а під візуальним інтерфейсом яких виконується програмний код, що написаний за допомогою синтаксичних конструкцій, представляє собою візуальну мову програмування.

Візуальна мова програмування дозволяє розробнику мислити природно і логічно, на відміну від традиційних мов програмування. Користувач таких мов повинен думати про те, як він/вона може «пояснити» програму комп'ютеру.

В якості прикладу, візьмемо одну невелику аналогію, наприклад, якщо потрібно запрограмувати таблицю множення, то при використанні традиційної мови програмування необхідно написати цикл і за допомогою нього можна виконати обчислення та вивести на дисплей таблицю множення. На візуальній мові програмування просто потрібно додати блок, який має вбудований код типу «of loop» та вказати значення, а результат буде таким же, як і в попередньому випадку.

Visual Programming Language (VPL) можна використовувати в багатьох сферах, таких як мультимедіа, навчальні цілі, відеоігри, автоматизація та інтеграція різних систем. Розглянемо їх коротко:

– Мультимедіа – VPL допомагає користувачам створювати мультимедіа, не турбуючись про реальний код або інші складні функції. Він звужується до конкретних функцій, і за допомогою них створюється мультимедіа.

– Навчальна мета – використовуються, щоб допомогти студентам у їхніх проектах та ознайомити їх із кодуванням.

– VideoGames – допомагає створювати відеоігри без написання рядків кодів.

– Прототипування систем – у випадку не знання мови програмування, можна створити прототип програмного забезпечення.

Візуальні елементи легко зрозуміти, вона включає в себе різноманітні вбудовані об'єкти, які потрібні під час створення продукту і це зручно для початківців, які не переймаються написанням рядків коду

Додавання спеціального коду також доступне та просте, оскільки дозволяє створювати блоки відповідно до зручності користувача.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ТЕРНОПІЛЬСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
ІМЕНІ ІВАНА ПУЛЮЯ

М А Т Е Р І А Л И

IX НАУКОВО-ТЕХНІЧНОЇ КОНФЕРЕНЦІЇ

«ІНФОРМАЦІЙНІ МОДЕЛІ, СИСТЕМИ ТА ТЕХНОЛОГІЇ»

8–9 грудня 2021 року

ТЕРНОПІЛЬ
2021

УДК 004.4’236
В.В. Яцишин, канд. техн. наук, доцент, Х.В. Яворська
(Тернопільський національний технічний університет імені Івана Пулюя, Україна)

ВІДМІНОСТІ LOW-CODE/NO-CODE РОЗРОБКИ

UDC 004.4’236
V.V. Yatsyshyn PhD, Assoc. Prof., K.V. Yavorska

DIFFERENCES IN LOW-CODE/NO-CODE DEVELOPMENT

Платформи розробки Low-Code/No-Code – це типи візуальних середовищ розробки програмного забезпечення, які дозволяють розробникам перетягувати компоненти програми, з’єднувати їх разом і створювати мобільні або веб- програми. Ці платформи часто обговорюються як синонім методів розробки, які вони втілюють.

Модульні підходи Low-Code/No-Code дозволяють розробникам швидко створювати програми, позбавляючи їх від необхідності писати код рядок за рядком. Вони також дозволяють бізнес-аналітикам, офісним адміністраторам, власникам малого бізнесу та іншим особам, які не є розробниками програмного забезпечення, створювати та тестувати програми. Ці люди можуть створювати програми, практично не знаючи традиційних мов програмування, машинного коду або розробки компонентів платформи, що налаштовуються. Системи Low-Code/No-Code пропонують однакові фундаментальні переваги, але їх назви вказують на ключову різницю між цими двома методами розробки додатків.

Розробка Low-Code вимагає від користувачів певного рівня кодування, хоча й набагато менше, ніж це потрібно для традиційної розробки додатків. Професійні розробники та програмісти використовують низький код для швидкої доставки додатків і перекидання своїх зусиль із простих завдань програмування на більш складну та унікальну роботу, яка має більший вплив та більшу цінність для організації. Не IT-спеціалісти з деякими знаннями програмування також використовують інструменти з низьким кодом для розробки простих програм або розширених функцій у програмі.

Розробка No-Code орієнтована на нетехнічних користувачів у різних бізнес-функціях, які розуміють бізнес-потреби та правила, але володіють невеликим досвідом кодування та навичками мови програмування.

Користувачі можуть використовувати безкодовий код для легкого та швидкого створення, тестування та розгортання своїх бізнес-додатків, якщо вибрані інструменти відповідають цим основним функціям і можливостям.Існують також деякі відмінності в тому, як користувачі застосовують Low-Code/No-Code. No-code зазвичай використовується для створення додатків для обробки простих функцій. Low-Code також можна використовувати в цих випадках, але додатково для створення програм, які запускають процеси, які є критичними для бізнесу або основних систем організації, наприклад, певні інтеграції та ініціативи цифрової трансформації.

Межа між Low-Code/No-Code не завжди чітка - і це переходить на самі платформи з низьким кодом і без коду. Багато аналітиків технологічних продуктів вважають частину ринку низького коду без коду, оскільки навіть найпотужніші платформи вимагають певного рівня кодування для частин процесу розробки та розгортання додатків. Платформи без коду – це спеціалізований тип хмарної платформи з низьким кодом, у якій необхідні візуальні компоненти відповідають галузевим функціям, конкретному напрямку бізнесу або підтримують корпоративний бренд конкретної компанії.