МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: 123 «Комп'ютерна інженерія»

Освітня програма: «Комп'ютерна інженерія»

Група: 2БКС-28

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

здобувача освіти денної форми навчання

БКС.28.18.000.КРБ

НЕСТЕРЕНКО ВОЛОДИМИРА ДМИТРОВИЧА

м. Одеса
2024 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: **123 «Комп'ютерна інженерія»**

Освітньо-професійна програма: **«Комп'ютерна інженерія»**

Група: **2БКС-28**

ПОЯСНЮВАЛЬНА ЗАПИСКА

До кваліфікаційної роботи бакалавра на тему: **Аналіз засобів взаємодії з web-хостингом GitHub з метою підвищення навичок у IT-галузі**

Проектний матеріал складається з пояснювальної записки на 60 сторінках та графічного (презентаційного) матеріалу на 10 аркушах (слайдах)

Виконавець _____ (Нестеренко В.Д.)

Керівник проекту _____ (Джабраилов Д.В.)

Консультанти:

з розділу охорони праці та техніки безпеки _____ (Чорновол Н.І.)

з нормоконтролю _____ (Петрашова В.І.)

старший консультант _____ (Кривченко Ю.В.)

До захисту допущений

Завідувач кафедри _____ (Іванова Л.В.)

Завідувач відділення _____ (Скорнякова О.В.)

Захист «24» 06 2024 р. Протокол ЕК № 1

Оцінка ЕК 5 (відмінно) 95

Секретар ЕК _____

АНОТАЦІЯ

У даній роботі досліджується використання платформи GitHub для управління проектами та підвищення професійних навичок в галузі інформаційних технологій.

GitHub, заснований на системі контролю версій Git, надає розробникам широкий набір інструментів для зберігання та управління кодом, відстеження помилок і завдань, а також для спільної роботи над проектами.

У роботі розглянуто ключові функціональні можливості GitHub, такі як репозиторії, pull requests, issues і GitHub Actions.

Особлива увага приділена методам підвищення якості програмного забезпечення через використання механізмів контролю версій та автоматизації робочих процесів.

Також проаналізовано візуальні та консольні способи роботи з GitHub, включаючи використання командного рядка та інтегрованих середовищ розробки (IDE).

Дослідження включає огляд можливостей GitHub для навчання і професійного розвитку, а також оцінку ефективності платформи в управлінні проектами та організації командної роботи.

Висновки підтверджують, що освоєння GitHub є важливим кроком для початківців спеціалістів в IT, сприяючи їх професійному зростанню та покращенню навичок командної роботи і управління проектами.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

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

Спеціальність *123 «Комп'ютерна інженерія»*

Освітньо-професійна програма *«Комп'ютерна інженерія»*

ЗАТВЕРДЖУЮ:

Заст. дир. з НВР *Беркань І.В.*

"15" 01 2024 р.

ЗАВДАННЯ

на кваліфікаційну роботу бакалавра

Здобувачеві освіти *Нестеренко Володимиру Дмитровичу*

(прізвище, ім'я, по батькові)

1. Тема проекту *Аналіз засобів взаємодії з web-хостингом GitHub з метою підвищення навичок у ІТ-галузі*

затверджена наказом по коледжу від "2" 11 2023 р. № 244-12-ОД

2. Термін здачі закінченого проекту 10.06.2024

3. Вихідні данні до проекту *Технології систем контролю версій; Використання мови програмування C# та стандартних бібліотек Windows Form; Використання принципів елементно-орієнтованої розробки додатків; Реалізація ПЗ для навчання взаємодії з Git-системами.*

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

Аналіз систем контрою версій; Дослідження проблематики важливості знання систем контрою версій; Постановка задачі; Вибір інструментів розробки; Вибір мови програмування; Розробка програми симулятора системи контролю версій; Проведення тестування впливу симулятора на якість використання Git-систем; Тестування та відладка помилок у симуляторі системи контролю версій.

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

Види Git-систем та важливість навичок користування ними; Мета роботи та цілі для досягнення поставленого завдання; Огляд технологій розробки ПЗ для навчання взаємодії з Git-системами; Проектування ПЗ для навчання взаємодії з Git-системами; Розробка ПЗ для навчання взаємодії з Git-системами; Тестування та відлагодження ПЗ для навчання взаємодії з Git-системами; Аналіз ефективності використання розробленого ПЗ.

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

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Основний розділ	*Джабраілов Д.В.*		
Розділ охорони праці	*Чорновол Н.І.*		
Нормоконтроль	*Петрашова В.І.*		
Старший консультант	*Кривченко Ю.В.*		

7. Дата видачі завдання 13.04.2024

Керівник *Джабраілов Д.В.* (підпис)

Завдання прийняв до виконання *Нестеренко В.Д.* (підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/р	Назва етапів дипломного проекту	Термін виконання етапів дипломного проекту (роботи)	Відмітка про виконання
1	*Вступ та аналіз завдання*	*04.05.2024*	виконав
2	*Дослідження проблеми, що розглядається*	*08.05.2024*	виконав
3	*Формування методів вирішення проблематики питання*	*12.05.2024*	виконав
4	*Обрання технологій для розробки необхідного ПЗ*	*14.05.2024*	виконав
5	*Проектування ПЗ для навчання*	*15.05.2024*	виконав
6	*Розробка ПЗ для навчання*	*16.05.2024*	виконав
7	*Тестування та відлагодження ПЗ для навчання*	*20.05.2024*	виконав
8	*Виправлення отриманих помилок у ПЗ*	*22.05.2024*	виконав
9	*Випробування ПЗ з добровольцями*	*24.05.2024*	виконав
10	*Аналіз ефективності розробленого ПЗ*	*30.05.2024*	виконав
11	*Підготовка роботи до малого захисту*	*10.06.2024*	виконав
12	*Розробка питань з охорони праці*	*11.06.2024*	виконав
13	*Підготовка слайдів презентації*	*12.06.2024*	виконав
14	*Підготовка графічної частини проекту*	*13.06.2024*	виконав
15	*Підготовка проекту до захисту та тестування ПП*	*14.06.2024*	виконав

Дипломник (підпис)

Керівник (підпис)

Формат	Зона	Поз.	Позначення	Назва	Кіл.	Примітка
				Документація		
			БКС 28. 18 000. 00 КРБ	Випускна робота		
А4			БКС 28. 18 000. 00 КРБ ПЗ	Пояснювальна записка	1	

Зм.	Арк.	№ докум.	Підпис	Дата	БКС 28. 18 000. 00 КРБ ПЗ		
Розробив		Нестеренко В.Д.		13.06	Аналіз засобів взаємодії з web-хостингом GitHub з метою підвищення навичок у ІТ-галузі	Літ.	Аркуш 4 / Аркушів 61
Перевірив		Джабраилов Д.В.		13.06		Н Д П	
Н. Контр.		Петрашова В.І.		13.06		ВСП "ОТФК ОНТУ" гр. 2БКС-28	
Затверд.		Іванова Л.В.		13.06			

ЗМІСТ

ВСТУП……7
1 ОСНОВНИЙ РОЗДІЛ……9
1.1 Аналіз систем контрою версій……9
1.2 Дослідження проблематики важливості знання систем контрою версій…16
1.3 Постановка задачі……26
1.4 Вибір інструментів розробки……29
1.4.1 Характеристика IDE: Visual Studio (VS) ……29
1.4.2 Характеристика IDE: IntelliJ IDEA……32
1.4.3 Характеристика IDE: PyCharm……34
1.4.4 Обґрунтування обрання Visual Studio (VS) як основну IDE……35
1.5 Вибір мови програмування……36
1.6 Розробка програми симулятора системи контролю версій……38
1.7 Проведення тестування впливу симулятора на якість використання Git-систем……48
1.8 Тестування та відладка помилок у симуляторі системи контролю версій……53
2 ОХОРОНА ПРАЦІ……55
2.1 Вступ……55
2.2 Аналіз та безпека умов праці працівника на робочому місці……55
2.2.1Організація робочого місця……55
2.2.2Аналіз шкідливих та небезпечних чинників……55
2.2.3Організація робочого місця. ……56
2.2.4Мікроклімат……57
2.2.5Освітлення……58
2.2.6Електробезпека……58
2.3 Пожежна безпека……58
2.4 Висновки……59
ВИСНОВКИ……60
ПЕРЕЛІК ВИКОРИСТАНИХ ІНФОРМАЦІЙНИХ ДЖЕРЕЛ……61

ВСТУП

GitHub є однією з найпопулярніших платформ для хостингу проєктів та спільної розробки програмного забезпечення, що базується на системі контролю версій Git. Ця платформа надає розробникам широкий спектр можливостей, включаючи зберігання коду, що дозволяє зберігати та керувати репозиторіями, відстеження помилок через issues для управління дефектами та запитами на нові функції, управління завданнями за допомогою інструменту Projects для організації та відстеження прогресу роботи. Також GitHub спрощує спільну роботу завдяки механізму pull requests, який забезпечує огляд та обговорення змін у коді перед їх об'єднанням, підтримує інтеграцію з багатьма сторонніми сервісами, такими як CI/CD системи, системи для моніторингу та сервіси для тестування, що дозволяє автоматизувати процеси, такі як тестування та розгортання додатків. За допомогою GitHub Actions розробники можуть створювати власні робочі процеси для автоматизації різних завдань, що робить процес розробки більш ефективним та надійним.

Використання GitHub дозволяє командам розробників ефективно взаємодіяти, ділитися знаннями та підтримувати високий рівень якості програмного забезпечення. Ця платформа забезпечує зручне зберігання коду, відстеження змін та помилок, управління завданнями і полегшує спільну роботу через механізми pull requests та code review, що сприяє покращенню якості кінцевого продукту. GitHub також інтегрується з різними сторонніми сервісами для автоматизації тестування, розгортання та інших процесів, що робить розробку більш ефективною. Для початківців у IT-галузі освоєння GitHub є важливим кроком на шляху до професійного розвитку, оскільки цей інструмент широко застосовується в індустрії і допомагає набувати навичок командної роботи, управління проєктами та використання сучасних практик розробки.

У даному аналізі буде розглянуто засоби взаємодії з веб-хостингом GitHub, їх використання для покращення навичок роботи в IT-галузі та переваги для управління проєктами та співпраці. GitHub пропонує зручний інтерфейс для

зберігання та керування кодом, відстеження змін через систему контролю версій Git, що дозволяє розробникам ефективно керувати версіями проєктів та працювати з різними гілками. Інструменти для відстеження помилок і управління завданнями допомагають організувати роботу над проєктами, що сприяє підвищенню продуктивності та якості кінцевого продукту. Використання pull requests та code review дозволяє розробникам обмінюватися знаннями, здійснювати огляд коду та впроваджувати зміни лише після їх ретельної перевірки, що сприяє навчанням і підвищенню професійних навичок. Інтеграція з іншими сервісами, такими як CI/CD системи та інструменти для моніторингу, дозволяє автоматизувати багато процесів, що полегшує тестування та розгортання програмного забезпечення. Таким чином, використання GitHub сприяє покращенню навичок розробників, полегшує управління проєктами та забезпечує ефективну співпрацю в команді, що є ключовими аспектами успішного розвитку в IT-галузі.

1 ОСНОВНИЙ РОЗДІЛ

1.1 Аналіз систем контрою версій

GitHub – це веб-платформа для хостингу та спільної розробки програмного забезпечення, яка використовує систему контролю версій Git. Вона надає розробникам можливість зберігати свої проекти в репозиторіях, співпрацювати з іншими розробниками, відстежувати зміни в коді, управляти версіями і випусками програмного забезпечення.

GitHub дозволяє ефективно організувати командну роботу, пропонуючи інструменти для обговорення змін у коді через pull request, здійснення код-рев'ю, та автоматизації процесів розробки за допомогою GitHub Actions. Він також інтегрується з багатьма іншими сервісами та інструментами розробки, що робить його потужним засобом для управління проектами будь-якої складності. Завдяки GitHub, розробники можуть забезпечити високу якість програмного забезпечення, швидке виявлення і виправлення помилок та ефективну координацію роботи всіх учасників проекту. На рисунку 1.1 можна побачити середу GitHub:

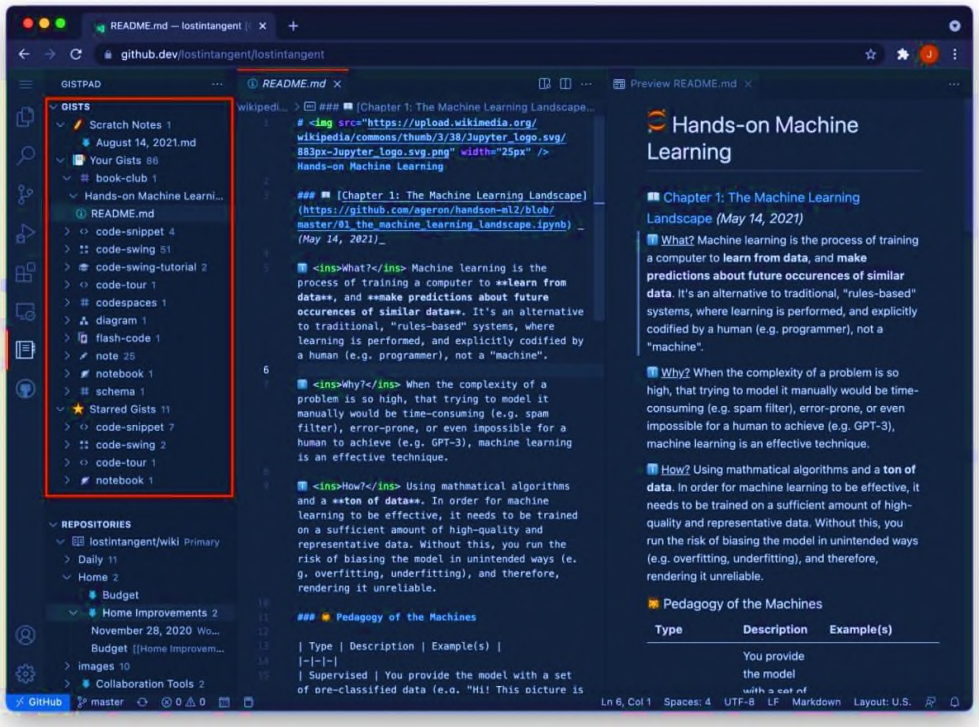


Рисунок 1.1. Середа GitHub

Одним із основних засобів є контроль версій за допомогою Git. Це є основою для роботи з кодом на GitHub. Git дозволяє відстежувати зміни в коді, працювати з гілками та об'єднувати зміни з різних джерел, що робить його незамінним інструментом для розробників. Освоєння Git — перший крок для роботи з GitHub, оскільки GitHub будується на його функціональності. Початківцям важливо розуміти концепції гілкування, яке дозволяє розробляти нові функції чи виправляти помилки без впливу на основний код, злиття, яке інтегрує зміни з різних гілок, вирішення конфліктів, що виникають при злитті змін, а також використання командного рядка, який надає більше контролю та гнучкості для ефективного управління кодом.

Репозиторії є основною структурою даних на GitHub, де зберігаються файли проекту та вся історія змін. Вони можуть бути публічними або приватними, що дозволяє контролювати доступ до коду. Управління репозиторіями включає створення нових репозиторіїв, клонування існуючих для локальної роботи, створення форків для внесення змін без впливу на оригінальний проект, оновлення репозиторіїв з останніми змінами та моніторинг внесених змін. Це сприяє організації та структуруванню проектів, забезпечуючи ефективне керування кодом та співпрацю між розробниками.

Issues і Pull Requests є важливими інструментами для управління завданнями та обговорення змін у коді на GitHub. Issues використовуються для відстеження помилок, додавання нових завдань і функцій, що дозволяє команді бачити та планувати роботу. Pull Requests використовуються для пропозицій змін у коді та їх обговорення перед об'єднанням в основну гілку, що забезпечує контроль якості та колективне прийняття рішень. Освоєння цих інструментів допомагає початківцям розробникам ефективно комунікувати з командою, брати участь в обговореннях, вносити покращення в проекти та отримувати зворотній зв'язок, що сприяє їхньому професійному розвитку і покращує колективну роботу над проектами.

GitHub Actions — це інструмент для автоматизації робочих процесів, який дозволяє налаштувати безперервну інтеграцію та безперервне розгортання (CI/CD). Це допомагає підтримувати високий рівень якості коду та прискорює процес

розробки, автоматизуючи рутинні завдання. Знання GitHub Actions дозволяє розробникам автоматизувати тестування, збірку та розгортання проектів, що значно підвищує ефективність роботи і є важливою навичкою у сучасній розробці програмного забезпечення. Цей інструмент сприяє зменшенню кількості помилок і забезпечує стабільність і надійність кінцевого продукту, полегшуючи процеси інтеграції та розгортання.

Проекти та дошки на GitHub є засобами для управління проектами, які дозволяють візуалізувати завдання та контролювати їх виконання. Це корисний інструмент для організації роботи команди, що допомагає відстежувати прогрес і систематизувати робочі процеси. Використання проектів і дошок дозволяє розставляти пріоритети, стежити за виконанням завдань і координувати зусилля команди, що підвищує її продуктивність і ефективність. Завдяки цим інструментам команда може більш чітко бачити загальну картину проекту, своєчасно реагувати на проблеми та зосереджуватися на важливих аспектах роботи, що сприяє успішному завершенню проекту.

GitHub надає широкі можливості для спільної роботи та обміну знаннями, підтримуючи спільне редагування коду і обговорення, а також пропонуючи доступ до навчальних ресурсів і документації. Платформа сприяє колаборації між розробниками, дозволяючи їм працювати над одним кодом, ділитися ідеями та вирішувати проблеми разом. Участь у відкритих проектах, читання документації та використання навчальних матеріалів, доступних на GitHub, допомагають початківцям розробникам накопичувати знання і набувати практичних навичок. Це дає змогу розвиватися професійно, освоювати нові технології і методи, а також покращувати якість коду завдяки взаємодії з досвідченими розробниками та доступу до найкращих практик у галузі.

Вивчення і застосування цих засобів дозволяє початківцям не тільки підвищити свої навички в управлінні проектами та роботі з кодом, але й стати більш цінними учасниками команди розробки. GitHub надає можливість працювати з реальними проектами, що сприяє накопиченню практичного досвіду та розширенню професійних компетенцій. Зараз існує безліч систем контролю

версій розглянемо декілька з них:

GitHub - це онлайн-платформа для розробки програмного забезпечення, яка базується на системі контролю версій Git. Вона дозволяє розробникам спільно працювати над проектами, зберігаючи код у веб-репозиторіях, які можна легко керувати та взаємодіяти з ними. Крім зберігання коду, GitHub надає широкий спектр інструментів для спільної роботи, таких як система відстеження завдань, можливість обговорення коду, проведення перегляду змін, автоматизація процесів з використанням сервісів Continuous Integration та Continuous Deployment (CI/CD), а також інші функції, спрямовані на покращення продуктивності та якості розробки програмного забезпечення. Відкритий характер GitHub дозволяє розробникам спільно працювати над відкритими проектами, а також створювати приватні репозиторії для комерційних та закритих проектів. Загалом, GitHub є потужним інструментом для розробки програмного забезпечення, який допомагає забезпечити ефективну та продуктивну спільну роботу розробників.

GitHub опублікував огляд своєї діяльності за 2022 рік, у якому представлено цікаві дані про платформу. Наприклад, кількість користувачів зросла на 21 мільйон, досягнувши 94 мільйони. Це більше, ніж населення Німеччини! Загалом, користувачі створили 87,5 мільйонів нових репозиторіїв і зробили понад 227 мільйонів запитів на злиття, додавши понад 3,5 мільярда змін.

Найпопулярнішими мовами програмування на GitHub залишаються JavaScript, Python та Java. В цьому році мова програмування C++ випередила PHP і зайняла четверте місце.

Звіт також показує, що в 2022 році популярність GitHub продовжує зростати у всьому світі, за винятком Антарктиди та острова Норфолк. Найбільший приріст нових користувачів спостерігається в Індії, і за такими темпами вони можуть перевершити кількість американських розробників на платформі до 2025 року.

GitHub залишається невід'ємним інструментом для розробників з усього світу, де вони можуть співпрацювати, ділитися кодом і знаходити нові можливості для свого розвитку. Є декілька видів роботи з гіт-системи це Візуальний та Консольний

Візуальні способи роботи з GitHub включають веб-інтерфейс та додаток GitHub Desktop. Веб-інтерфейс GitHub дозволяє користувачам створювати та керувати репозиторіями, створювати пул реквести, відстежувати задачі через систему Issues, а також використовувати GitHub Actions для автоматизації процесів CI/CD. Користувачі можуть переглядати історію комітів, коментувати зміни та об'єднувати гілки, а також використовувати інструменти для організації роботи, такі як проекти та Wiki. GitHub Desktop надає графічний інтерфейс для локальної роботи з репозиторіями, спрощуючи процеси клонування, створення комітів, перегляду історії змін та синхронізації з віддаленими репозиторіями. Цей додаток також включає інструменти для вирішення конфліктів при об'єднанні гілок. На рисунку 1.2 можна побачити приклад візуального інтерфейсу:

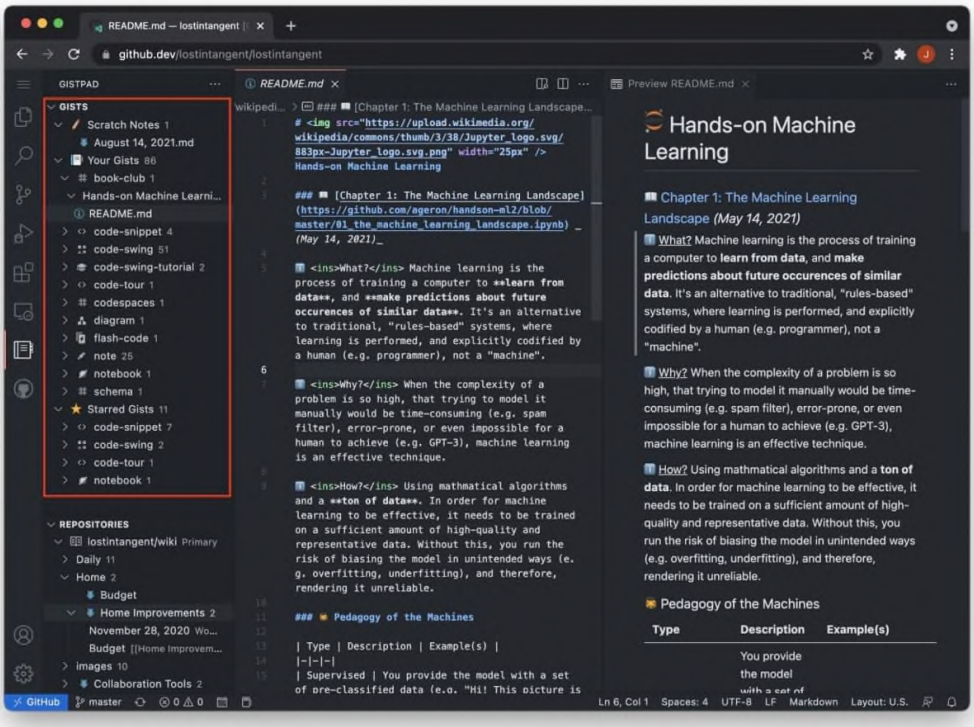


Рисунок 1.2. Приклад візуального інтерфейсу

Консольні способи роботи з GitHub включають використання Git та GitHub CLI. Git є основним інструментом для роботи з системою контролю версій, що дозволяє ініціалізувати репозиторії, клонувати їх, додавати зміни до індексу, створювати коміти, відправляти зміни до віддаленого репозиторію, отримувати зміни та керувати гілками через командний рядок. GitHub CLI інтегрує функції

GitHub у командний рядок, дозволяючи аутентифікуватися, створювати репозиторії, переглядати та створювати пул реквести, а також керувати Issues. Використання цих інструментів надає розробникам гнучкість та потужність при роботі з репозиторіями Git та проектами на GitHub.

Консольні засоби, такі як Git та GitHub CLI, надають розробникам високий рівень гнучкості та контролю над процесом розробки. Вони дозволяють виконувати широкий спектр команд і операцій, що забезпечує глибокий рівень взаємодії з системою контролю версій. Наприклад, командний рядок дозволяє швидко виконувати складні операції, які можуть бути обмежені або недоступні у візуальних інтерфейсах. Консольні засоби також забезпечують кращу автоматизацію процесів за допомогою скриптів, що підвищує ефективність роботи над проектом. До того ж, розробники, які працюють з командним рядком, мають універсальний інструмент, який працює на різних операційних системах і середовищах розробки, що робить його надзвичайно зручним для командної роботи та інтеграції в різні системи. На рисунку 1.3 можна побачити приклад використання систем контролю версій:

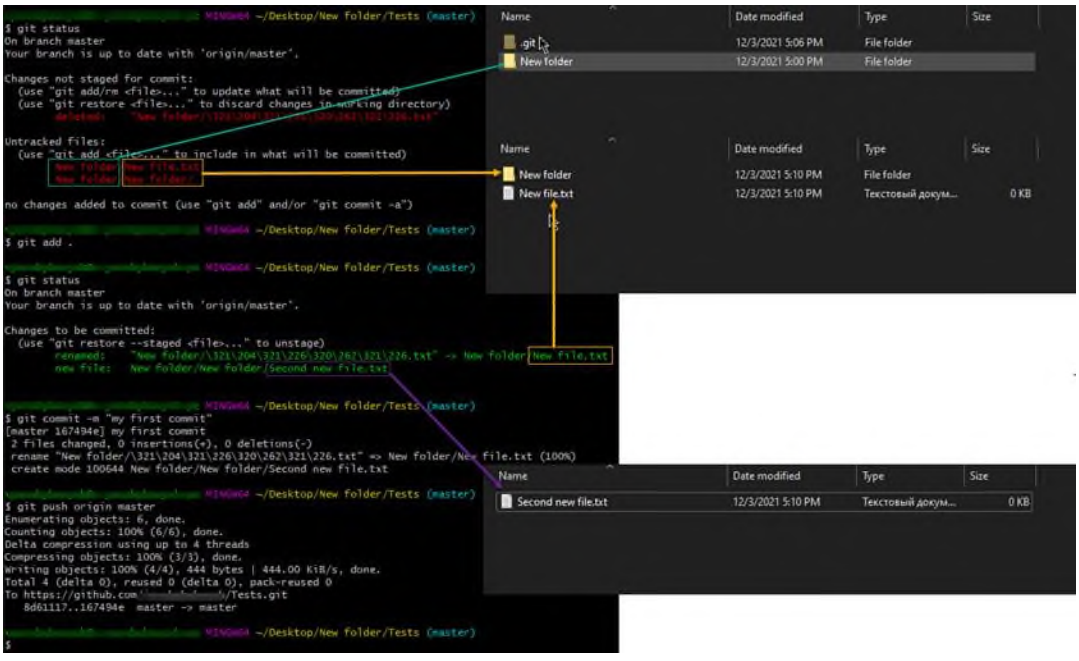


Рисунок 1.3 Приклад консольного використання систем контролю версій.

GitLab - це веб-платформа для розробки програмного забезпечення, яка надає можливості зберігання, керування та спільної роботи над проектами з використанням системи контролю версій Git. Одним з основних відмінностей

GitLab від інших аналогічних платформ, наприклад, GitHub, є те, що він має відкритий вихідний код. Це означає, що користувачі мають можливість самостійно розглядати, модифікувати та розповсюджувати програмне забезпечення GitLab за власними потребами.

GitLab також надає розширені можливості управління проектами, включаючи систему відстеження завдань, управління вимогами, планування та виконання проектів. Крім того, він має вбудовані інструменти для розробки, такі як інтеграція з Continuous Integration / Continuous Deployment (CI/CD), що дозволяє автоматизувати процеси тестування, збірки та розгортання програмного забезпечення.

Додатково, GitLab може бути встановлений на власних серверах, що дозволяє створювати приватні екземпляри платформи та контролювати доступ до даних та функцій. Це особливо корисно для організацій, які мають вимоги щодо безпеки та конфіденційності, або працюють у сфері, де важлива локальна обробка даних.

Загалом, GitLab є потужною інструментальною платформою, яка допомагає командам розробників спільно працювати над проектами та автоматизувати різні аспекти розробки програмного забезпечення.

Bitbucket – це веб-платформа для зберігання та спільної роботи над проектами, яка базується на системі контролю версій Git. Вона подібна до інших популярних платформ, таких як GitHub та GitLab, проте має деякі унікальні особливості. На рисунку 1.4 зображено логотип Bitbucket:

Рисунок 1.4. Логотип Bitbucket

Однією з головних переваг Bitbucket є можливість безкоштовно створювати та використовувати приватні репозиторії для невеликих команд. Це робить його привабливим вибором для проектів, які потребують конфіденційності даних або обмеженого доступу.

Крім того, Bitbucket інтегрується з іншими інструментами розробки програмного забезпечення, такими як Jira, для кращого керування проектами та відстеження завдань. Він також надає можливості для автоматизації процесів розробки за допомогою CI/CD інтеграції.

Окрім цього, Bitbucket може бути встановлений як самостійний продукт на власному сервері, що дозволяє контролювати безпеку та приватність даних, а також налаштовувати його під конкретні потреби організації.

Узагальнюючи, Bitbucket є універсальним інструментом для зберігання та спільної роботи над проектами Git, зокрема з врахуванням потреб в конфіденційності та керуванні проектами.

1.2 Дослідження проблематики важливості знання систем контрою версій

У сучасному світі технологій все більше компаній та команд розробників використовують системи контролю версій для управління кодом і забезпечення ефективної співпраці. Однією з найбільш популярних платформ для цього є GitHub. Незважаючи на численні переваги, багато початківців у IT-галузі стикаються з труднощами при використанні GitHub. Основними проблемами є:

Складність освоєння Git полягає в тому, що новачки можуть зіткнутися з труднощами при розумінні концепцій гілкування, злиття та вирішення конфліктів. Гілкування (branching) вимагає знання того, як створювати та використовувати гілки для розробки нових функцій або виправлення помилок. Процес злиття (merging) може бути заплутаним, особливо коли виникають конфлікти, і потрібно вміти правильно вирішувати ці конфлікти, щоб не зламати код.

Недостатнє розуміння репозиторіїв також створює проблеми для початківців. Вони можуть не знати, як правильно створювати репозиторії і налаштовувати їх для спільної роботи. Клонування репозиторіїв (cloning) є ще однією важливою

навичкою, яку потрібно опанувати, щоб працювати з локальною копією проекту. Форкування репозиторіїв (forking) дозволяє створювати копії чужих проектів для внесення своїх змін без впливу на оригінальний проект. Оновлення репозиторіїв (updating) допомагає синхронізувати локальну копію з віддаленим репозиторієм, щоб завжди мати актуальний код.

Труднощі з управлінням завданнями виникають через те, що новачки можуть не розуміти, як користуватися інструментами GitHub для управління проектами. Наприклад, створення, коментування та закриття завдань (issues) є важливими для відстеження роботи над проектом. Pull Requests використовуються для внесення змін в основну гілку проекту, і новачки можуть не знати, як проходить процес рев'ю та злиття цих змін.

Автоматизація процесів за допомогою GitHub Actions може бути складною для новачків, оскільки вони можуть не знати, як створювати workflow для автоматизації різних завдань, таких як тестування, збірка і деплой. Правильне налаштування і конфігурація workflow є важливою складовою успішної автоматизації.

Організація роботи команди також потребує певного досвіду. Використання дощок (project boards) допомагає відстежувати завдання, призначати відповідальних та планувати роботу. Управління проектами включає організацію командної роботи для забезпечення ефективної комунікації та координації дій.

Освоєння цих аспектів вимагає часу і практики, але вони значно підвищують ефективність роботи з Git та GitHub, дозволяючи ефективно управляти проектами та автоматизувати рутинні завдання.Аналіз засобів взаємодії з web-хостингом GitHub з метою підвищення навичок у IT-галузі

GitHub пропонує безліч інструментів і можливостей, які допомагають вирішити вищезгадані проблеми та підвищити навички у IT-галузі:

Git - це потужна система контролю версій, яка надає зручні інструменти для відстеження змін у програмному коді. Розробники можуть легко переглядати історію змін, відстежувати, які зміни були внесені, і хто саме вніс ці зміни. Крім того, робота з гілками у Git дозволяє створювати відокремлені лінії розвитку для

різних функціональностей чи виправлень помилок. Це дозволяє розробникам працювати паралельно, не впливаючи один на одного, і забезпечує швидкий та ефективний розвиток проекту. Крім того, можливість об'єднання змін з різних джерел допомагає збирати різні частини розробленого функціоналу чи виправлень і об'єднувати їх у єдиний шлях розвитку проекту, що сприяє підтримці чистоти і структурованості кодової бази.

GitHub - це платформа, яка надає широкі можливості для роботи з репозиторіями. Користувачі можуть створювати нові репозиторії для зберігання свого коду, клонувати існуючі репозиторії для роботи з ними локально на своєму комп'ютері. Крім того, вони можуть створювати форки - копії інших репозиторіїв, з якими вони можуть працювати та редагувати незалежно. GitHub також дозволяє відстежувати зміни в репозиторіях, оновлювати їх та контролювати доступ до коду за допомогою різноманітних інструментів для управління репозиторіями. Це допомагає організовувати робочі процеси, забезпечує структурованість роботи та сприяє ефективному співробітництву в команді розробників.

Issues та Pull Requests (PR) - це важливі інструменти у спільній роботі над проєктами на GitHub.

Issues використовуються для відстеження різних аспектів проєкту, таких як помилки, пропозиції та завдання. Вони дозволяють користувачам описувати проблеми або ідеї, маркувати їх за допомогою міток для кращої організації, призначати конкретним розробникам для вирішення та вести обговорення щодо вирішення цих питань. Issues допомагають команді зосередитися на важливих завданнях та відстежувати їх стан.

Pull Requests (PR) використовуються для запропонування змін у коді та обговорення їх з іншими учасниками проєкту перед об'єднанням у головну гілку. Користувачі можуть створювати PR для представлення нового функціоналу, виправлення помилок або будь-яких інших змін у коді. Це дозволяє розробникам обговорювати та переглядати зміни перед тим, як вони вплинуть на основний код проєкту, що сприяє підвищенню якості та безпеки програмного забезпечення.4. GitHub Actions:

GitHub Actions - це потужний інструмент для автоматизації різних робочих процесів у проєктах на GitHub. За допомогою GitHub Actions можна створювати та налаштовувати різноманітні сценарії для автоматизації таких завдань, як збирання, тестування та розгортання коду. Крім того, ви можете налаштувати виконання будь-яких інших автоматизованих завдань, що є необхідними для проєкту. Це дозволяє автоматизувати рутинні операції та спростити робочі процеси, що призводить до збільшення продуктивності розробки та підвищення якості коду.

Проєкти та дошки - це інструменти на GitHub, які сприяють організації та візуалізації робочих завдань та їх виконання.

Проєкти на GitHub дозволяють створювати віртуальні дошки, на яких можна візуалізувати завдання та контролювати їх виконання. Вони можуть бути організовані за різними проектами чи функціональними блоками, що спрощує роботу з командою та дозволяє краще розуміти загальний стан проєкту.

Дошки GitHub - це спеціальні дошки, які дозволяють створювати кастомні персоналізовані дошки для відстеження прогресу завдань, призначення їх розробникам та визначення пріоритетів. Вони можуть бути налаштовані відповідно до потреб команди та дозволяють швидко переглядати та управляти завданнями від початку до кінця розробки. Це допомагає забезпечити більшу структурованість та ефективність управління проєктом.

GitHub надає зручні засоби для спільного редагування коду, обговорення різних аспектів проєкту та доступу до навчальних ресурсів, таких як документація та відкриті проєкти. Це створює сприятливі умови для колаборації та навчання, оскільки користувачі можуть активно співпрацювати над кодом, обмінюватися досвідом та знаннями, а також вивчати нові технології та практикувати свої навички. Будучи частиною великої спільноти розробників на GitHub, користувачі мають можливість взяти участь у відкритих проєктах, долучитися до обговорень та внести свій внесок у розвиток програмного забезпечення. Це допомагає покращувати навички програмування, розуміння кращих практик розробки програмного забезпечення та розвивати себе як професіонала у сфері IT.

Вивчення і застосування засобів GitHub дозволяє початківцям у IT-галузі

значно покращити свої навички в управлінні проєктами та роботі з кодом. GitHub надає можливість працювати з реальними проєктами, що сприяє накопиченню практичного досвіду та розширенню професійних компетенцій. Використання GitHub у навчанні та роботі допомагає початківцям стати більш цінними учасниками команди розробки, ефективно взаємодіяти з колегами та підтримувати високий рівень якості програмного забезпечення.

Також розглянемо, як працюють системи контролю версій. На рисунку 1.5 наведено приклад простої системи репозитарію:

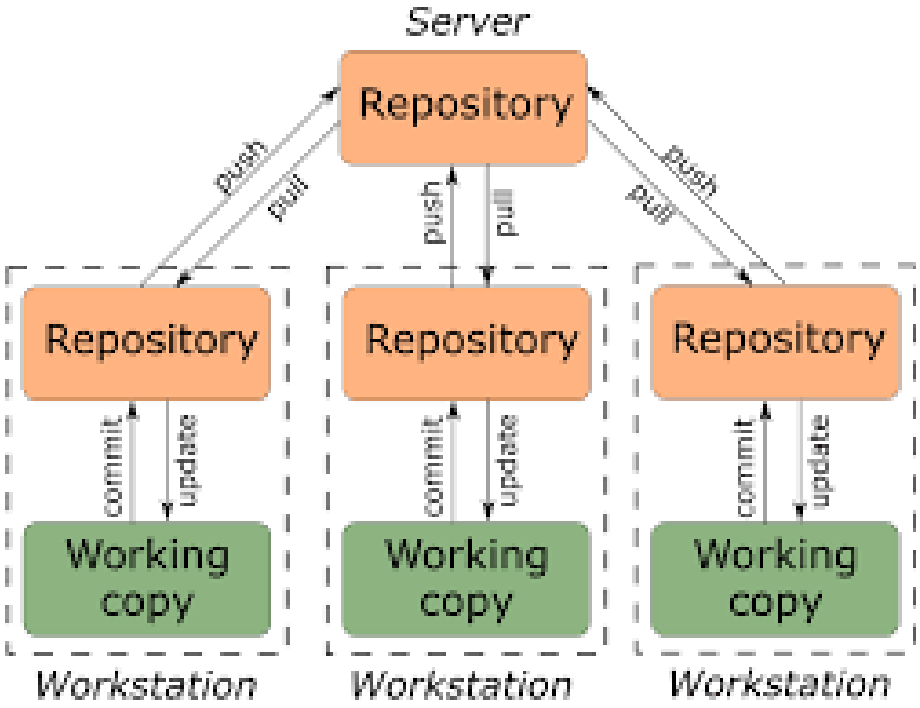


Рисунок 1.5. Приклад побудови простої системи репозиторію

На цій діаграмі зображено модель розподіленої системи контролю версій, де показано взаємодію між центральним сервером і робочими станціями. Центральний сервер містить основний репозиторій, який зберігає всі версії та історію змін проекту. На кожній робочій станції також є локальний репозиторій, який є копією центрального. Кожна робоча станція має робочу копію проекту, з якою розробники працюють.

Процес роботи розробника включає збереження змін у локальний репозиторій, що створює нову версію проекту, а також оновлення робочої копії для синхронізації з локальним репозиторієм. Щоб зміни стали доступними іншим,

розробник відправляє їх до центрального репозиторію. Навпаки, для отримання змін, зроблених іншими розробниками, виконується операція отримання змін із центрального репозиторію в локальний.

Таким чином, ця модель забезпечує ефективну розподілену роботу над проектом, дозволяючи кожному розробнику працювати автономно з можливістю синхронізації змін з іншими через центральний сервер. На рисунку 1.6 наведено діаграму роботи системи контролю версій:

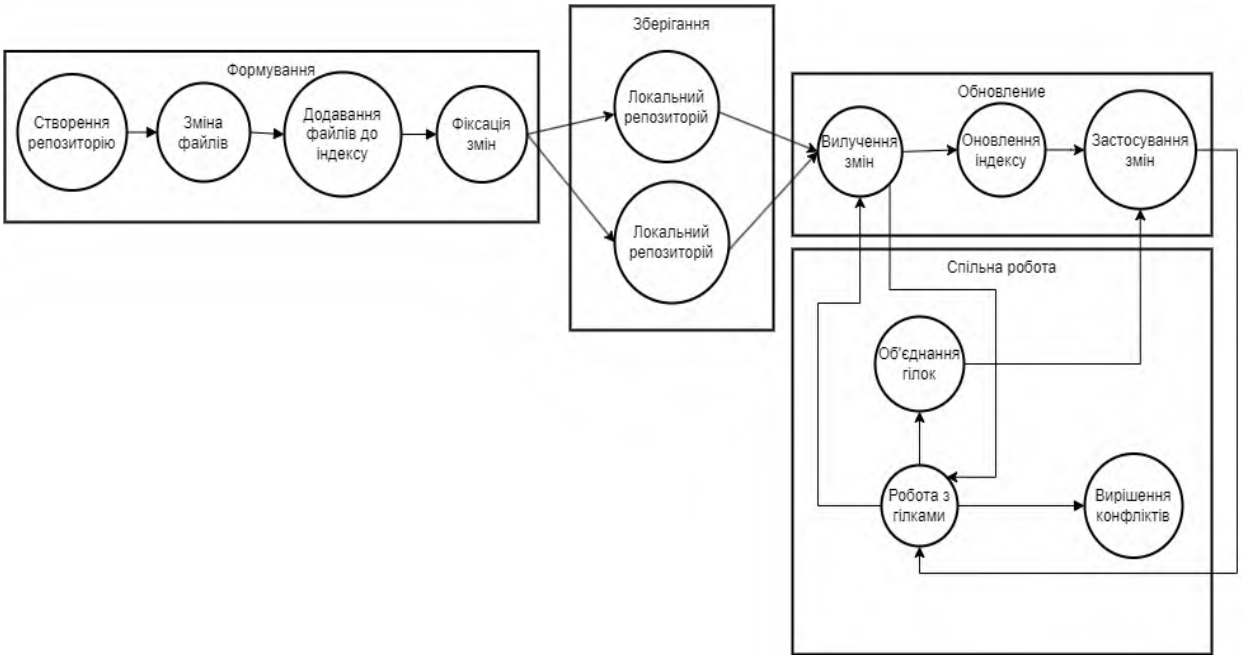


Рисунок 1.6. Діаграма роботи системи контролю версій

Як можна бачити, діаграма поділена на чотири основні групи. Формування - це початковий етап роботи з системою контролю версій, який включає створення і налаштування проекту. На цьому етапі відбувається ініціалізація репозиторію, створення структури проекту з додаванням необхідних директорій і файлів, а також налаштування середовища, що включає визначення файлів і директорій, які потрібно виключити з контролю версій за допомогою файлу .gitignore. Після цього створюється перший коміт для збереження початкового стану проекту. На рисунку 1.7 можна побачити процес формування коміту для репозиторію:

Рисунок 1.7. Процес формування коміту для репозиторію

Зберігання - це процес фіксації змін, внесених у проект. Зміни додаються до індексу (стейджинг) за допомогою команди git add, після чого створюється коміт за допомогою команди git commit, яка фіксує зміни в репозиторії з додаванням опису змін (повідомлення коміту). Для перегляду історії змін використовується команда git log. Також на цьому етапі може створюватися нова гілка проекту за допомогою команди git branch, що дозволяє працювати над різними частинами проекту паралельно. На рисунку 1.8 схематично зображено збереження проекту у новій гілці:

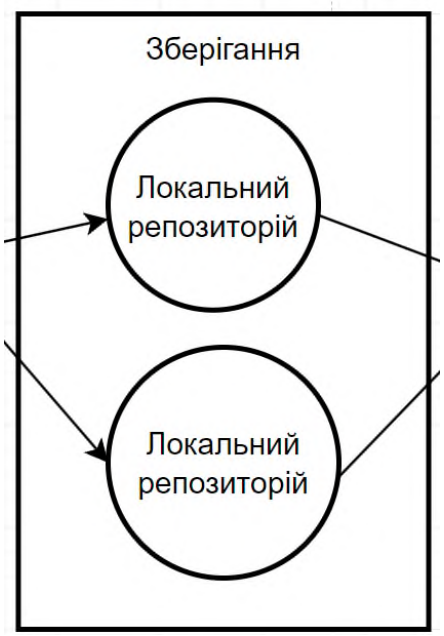


Рисунок 1.8. Схематичне зображення збереження проекту у новій гілці

Оновлення - це процес інтеграції змін з різних гілок або віддалених репозиторіїв. Використовується команда git merge для злиття змін з однієї гілки в іншу, а команда git rebase дозволяє перенести коміти з однієї гілки на іншу для отримання лінійної історії. Команда git pull використовується для отримання і злиття змін з віддаленого репозиторію, а команда git push дозволяє відправити локальні коміти у віддалений репозиторій. На рисунку 1.9 можна побачити

схематичне зображення процесу оновлення проекту з гілки:

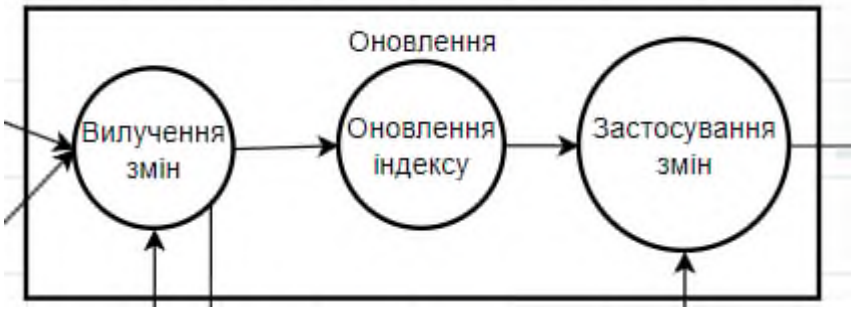


Рисунок 1.9. Схематичне зображення процесу оновлення проекту з гілки

Спільна робота - це процес співпраці з іншими розробниками над одним проектом. Він включає клонування репозиторію для створення локальної копії віддаленого репозиторію, форкання (fork) для створення особистої копії чужого репозиторію для внесення власних змін, і створення запитів на злиття (pull request) для злиття своїх змін з основною гілкою проекту після перегляду і затвердження іншими учасниками проекту. Також важливими аспектами є перегляд і обговорення коду в рамках запитів на злиття та спільне вирішення конфліктів, які можуть виникнути при злитті змін з різних гілок. На рисунку 1.10 схематично зображено спільну роботу над проектом у репозитарії:

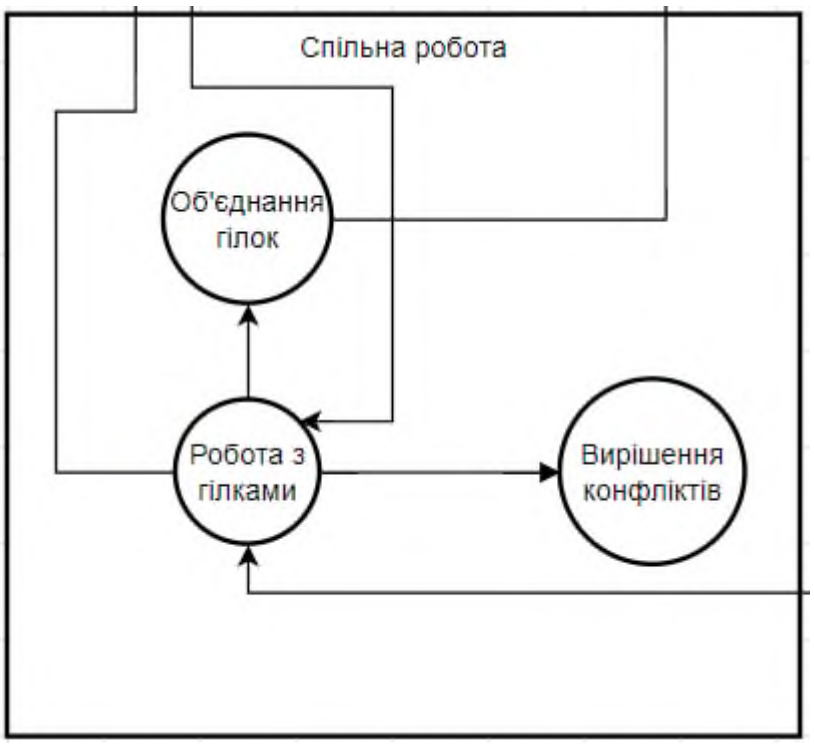


Рисунок 1.10. Схематичне зображення спільної роботи над проектом у репозиторії

Далі ми розглянимо тендеції росту GitHub та взагалі систем контрою версій. Як показують дані зростання загальної кількості користувачів і репозиторіїв на GitHub з лютого 2008 року по листопад 2019 року. На рисунку 1.11 можна побачити цю тенденцію:

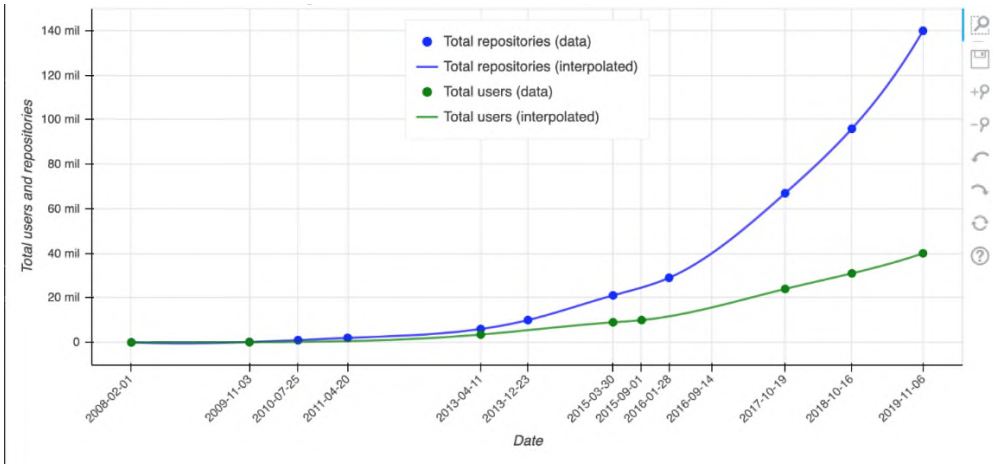


Рисунок 1.11. Графік зростання кількості користувачів GitHub

З початку спостережень кількість репозиторіїв на GitHub значно зросла, про що свідчить інтерпольована синя лінія, яка демонструє експоненціальне зростання. Особливо помітне це зростання після 2013 року, що можна пояснити збільшенням активності користувачів і зростаючою популярністю платформи. Такий стрімкий ріст кількості репозиторіїв свідчить про те, що все більше розробників вибирають GitHub для розміщення своїх проектів, створення нових репозиторіїв та співпраці з іншими. Це також відображає загальну тенденцію до більшої відкритості та колаборації в програмній розробці, що є важливим фактором для інноваційного розвитку в цій сфері.

Зростаюча кількість користувачів на GitHub свідчить про наявність активної та розширюваної ком'юніті розробників, що створює сприятливе середовище для новачків. Така велика і динамічна спільнота забезпечує підтримку нових користувачів, дозволяючи їм швидко знайти допомогу, відповіді на питання та ресурси для своїх проектів. Розробники можуть легко взаємодіяти один з одним, обмінюватися знаннями та досвідом, що сприяє швидкому освоєнню нових технологій та інструментів. Це також підвищує рівень співпраці, оскільки користувачі можуть разом працювати над спільними проектами, вносячи свій вклад і отримуючи цінний зворотній зв'язок, що стимулює подальший розвиток та

інновації на платформі.

Швидке зростання кількості репозиторіїв на GitHub свідчить про те, що на платформі постійно з'являються нові інноваційні проекти, що відображає активний розвиток програмного забезпечення. GitHub сприяє цьому розвитку, надаючи потужні інструменти для версіонування коду, які дозволяють розробникам відстежувати зміни та працювати над своїми проектами більш ефективно. Платформа також пропонує засоби для співпраці, що дозволяють командам розробників спільно працювати над проектами незалежно від їхнього місцезнаходження. Крім того, GitHub інтегрується з іншими сервісами та інструментами, що забезпечує безшовний робочий процес і полегшує використання різних технологій. Усе це робить GitHub важливим інструментом для створення та розвитку сучасного програмного забезпечення, сприяючи інноваціям та ефективній колаборації.

Системи котрою версій використовують майже всюди, не один бфльш меньш великий проект не обходиться без використання системи контрою версій, та їїї використовують майже для всіх мов програмування. На рисунку 1.12 продесонмтрована доля мов програмування у загальній кількості проектів:

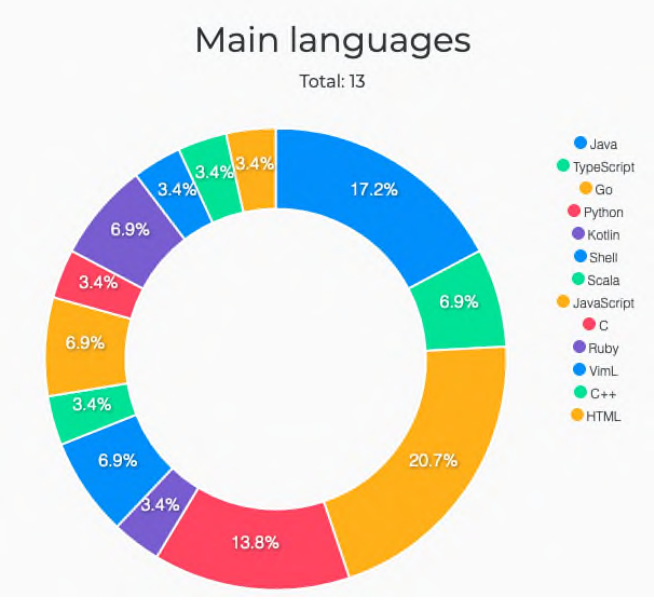


Рисунок 1.12. Діаграма кількості співвідношення мов програмування

Також Системами котрлю версій користуються по всьому світу із чого ми можемо робити висновки що комьюніті дуже велике. На рисунку 1.13 зображено

кількість користувачів по країнам світу:

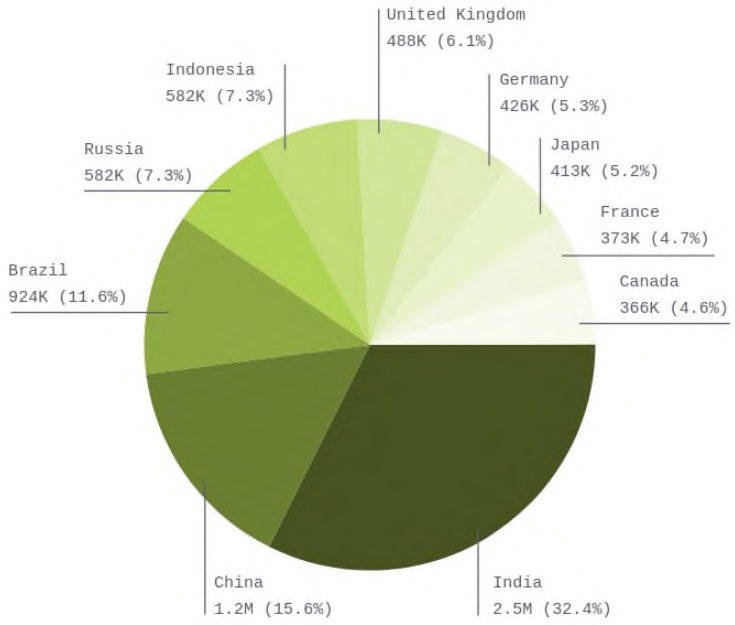


Рисунок 1.13. Діаграма кількості користувачів з різних стран

1.3 Постановка задачі

Аналіз засобів взаємодії з web-хостингом GitHub показує, що існує кілька ключових напрямків, у яких можна зосередити зусилля для підвищення навичок у IT-галузі. Ось методи вирішення основних проблем, з якими стикаються початківці:

Освоєння Git та контролю версій може бути спрощеним та ефективним завдяки різноманітним методам навчання. Інтерактивні курси та тренінги дозволяють студентам освоїти основи Git через практичні приклади, які засвоюються швидше та ефективніше. Практичні заняття, такі як лабораторні роботи або воркшопи, допомагають учням закріпити свої знання через практичний досвід та вирішення реальних завдань з використання Git. Документація та ресурси, такі як відеоуроки та посібники, створюють доступні інформаційні матеріали, які можна використовувати для самостійного вивчення та поглиблення знань про Git. Ці методи разом створюють повноцінне навчальне середовище, яке сприяє успішному освоєнню Git та контролю версій.

Управління репозиторіями може бути спрощеним та ефективним завдяки низці методів та ресурсів. Покрокові інструкції з управління репозиторіями

допомагають користувачам оволодіти основними операціями, такими як створення, клонування, форк, оновлення та моніторинг змін, що сприяє розумінню процесу та відображає кращі практики. Доступ до демонстраційних проєктів дозволяє початківцям практикуватися на реальних прикладах організації репозиторіїв, що допомагає закріпити знання та навички. Крім того, надання рекомендацій щодо структури файлів і каталогів у репозиторіях сприяє їх зручному використанню та полегшує навігацію користувачам, забезпечуючи ефективне управління проектами.

Для ефективного оволодіння процесами використання Issues та Pull Requests пропонуються різноманітні методи навчання. Серія вебінарів і відеоуроків надасть студентам можливість вивчити ключові аспекти використання цих інструментів через практичні демонстрації та інструкції. Практичні завдання, які включають створення та обробку Issues і Pull Requests, допоможуть учням закріпити свої знання та вміння шляхом активної практики. Розробка шаблонів для створення Issues і Pull Requests спростить процес та стандартизує його, що дозволить ефективніше використовувати ці інструменти в робочому процесі. Ці методи навчання разом створюють повноцінні умови для успішного оволодіння управлінням завданнями та обговоренням змін у коді через Issues та Pull Requests на GitHub.

Для ефективного оволодіння GitHub Actions та їх використання у робочих процесах пропонується комплексний підхід до навчання. Створення детальних посібників та відеоуроків дозволить користувачам крок за кроком ознайомитися з основами налаштування та використання GitHub Actions, розкривши різні сценарії автоматизації робочих процесів. Надання демо-проєктів з налаштованими GitHub Actions допоможе учням бачити реальні приклади використання цих інструментів у практиці, особливо у контексті CI/CD. Організація воркшопів дозволить учасникам відразу застосовувати отримані знання на практиці, налаштовуючи GitHub Actions для власних проєктів та отримуючи реальний досвід роботи з цим інструментом. Такий підхід забезпечить глибоке розуміння та практичне вміння використовувати GitHub Actions для автоматизації робочих процесів у різних

сферах розробки програмного забезпечення.

Для сприяння систематизації та управління роботою над проєктами пропонується ряд ініціатив. Спочатку, розробка посібників та шаблонів допоможе користувачам швидко розпочати роботу з проєктами та дошками, надаючи чіткі інструкції та зразки для подальшого використання. Практичні заняття, організовані у формі сесій, дозволять початківцям навчитися створювати та налаштовувати проєкти та дошки для ефективного управління завданнями на практиці. Крім того, демонстрація інтеграції дошок з іншими інструментами управління проєктами допоможе показати користувачам, як можна покращити ефективність та зручність роботи, використовуючи синергію різних інструментів у комбінації з проєктами та дошками на GitHub. Ці ініціативи сприятимуть ефективному та організованому управлінню завданнями та проєктами для користувачів будь-якого рівня досвіду.

Для підтримки колаборації та навчання пропонуються різноманітні ініціативи, спрямовані на покращення навичок та знань початківців у сфері програмування. Залучення початківців до відкритих проєктів з менторською підтримкою досвідчених розробників надасть їм можливість отримати цінний досвід та відчутися частиною спільноти. Регулярні спільні сесії кодування дозволять початківцям працювати разом з досвідченими колегами над реальними завданнями, отримуючи конкретну допомогу та поради. Крім того, доступ до різноманітних ресурсів для самонавчання, таких як книги, статті, онлайн-курси та форуми для обговорення, допоможе початківцям активно поглиблювати свої знання та розвивати навички у власному темпі. Ці ініціативи сприятимуть ефективному навчанню та розвитку в сфері програмування, забезпечуючи початківцям підтримку та ресурси для успішного засвоєння матеріалу та підвищення професійного рівня.

Впровадження цих методів дозволить початківцям у IT-галузі швидше та ефективніше освоїти роботу з GitHub, підвищити свої професійні навички та стати цінними учасниками команд розробки. Використання інтерактивних навчальних матеріалів, практичних занять та спільної роботи з досвідченими розробниками допоможе їм набути впевненості та компетентності у використанні цього

важливого інструменту.

Використання GitHub значно покращує ефективність роботи команд розробників завдяки широкому спектру можливостей, таких як зберігання коду, відстеження змін та помилок, управління завданнями, спільна робота через pull requests та code review, а також інтеграція з іншими сервісами для автоматизації процесів. Ці інструменти сприяють підвищенню якості програмного забезпечення, організації робочого процесу та обміну знаннями серед членів команди. GitHub також допомагає розробникам покращувати свої професійні навички, забезпечуючи реальний досвід роботи з сучасними практиками розробки.

Чому потрібно зробити симулятор Git систем для навчання

Створення симулятора Git систем для навчання є важливим кроком у підготовці початківців до професійної роботи в IT-галузі. Такий симулятор дозволить новачкам безпечно ознайомитися з основними концепціями та командами Git, відстежувати зміни коду, працювати з гілками, виконувати merge та використовувати pull requests. Це забезпечить інтерактивне навчання в умовах, максимально наближених до реальних робочих ситуацій, без ризику пошкодження реальних проєктів. Крім того, симулятор може надавати навчальні матеріали, підказки та приклади, що сприятиме кращому розумінню принципів роботи з системами контролю версій. Такий підхід допоможе новачкам швидше освоїти Git та GitHub, підвищуючи їхню впевненість і готовність до роботи в команді, що є критично важливим для успішного професійного розвитку в IT-індустрії.

1.4 Вибір інструментів розробки

1.4.1 Характеристика IDE: Visual Studio (VS)

Visual Studio (VS) — це потужна інтегрована середа розробки (IDE), розроблена компанією Microsoft. Вона є однією з найпопулярніших IDE серед розробників, особливо для розробки на мовах, що підтримуються платформою .NET, таких як C#, F# та VB.NET.

Visual Studio - це потужне інтегроване середовище розробки (IDE), яке надає широкі можливості для роботи з різними мовами програмування. Серед підтримуваних мов є такі популярні як C#, C++, F#, VB.NET, Python, JavaScript,

TypeScript та багато інших. У Visual Studio також є вбудовані інструменти, що дозволяють розробникам створювати різноманітні типи програм, включаючи веб-додатки, хмарні сервіси, мобільні додатки та настільні програми. Це забезпечує розробникам зручну та продуктивну робочу обстановку для втілення їхніх ідей у реальність незалежно від мови програмування чи типу програмного забезпечення, яке вони розробляють.

Visual Studio має розширену підтримку для платформи .NET, що включає глибоку інтеграцію з .NET та .NET Core, а також засоби для розробки різноманітних типів додатків. Це охоплює не лише традиційні консольні та настільні програми, але й веб-додатки на основі ASP.NET. Завдяки цьому, розробники можуть використовувати різноманітні функції, які надаються платформою .NET, такі як механізми аутентифікації та авторизації, інтеграція з базами даних, а також підтримка мови C#, для створення потужних та ефективних додатків. Разом з цими можливостями, Visual Studio надає розробникам широкий набір інструментів для налагодження, тестування та розгортання додатків, що дозволяє створювати високоякісне програмне забезпечення на базі платформи .NET.

Visual Studio пропонує розширені можливості для розробників, включаючи інтелектуальне автозаповнення та інструменти аналізу коду, які допомагають підвищити продуктивність та якість програмного забезпечення. Функція IntelliSense забезпечує контекстне автозаповнення, підказки по синтаксису, виявлення помилок у реальному часі та рекомендації з виправлення коду, що значно полегшує процес написання коду та допомагає уникнути потенційних помилок. Крім того, у Visual Studio є різноманітні інструменти для глибокого аналізу коду та оптимізації продуктивності, які допомагають розробникам зрозуміти структуру програмного забезпечення, виявити проблеми та шляхи їх вирішення, а також підвищити ефективність роботи над проектом. Всі ці функції роблять Visual Studio потужним інструментом для розробки програмного забезпечення, який допомагає розробникам створювати високоякісні та ефективні додатки.

Visual Studio надає розробникам інтегровану підтримку для хмарних сервісів, зокрема з Microsoft Azure. Це дозволяє розробникам легко розробляти, тестувати та розгортати свої хмарні додатки безпосередньо з інтерфейсу розробки. Інтеграція з Azure включає в себе інструменти для управління хмарними ресурсами та сервісами, що дозволяє розробникам ефективно керувати своїми проектами та ресурсами прямо з інтегрованого середовища розробки. Це спрощує розробку та розгортання хмарних додатків та дозволяє розробникам зосередитися на самому процесі розробки, мінімізуючи необхідність вручного управління інфраструктурою.

Visual Studio відомий своєю величезною кількістю розширень, які доступні через Visual Studio Marketplace. Ці розширення дозволяють розробникам розширити функціональність свого редактора, додати нові інструменти та покращити продуктивність розробки. Крім того, Visual Studio надає можливість налаштування середовища розробки під конкретні потреби кожного розробника. Від індивідуальних налаштувань редактора до налаштувань проекту та робочого потоку - Visual Studio дозволяє розробникам налаштовувати робоче середовище так, як вони цього прагнуть, щоб забезпечити найбільш комфортне та продуктивне оточення для розробки.

Visual Studio надає широкі можливості для розробки на платформі .NET, включаючи ASP.NET, WPF, WinForms та інші. Його інтегроване середовище сприяє зручній розробці, налагодженню та розгортанню додатків на основі технологій Microsoft.

Visual Studio має вбудовані засоби для ефективного налагодження коду, такі як дебагер з точками зупинки, моніторингом змінних та іншими функціями. Крім того, інструменти аналізу коду допомагають виявляти помилки та оптимізувати продуктивність програмного коду.

Visual Studio підтримує широкий спектр мов програмування та типів проєктів, що дозволяє розробникам працювати з різноманітними технологіями та платформами.

Visual Studio відомий своєю високою продуктивністю та надійністю, що

дозволяє розробникам працювати ефективно та безпечно над своїми проектами.

Ліцензійні витрати на комерційне використання Visual Studio можуть бути значними, особливо для невеликих компаній або початкових підприємств. Visual Studio є потужним інтегрованим середовищем розробки, що може вимагати значно більше системних ресурсів порівняно з легкими редакторами коду, що може призвести до обмежень для користувачів з менш потужними комп'ютерами.

Для новачків або тих, хто переходить з більш простих середовищ розробки, Visual Studio може виявитися складним через велику кількість функцій та налаштувань, які можуть здатися складними для освоєння. Visual Studio є одним з найпотужніших інструментів для розробки програмного забезпечення, що надає всі необхідні можливості для професійної розробки, тестування та підтримки додатків на різних платформах. На рисунку 1.14 зображено середу розробки Visual Studio Code:

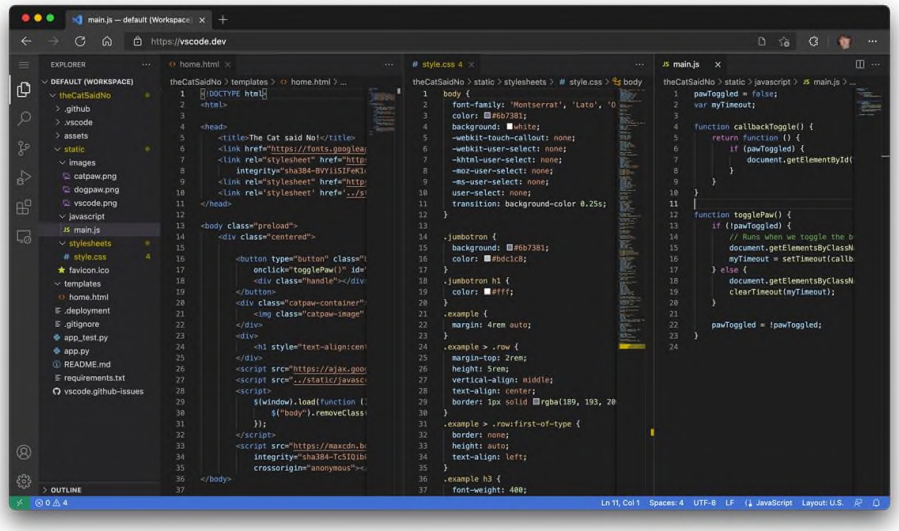


Рисунок 1.14. Зовнішній вигляд середи розробки Visual Studio Code

1.4.2 Характеристика IDE: IntelliJ IDEA

IntelliJ IDEA — це професійна інтегрована середа розробки (IDE), розроблена компанією JetBrains, яка відома своєю потужною підтримкою Java та інших мов програмування. IntelliJ IDEA є однією з найкращих IDE для розробки на Java та інших JVM-мовах.

Переваги IntelliJ IDEA очевидно виокремлюють її як потужний інструмент для розробки програмного забезпечення. Її глибока інтеграція з Java та JVM-

мовами, такими як Kotlin, Groovy, Scala, надає розробникам широкі можливості для творчості в розробці програм. Потужні інструменти аналізу та рефакторингу коду роблять процес розробки більш продуктивним та ефективним. Інтеграція з популярними фреймворками та іншими інструментами для розробки, такими як Spring, Hibernate, Maven та Gradle, сприяє зручності розробки. Що найважливіше, висока продуктивність та надійність дозволяють розробникам концентруватися на розвитку програмного забезпечення, забезпечуючи стабільну та ефективну роботу з проєктами.

Незважаючи на свої переваги, IntelliJ IDEA також має свої недоліки. Висока вартість для комерційного використання може бути бар'єром для деяких команд та розробників, особливо для стартапів та невеликих підприємств. Крім того, через велику кількість функцій та можливостей, IntelliJ IDEA може вимагати більше системних ресурсів порівняно з легкими редакторами коду, що може вплинути на продуктивність роботи на менш потужних комп'ютерах. Також важливо враховувати, що деякі функції можуть бути складними для початківців, особливо тих, хто тільки починає свій шлях у програмуванні та розробці програмного забезпечення. IntelliJ IDEA є однією з найпотужніших і найзручніших IDE для розробки на Java та інших мовах, що працюють на JVM. На рисунку 1.15 зображено зовнішній вигляд середи розробки IntelliJ IDEA:

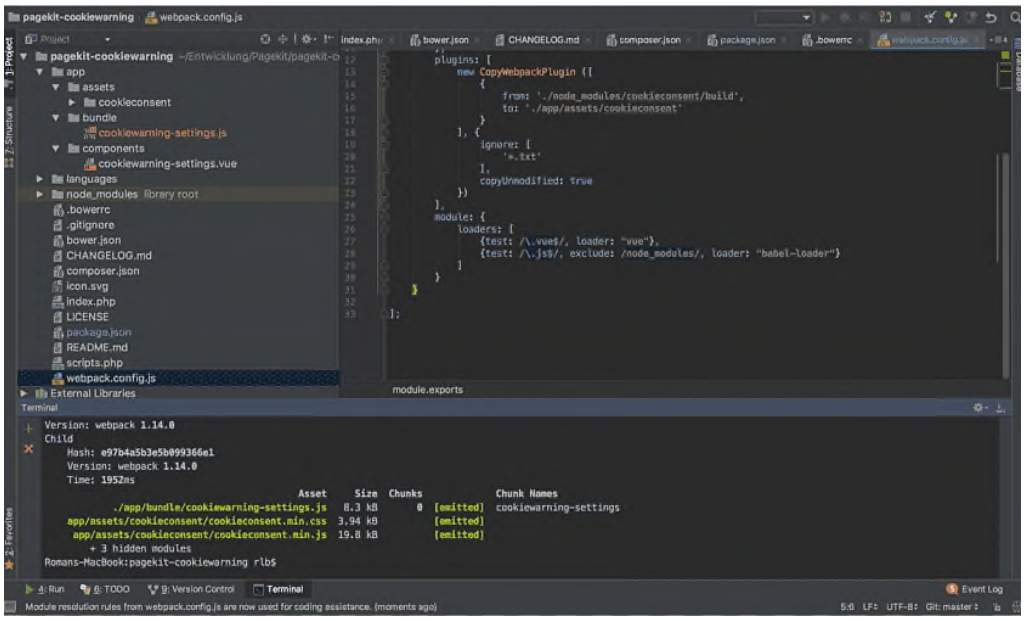


Рисунок 1.15. Зовнішній вигляд середи розробки IntelliJ IDEA

1.4.3 Характеристика IDE: PyCharm

PyCharm — це потужна інтегрована середа розробки (IDE) для мови програмування Python, розроблена компанією JetBrains. PyCharm є одним з найпопулярніших інструментів для розробників, що спеціалізуються на Python, завдяки своїм численним можливостям та зручності використання.

PyCharm - це відоме інтегроване середовище розробки (IDE) для Python, яке забезпечує глибоку інтеграцію з цією мовою програмування. Завдяки повній підтримці Python, користувачі можуть користуватися такими корисними функціями, як автозаповнення коду, аналіз коду в реальному часі, рефакторинг та відладка, що дозволяє значно підвищити продуктивність та ефективність розробки. Крім того, PyCharm підтримує різні версії Python та віртуальні середовища, такі як venv і conda, що дозволяє розробникам легко керувати середовищами та забезпечує гнучкість у виборі конфігурацій для їх проектів. На рисунку 1.16 зображено зовнішній вигляд середи розробки PyCharm:

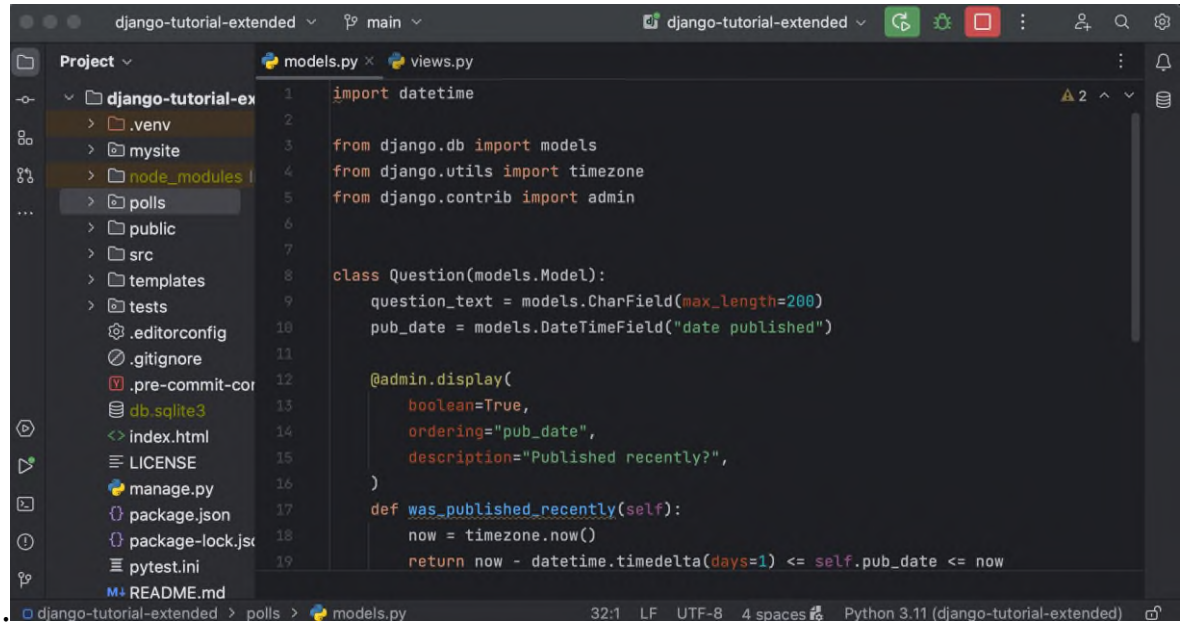


Рисунок 1.16. Зовнішній вигляд середи розробки PyCharm

PyCharm відомий своєю глибокою інтеграцією з Python та популярними фреймворками, такими як Django та Flask, що забезпечує зручний та продуктивний розвиток веб-додатків. Завдяки потужним інструментам аналізу та рефакторингу коду, розробники можуть ефективно виявляти й виправляти помилки, підвищуючи якість свого програмного забезпечення. Крім того, наявність інструментів для веб-

розробки та наукових обчислень дозволяє розробникам працювати в різних областях, від створення веб-додатків до аналізу даних. Загалом, завдяки високій продуктивності та надійності, PyCharm є потужним інструментом для реалізації різноманітних проектів з Python.

PyCharm, хоча і є потужним інструментом для розробки програмного забезпечення, має кілька недоліків. Одним з них є висока вартість для комерційного використання, що може стати перешкодою для невеликих компаній або початківців. Крім того, через велику кількість функцій та можливостей, PyCharm може виявитися складним для освоєння початківцями, що може вимагати часу на вивчення. Крім того, використання PyCharm може потребувати більше системних ресурсів у порівнянні з легкими редакторами коду, що може вплинути на продуктивність на старіших або менш потужних пристроях. Таким чином, хоча PyCharm є потужним інструментом для розробників, варто враховувати ці недоліки при виборі середовища розробки.

PyCharm є однією з найкращих IDE для розробки на Python, що забезпечує всі необхідні інструменти для професійної розробки, тестування та підтримки програмного забезпечення.

Вибір IDE (Integrated Development Environment) для розробки проекту з аналізу засобів взаємодії з веб-хостингом GitHub для підвищення навичок в IT-галузі є ключовим етапом, оскільки від цього залежить продуктивність, якість коду та швидкість розробки. У даному випадку рекомендується обрати

1.4.4 Обґрунтування обрання Visual Studio (VS) як основну IDE

Широкий спектр мов програмування: VS підтримує не лише мови, що популярні серед веб-розробників (такі як JavaScript, TypeScript, HTML, CSS), але й інші мови, що використовуються у різних галузях IT, такі як C#, Python, Java тощо. Це дозволяє розробнику працювати з різними типами проектів та використовувати одну IDE для всіх них.

Інтеграція з Git та GitHub: VS має вбудовану підтримку Git, що дозволяє зручно взаємодіяти з репозиторіями на GitHub без необхідності встановлювати додаткові плагіни або інструменти. Це спрощує процес контролю версій та спільної

роботи над проектами з командою.

Інтелектуальні інструменти аналізу коду та дебаггер: VS надає потужні інструменти для аналізу коду та пошуку помилок, що допомагає виявляти та виправляти проблеми швидше. Також, він має високоефективний дебагер з рядом корисних функцій, що спрощує відлагодження програм.

Підтримка різних платформ: VS дозволяє розробляти не лише веб-додатки, але й десктопні, мобільні, хмарні та інші типи програм. Це дозволяє розробнику ефективно працювати з різними типами проектів без необхідності переходити до інших IDE.

Отже, обрання Visual Studio для розробки проекту з аналізу засобів взаємодії з веб-хостингом GitHub дозволить отримати високий рівень продуктивності, зручність у взаємодії з Git та інтелектуальні інструменти для аналізу коду, що в свою чергу підвищить навички розробника в IT-галузі.

1.5 Вибір мови програмування

Java відома своєю платформонезалежністю, що означає, що програми, написані на Java, можуть працювати на будь-якій платформі, яка підтримує віртуальну машину Java (JVM).

Велика спільнота розробників: Java має одну з найбільших спільнот розробників у світі, що призводить до великої кількості доступних бібліотек, фреймворків та ресурсів для розробників.

В порівнянні з іншими мовами, такими як C++ або Rust, Java може мати меншу продуктивність та швидкість виконання через використання віртуальної машини.

Python славиться своїм простим та зрозумілим синтаксисом, що робить його відмінним вибором для початківців та для швидкого розроблення прототипів.

Python має велику кількість бібліотек, які включають в себе все, від обробки рядків до роботи з мережами, що робить його дуже потужним для розробки різних застосунків.

Однією з головних недоліків Python є його швидкодія в порівнянні з іншими мовами, такими як C++ або Java. Він може бути повільнішим у виконанні завдань, особливо при роботі з великими обсягами даних.

C# має значну підтримку від корпорації Microsoft, що призводить до стабільної та надійної розробки. Інтегрована середа розробки (IDE) Visual Studio надає широкі можливості для розробки та налагодження програм.

Крос-платформенність через .NET Core: За допомогою .NET Core, C# може бути використаний для розробки крос-платформенних додатків, що розширює його можливості та ринкову привабливість.

Обмежена крос-платформенність: Хоча .NET Core розширює крос-платформенність, C# все ще вважається основною мовою для розробки додатків під платформу Windows, що може обмежувати аудиторію.

При виборі мови програмування для розробки симулятора системи контролю версій Git важливо врахувати всі переваги та недоліки кожної мови, а також специфіку мого проекту та власних знань і вмінь. Звісно, ось додаткові причини, чому я обратв мову програмування C# для розробки симулятора системи контролю версій Git:

Інтеграція з іншими інструментами Microsoft: Якщо компанія використовує екосистему продуктів Microsoft, використання C# для розробки може забезпечити гладку інтеграцію з іншими інструментами, такими як Azure DevOps, Microsoft Teams і Office 365.

C# є мовою програмування, розробленою компанією Microsoft, тому ви можете розраховувати на активну підтримку, стабільність та надійність.

Інтегрована середа розробки Visual Studio: Visual Studio є потужною інтегрованою середою розробки, яка надає розширені засоби для розробки, тестування та налагодження програм на C#. Це робить процес розробки більш продуктивним і зручним.

Крос-платформенність через .NET Core: За допомогою .NET Core, ви можете розробляти крос-платформенні додатки на C#, що дозволяє симулятору Git бути доступним на різних операційних системах, таких як Windows, macOS і Linux.

Сильна типізація і безпека: C# має сильну систему типів і вбудовані засоби для роботи з безпекою, що допомагає уникнути багатьох типових помилок програмування і забезпечити високу якість коду симулятора Git.

Загалом, обираючи мову програмування для розробки симулятора системи контролю версій Git, важливо враховувати всі переваги та недоліки, а також конкретні потреби проекту і власну експертизу. Враховуючи всі ці фактори, C# може бути відмінним вибором проекту.

1.6 Розробка програми симулятора системи контролю версій

Почнемо з діаграми роботи проекту яка продемонстрована на рисунку 1.17:

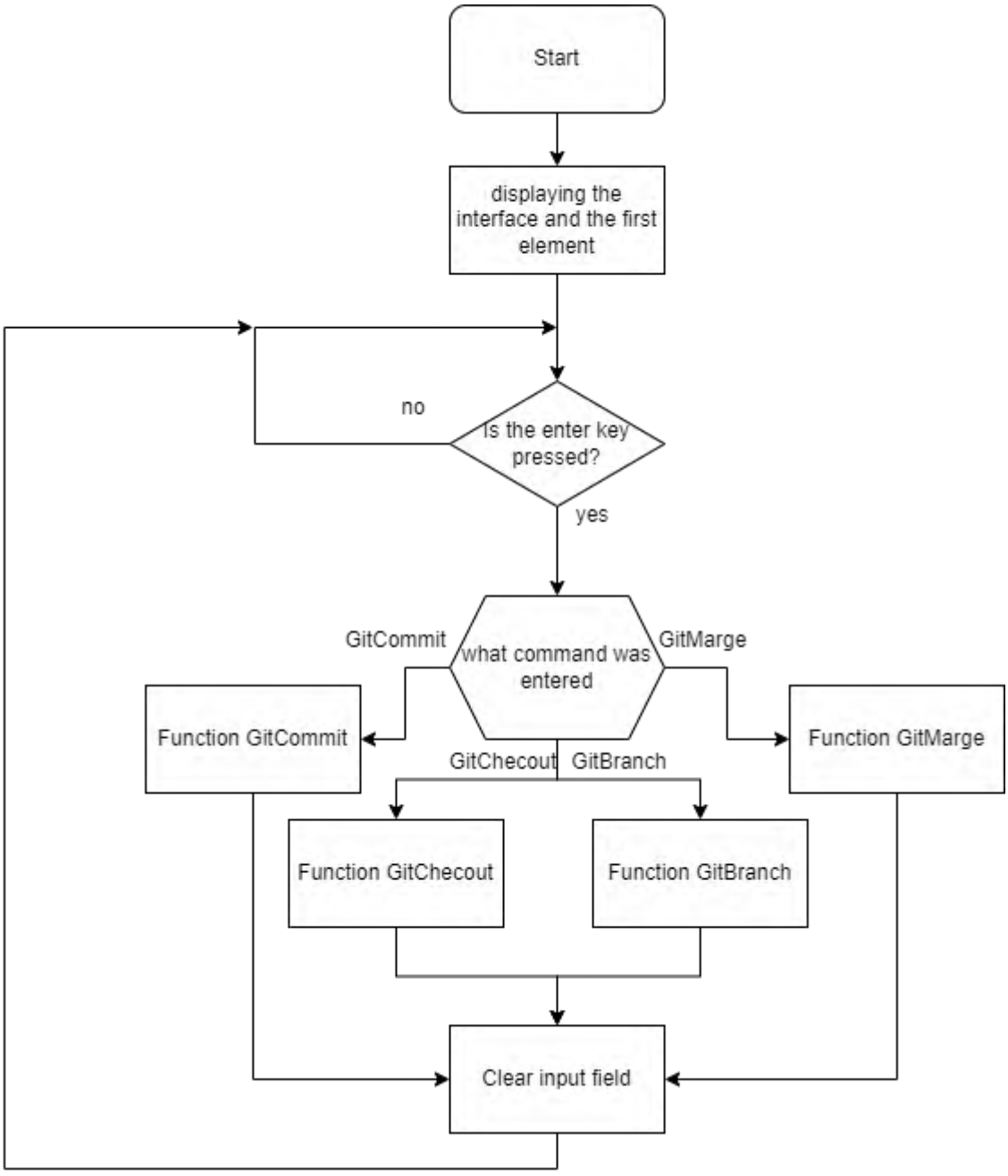


Рисунок 1.17 Діаграма порядку дій у програмі

Ця діаграма описує послідовність дій при взаємодії користувача з інтерфейсом, включаючи перевірку введених команд і виконання відповідних

функцій.

Start: Початок процесу. Це початкова точка блок-схеми.

Displaying the interface and the first element: Відображення інтерфейсу та першого елемента. На цьому етапі користувачу показується інтерфейс програми та початковий елемент.

Is the enter key pressed?: Перевірка натискання клавіші Enter. Це рішення перевіряє, чи натиснув користувач клавішу Enter.

- Якщо клавішу Enter не натиснуто, процес залишається на цьому етапі.

- Якщо клавішу Enter натиснуто, процес переходить до наступного етапу.

What command was entered: Яка команда була введена? Це рішення визначає, яку команду ввів користувач. В залежності від команди процес переходить до відповідної функції.

Function GitCommit: Виконання функції GitCommit. Ця гілка виконується, якщо введена команда GitCommit.

Function GitMerge: Виконання функції GitMerge. Ця гілка виконується, якщо введена команда GitMerge.

Function GitCheckout: Виконання функції GitCheckout. Ця гілка виконується, якщо введена команда GitCheckout.

Function GitBranch: Виконання функції GitBranch. Ця гілка виконується, якщо введена команда GitBranch.

Clear input field: Очищення поля вводу. Після виконання команди, незалежно від того, яка команда була введена, відбувається очищення поля вводу для подальшого введення нових команд.

Основним об`єктом моєї роботи кнопка з класом Bublle, який унаслідований від класу Button і має деякі додаткові властивості та методи для роботи з кнопкою.

Клас Bublle:

Опис конструкторів: Клас має два конструктори. Перший конструктор Bublle() встановлює деякі властивості кнопки за замовчуванням, такі як FlatStyle і BorderSize. Другий конструктор Bublle() приймає параметри для налаштування тексту, кольору фону, кольору тексту, розміру кнопки та деяких додаткових

властивостей.

Властивості: Клас має властивість ButtonColor для зберігання кольору кнопки. Також в конструкторі можна налаштувати текст, колір фону, колір тексту, розмір кнопки та інші властивості.

Цей метод перевизначає метод OnPaint базового класу Button і використовується для малювання кнопки у вигляді еліпса. Використовується GraphicsPath для створення шляху для еліпса, а потім цей шлях встановлюється як область кнопки.

В конструкторі другого класу є локальні змінні для зберігання номеру "гілки" (branch) та номеру "місця в кінці цієї гілки" (endbranch), а також назва "гілки" (NameBranch).

Цей клас може бути використаний для створення кнопок з еліптичною формою, які можуть бути додані до графічних інтерфейсів користувача у програмах Windows Forms. Він дозволяє налаштовувати кольори, розмір та інші властивості кнопок, що робить його корисним для розробки користувацьких інтерфейсів з власним дизайном.

Код классу наведенно нижче

```
public class Bublle : Button
{
   public Color ButtonColor;
   public Bublle()
   {
      this.FlatStyle = FlatStyle.Flat;
      this.FlatAppearance.BorderSize = 0;
   }
   public Bublle(string text, Color backgroundColor, Color textColor, Size size, string branch, int endbranch, int number)
   {
      int ThisIsEndThisBranch = endbranch;
      int bublenumber = number;
      string NameBranch = branch;
      this.Text = "A" + bublenumber ;
      this.BackColor = backgroundColor;
      this.ForeColor = textColor;
      this.Size = size;
      this.FlatStyle = FlatStyle.Flat;
```

```
            this.FlatAppearance.BorderSize = 0;
        }
        protected override void OnPaint(PaintEventArgs pevent)
        {
            GraphicsPath graphicsPath = new GraphicsPath();
            graphicsPath.AddEllipse(0, 0, ClientSize.Width, ClientSize.Height);
            this.Region = new Region(graphicsPath);
            base.OnPaint(pevent);
        }
    }
```

Цей код на C# створює динамічний список об'єктів типу `Bubble` (передбачається, що це клас користувача), додає в нього об'єкт, налаштовує його властивості, додає його на форму і прикріплює його до текстового поля за допомогою деякого валідатора. Давайте детально розберемо кожен крок.

Створення списку `Bubble`

```
List<Bubble> bubbles = new List<Bubble>();
```

Цей код створює новий порожній список об'єктів типу `Bubble` за допомогою дженерик-класу `List<T>`. `List<Bubble>` - це динамічний масив, який може збільшуватися в міру додавання нових елементів.

Додавання об'єкта `Bubble` до списку

```
bubbles.Add(new Bubble("A1", Color.Blue, Color.White, new Size(50, 50), "Main", 1, 1));
```

Цей код створює новий об'єкт `Bubble` із заданими параметрами та додає його до списку `bubbles`. Параметри, що передаються в конструктор `Bubble`, можуть бути такими:

- "A1" - ім'я або ідентифікатор міхура.
- `Color.Blue` – колір фону міхура.
- `Color.White` - колір тексту міхура.
- `New Size(50, 50)` - розмір міхура.

Встановлення якості `Location`

```
bubbles[0].Location = new Point((form.ClientSize.Width - bubbles[0].Width) / 2, 10);
```

Цей код встановлює властивість `Location` першого об'єкта `Bubble` у списку

`bubbles` так, щоб він був вирівняний по центру форми по горизонталі та на 10 пікселів від верхнього краю по вертикалі.

`form.ClientSize.Width` - ширина клієнтської області форми.

`bubbles[0]. Width` - ширина об'єкта `Bubble`.

Додавання об'єкта `Bubble` на форму

form.Controls.Add(bubbles[0]);

Цей код додає перший об'єкт `Bubble` зі списку `bubbles` до колекції елементів керування форми `form`. Це означає, що об'єкт `Bubble` відображатиметься на формі.

Прив'язка об'єкта `Bubble` до текстового поля за допомогою валідатора

TextBoxValidator.AttachValidation(textBox, bubbles[0]);

Цей код використовує якийсь клас або метод `TextBoxValidator` для прив'язки об'єкта `Bubble` до текстового поля `textBox`. Імовірно, `TextBoxValidator` - це клас користувача, який займається перевіркою або валідацією даних, що вводяться в `textBox`, залежно від стану або властивостей об'єкта `Bubble

Розглянемо наступний код:

```
public static class GitCommit
{
  public static async void GitCommitCheck(object secondbutton, object firstbutton, EventArgs e, string BranchName, int ButtonNumber)
    {
      Button ButtonFirst = (Button)firstbutton;
      Button Buttonsecond = (Button)secondbutton;
      Buttonsecond.Location = new Point(ButtonFirst.Left, ButtonFirst.Bottom); // Розташовуємо другу кнопку під першою
      Buttonsecond.Visible = false; // Встановлюємо видимість в false для початку анімації
      ((Form)ButtonFirst.Parent).Controls.Add(Buttonsecond); // Додаємо кнопку "A2" на форму
      await SmoothAppearance((Button)secondbutton);
      // Малюємо лінію між кнопками "A1" і "A2" за допомогою PictureBox
      PictureBox linePictureBox = new PictureBox();
      linePictureBox.BackColor = Color.Black; // Колір лінії
      linePictureBox.Size = new Size(2, Buttonsecond.Top - ButtonFirst.Bottom); // Розмір лінії
```

```
            linePictureBox.Location = new Point(ButtonFirst.Left + ButtonFirst.Width /
2, ButtonFirst.Bottom); // Початок лінії в центрі нижньої сторони першої кнопки
            ((Form)ButtonFirst.Parent).Controls.Add(linePictureBox); // Додаємо лінію
на форму
            async Task SmoothAppearance(Button button)
            {
               button.Visible = true; // Показуємо кнопку
               // Плавне переміщення кнопки вниз
               for (int i = 0; i <= button.Height; i += 5)
               {
                  button.Top += 1;
                  await Task.Delay(1); // Затримка для плавності анімації
               }
            }
            //Function.DeleteBranchPlace();
            //Function.CreateBranchPlace();
            TextBoxValidator.Bubbles[TextBoxValidator.Number                    -
1].ThisIsEndThisBranch = 0;
         }
      }
```

Статичний метод `GitCommitCheck`

Цей метод відповідає за створення та анімацію переміщення другої кнопки під першою кнопкою, а також за створення лінії між цими кнопками.

Параметри

- `secondbutton` (object): Об'єкт другої кнопки.

- `firstbutton` (object): Об'єкт першої кнопки.

- `e` (EventArgs): Аргументи події.

- `BranchName` (string): Ім'я гілки.

- `ButtonNumber` (int): Номер кнопки.

Процес виконання

1. Розташування кнопок:

   - Друга кнопка розташовується безпосередньо під першою кнопкою, але спочатку її видимість встановлюється в `false`.

   - Додається друга кнопка на форму.

2. Анімація появи другої кнопки:

   - Анімація плавного переміщення другої кнопки вниз здійснюється у асинхронному методі `SmoothAppearance`.

3. Малювання лінії:

- Створюється `PictureBox` для лінії, яка з'єднує першу і другу кнопку.

- Лінія додається на форму.

4. Оновлення статусу гілки:

- Змінюється значення властивості `ThisIsEndThisBranch` для останнього елемента масиву `Bubbles` у класі `TextBoxValidator`.

Цей код створює анімацію переміщення кнопок і лінії, що з'єднує їх, додаючи візуальний ефект до інтерфейсу користувача.

```
public static void SearchEndPoint(string BranchName)
{
   foreach(Bubble s in TextBoxValidator.Bubbles)
   {
      if (s.NameBranch == BranchName)
      {
         if (s.EndThisBranch == 1)
         {
            s.isHere = true;
         }
      }
   }
```

Метод SearchEndPoint - Статичний метод, який приймає один параметр `BranchName` типу `string`.

Цикл `foreach`

Проходить через кожен об'єкт типу `Bubble` в колекції `TextBoxValidator.Bubbles`.

Перевірка назви гілки

- Для кожного об'єкта `Bubble` перевіряється, чи його властивість `NameBranch` відповідає значенню `BranchName`.

Перевірка кінцевої точки гілки - Якщо умова в пункті 3 виконується, перевіряється, чи властивість `EndThisBranch` цього об'єкта дорівнює `1`

Встановлення властивості `isHere`- Якщо обидві умови виконуються, властивість `isHere` об'єкта встановлюється в `true`.

Цей метод дозволяє знайти всі об'єкти типу `Bubble` з відповідною назвою гілки, які є кінцевими точками (`EndThisBranch == 1`), і встановити їх статус як

`isHere` для подальшої обробки або візуалізації.

У цьому фрагменті коду оголошуються статичні змінні `linePictureBox`, `label` та `blackSquarePanel`, які використовуються для зберігання відповідних елементів управління. Метод `CreateBranchPlace` відповідає за створення і додавання цих елементів на форму.

```
private static PictureBox linePictureBox;
private static Label label;
private static Panel blackSquarePanel;
public static void CreateBranchPlace()
```

У цьому коді оголошено статичні змінні `linePictureBox`, `label` та `blackSquarePanel`, які використовуються для зберігання елементів керування типу `PictureBox`, `Label` та `Panel` відповідно. Метод `CreateBranchPlace` призначений для створення та додавання цих елементів на активну форму. Змінна `linePictureBox` відповідає за малювання лінії, змінна `label` використовується для відображення текстової мітки, а змінна `blackSquarePanel` відповідає за панель чорного кольору. Усі ці елементи будуть додані на форму і налаштовані відповідно до їхніх властивостей, таких як розмір, колір та розташування.`.

```
linePictureBox = new PictureBox();
label = new Label();
blackSquarePanel = new Panel();
```

У цих рядках коду створюються нові екземпляри об'єктів типу PictureBox, Label та Panel. Змінна linePictureBox ініціалізується як новий об'єкт PictureBox, який буде використовуватися для малювання лінії на формі. Змінна label ініціалізується як новий об'єкт Label, який буде використовуватися для відображення текстової мітки. Змінна blackSquarePanel ініціалізується як новий об'єкт Panel, який буде використовуватися для створення панелі чорного кольору. Ці елементи керування будуть налаштовані та додані на форму в подальшому коді.

```
blackSquarePanel.Size = new Size(50, 50);
blackSquarePanel.BackColor = Color.Black; // Колір квадрата
blackSquarePanel.Location = new Point(TextBoxValidator.Bubbles[TextBoxValidator.Number].Right + blackSquarePanel.Width, TextBoxValidator.Bubbles[TextBoxValidator.Number].Top);
```

Ці рядки коду встановлюють розмір, колір та розташування панелі `blackSquarePanel`. Властивість `Size` задає розмір панелі як 50 на 50 пікселів.

Властивість `BackColor` задає колір панелі чорним. Властивість `Location` визначає позицію панелі на формі, розташовуючи її праворуч від поточного елемента `TextBoxValidator.Bubbles` на ширину панелі і вирівнюючи по верхній границі цього елемента. Це забезпечує правильне розташування чорної квадратної панелі відносно інших елементів інтерфейсу.

```
label.Text = TextBoxValidator.activbranch + "*";
label.ForeColor = Color.Red;
label.AutoSize = true;
label.Location = new Point(blackSquarePanel.Right - 50, blackSquarePanel.Top + (blackSquarePanel.Height - label.Height) / 2);
```

Ці рядки коду встановлюють текст, колір тексту, автоматичний розмір та розташування текстової мітки `label`. Властивість `Text` задає текст мітки, використовуючи значення змінної `TextBoxValidator.activbranch` з додаванням символу `*`. Властивість `ForeColor` задає червоний колір тексту. Властивість `AutoSize` встановлюється в `true`, що дозволяє мітці автоматично підлаштовувати свій розмір під розмір тексту. Властивість `Location` визначає позицію мітки на формі: по горизонталі мітка розташовується на 50 пікселів лівіше від правої межі панелі `blackSquarePanel`, а по вертикалі — по центру висоти панелі `blackSquarePanel`, враховуючи висоту самої мітки. Це забезпечує правильне розташування текстової мітки відносно чорної квадратної панелі.

```
linePictureBox.BackColor = Color.Black; // Колір лінії
linePictureBox.Size = new Size(50, 2); // Розмір лінії
linePictureBox.Location = new Point(TextBoxValidator.Bubbles [TextBoxValidator.Number].Right, TextBoxValidator.Bubbles [TextBoxValidator.Number].Bottom - TextBoxValidator.Bubbles [TextBoxValidator.Number].Height / 2);
```

Ці рядки коду встановлюють колір, розмір та розташування елемента `linePictureBox`, який є лінією. Властивість `BackColor` задає чорний колір для лінії. Властивість `Size` встановлює розмір лінії: ширина 50 пікселів, висота 2 пікселі. Властивість `Location` визначає позицію лінії на формі: лінія розташовується від правого краю поточного елемента в масиві `TextBoxValidator.Bubbles` і на рівні, який знаходиться по вертикалі в середині висоти цього елемента. Це забезпечує правильне розташування лінії, яка з'єднує поточний елемент з іншими елементами на формі.

```
activeForm.Controls.Add(linePictureBox);
activeForm.Controls.Add(label);
activeForm.Controls.Add(blackSquarePanel);
```

Ці рядки коду додають елементи керування на активну форму. Вони додають `linePictureBox`, `label` та `blackSquarePanel` до списку елементів управління, які відображаються на формі. Це забезпечує відображення лінії, текстової мітки та чорного квадрата на інтерфейсі користувача для візуального представлення даних.

```
public void enablePanel()
{
    linePictureBox.Visible = true;
    label.Visible = true;
    blackSquarePanel.Visible = true;
}
public void disablePanel()
{
    linePictureBox.Visible = false;
    label.Visible= false;
    blackSquarePanel.Visible = false;
}

public static void CreateBranchPlace_GitBranch()
{
    label.Text += "\n" + TextBoxValidator.variableText;
    MessageBox.Show($"Триггер найден. Переменная часть текста: {TextBoxValidator.variableText}");
}
```

Метод `enablePanel()`:

Цей метод призначений для включення відображення всіх елементів управління, пов'язаних з панеллю віток. Він робить видимою лінію, текстову мітку та чорну панель, встановлюючи відповідні властивості `Visible` на значення `true`.

Метод `disablePanel()`:

Цей метод призначений для вимкнення відображення всіх елементів управління, пов'язаних з панеллю віток. Він робить невидимою лінію, текстову мітку та чорну панель, встановлюючи відповідні властивості `Visible` на значення `false`.

Метод `CreateBranchPlace_GitBranch()`:

Цей метод викликається для оновлення текстової мітки для панелі віток під

час роботи з гілками Git. Він додає новий рядок тексту до існуючого тексту мітки, який містить змінну частину тексту, що отримана з `TextBoxValidator.variableText`. Крім того, він виводить спливаюче повідомлення з текстом, що містить змінну частину тексту, яку виявлено як тригер. На рисунку 1.18 зображено початкове положення об’єктів на старті роботи симулятору:

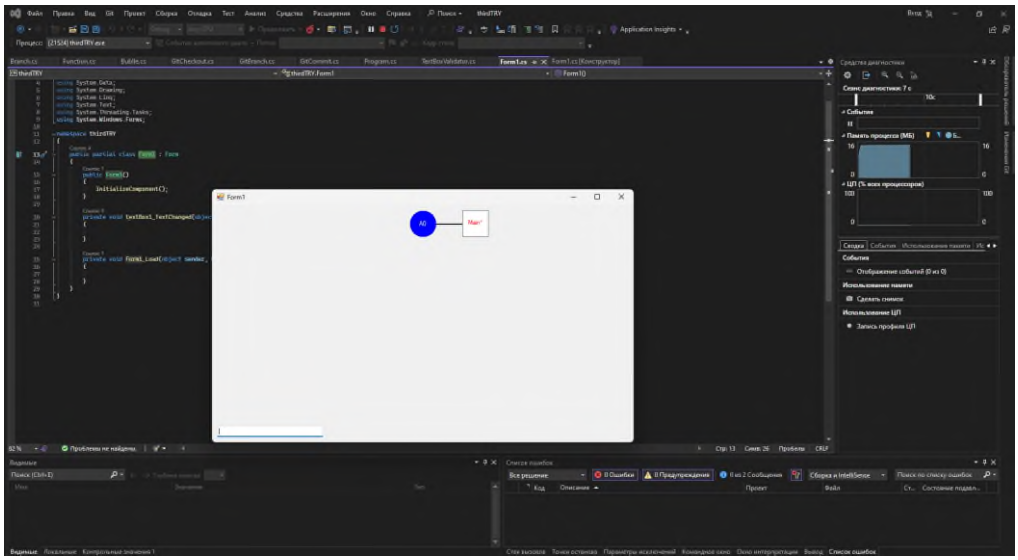


Рисунок 1.18 Початкове положення об’єктів на старті роботи симулятору

1.7 Проведення тестування впливу симулятора на якість використання Git-систем

1.7.1 Завдання для тестування

У ході експерименту з випробуванням симулятора Git шість добровольців-студентів отримали доступ до інтерактивного середовища для навчання. Кожен студент отримав завдання, які охоплювали різні аспекти використання Git, включаючи створення комітів, перехід між гілками, розв'язання конфліктів та інтеграцію з репозиторієм.

Наприклад, одне з завдань полягало в створенні нової гілки для розробки нової функціональності. Учасникам було показано, як використовувати команди Git для створення нової гілки, і потім їм було запропоновано самостійно створити гілку з конкретною назвою та описом.

Ще одне завдання передбачало розв'язання конфлікту під час злиття гілок. Учасникам було надано дві версії файлу з різними змінами, і їм потрібно було вирішити конфлікт та злити зміни в один файл, використовуючи відповідні

команди Git.

Також студентам давали завдання на виправлення помилок в існуючому коді, використовуючи функціонал Git для відновлення попередніх версій файлів та виправлення проблем.

Ці завдання допомагали студентам отримати практичний досвід у використанні Git та поглибити їх розуміння основних концепцій системи контролю версій.

Завдання 1: Зробити два комміта. Завдання має на увазі виконання якоїсь базової дії додавання нових версій файлу . На рисунку 1.1 зображено результат виконання цього завдання.

Рішенням цього завдання є двічі введення команди “git commit”.

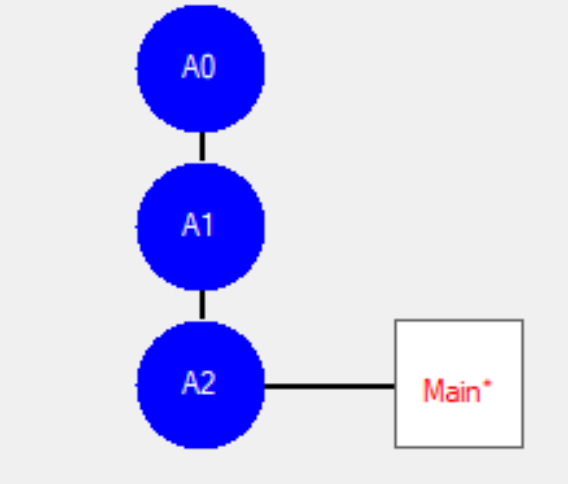


Рисунок 1.19 Результат виконання завдання 1

Завдання 2: Створити нову гілку з назвою «New» Завдання має на увазі виконання Базової дії створення нової гілки . На рисунку 1.20 зображено результат виконання цього завдання:

Рішення: Ввести команду «git branch new»

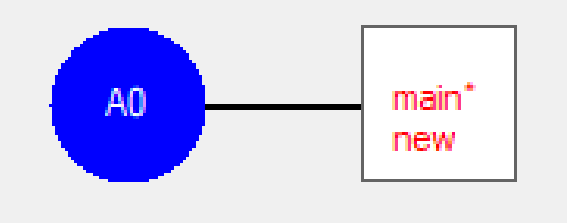


Рисунок 1.20 Результат виконання завдання 2

Завдання 3: Створити нову гілку та перейти на неї Завдання має на увазі виконання базової дії перемикання між гілками. На рисунку 1.21 зображено результат виконання цього завдання:

Рішення: Ввести команду «git branch new» Та потім команду «git checkout new»

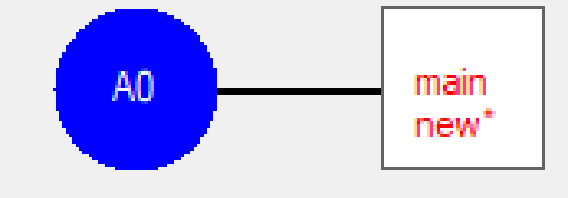


Рисунок 1.21 Результат виконання завдання 3

Завдання 4: Створити 2 гілки та зробивши 2 комміта в кожній.

Рішення: Створити нову гілку потім два комміта, повернутися в місце початку другої гілки та переходимо на першу гілку за допомогою команди «git checkout new» та зробити ще два комміти.

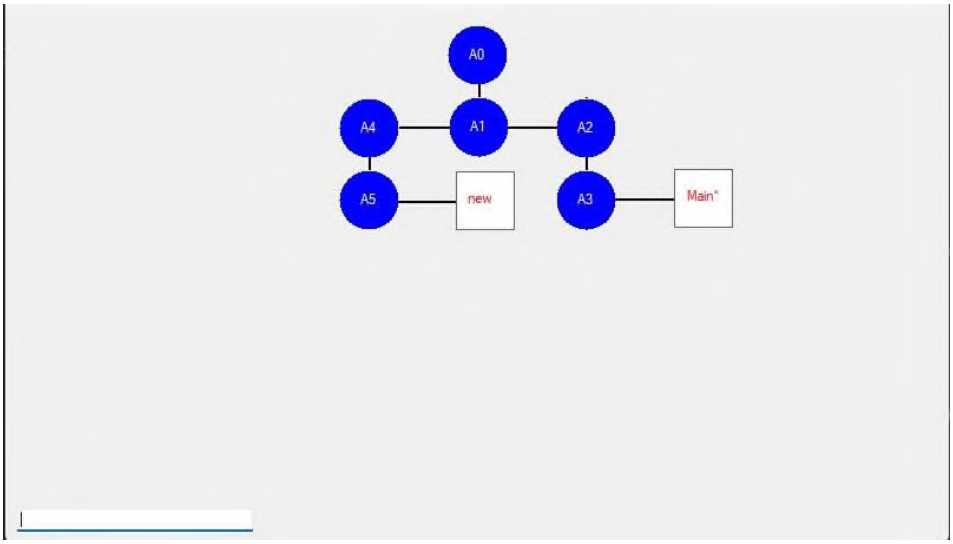


Рисунок 1.22 Результат виконання завдання 4

Це були самі базові завдання для здобування навичок користування Git. Спершу слід встановити Git та налаштувати ваше ім’я користувача і email. Після цього ініціалізуйте новий репозиторій у вашому проекті, додайте файли до індексу та зафіксуйте зміни, створивши перший коміт. Для роботи з історією комітів перегляньте її за допомогою `git log`. Створіть нову гілку для розробки нових функцій, а після завершення роботи злийте зміни назад в основну гілку. Підключіть віддалений репозиторій для спільної роботи та відправте ваші зміни до нього.

Оновлюйте локальний репозиторій останніми змінами з віддаленого репозиторію, щоб бути в курсі останніх змін. Ці кроки забезпечують фундаментальні навички для подальшої роботи з Git.

Після завершення експерименту було проведено аналіз результатів та оцінка ефективності використання симулятора Git у навчанні студентів. Результати показали, що учасники, які мали доступ до інтерактивного середовища для навчання, продемонстрували значно краще розуміння та впевненість у використанні Git порівняно зі студентами, які навчалися за допомогою традиційних методів, таких як читання документації та самостійна практика.

Наприклад, студенти, які виконали завдання на створення гілок та розв'язання конфліктів, показали більшу здатність до самостійного рішення проблем та використання необхідних команд Git. Також вони виявили більшу швидкість адаптації до нових завдань та меншу кількість помилок під час виконання завдань порівняно з традиційно навченими студентами.

Аналіз показав, що використання симулятора Git у навчанні сприяє більшому зацікавленню та активному залученню студентів, оскільки вони отримують можливість практичного використання знань в реальних ситуаціях. Крім того, інтерактивні завдання допомагають збільшити мотивацію до вивчення та підвищити ефективність навчання.

Узагальнюючи, використання симулятора Git у навчальному процесі може бути важливим кроком у розвитку студентів та покращенні їхніх навичок у використанні системи контролю версій. Такий підхід може стати основою для розробки нових методик навчання, що підвищать якість освіти та підготовку молодих спеціалістів у сфері програмування та розробки ПЗ.

Для більш наглядного розуміння результатів далі на рисунках 1.23 та 1.24 буде приведено порівняння швидкості виконання двох груп, перша що не використовує симулятор а друга що використовує .

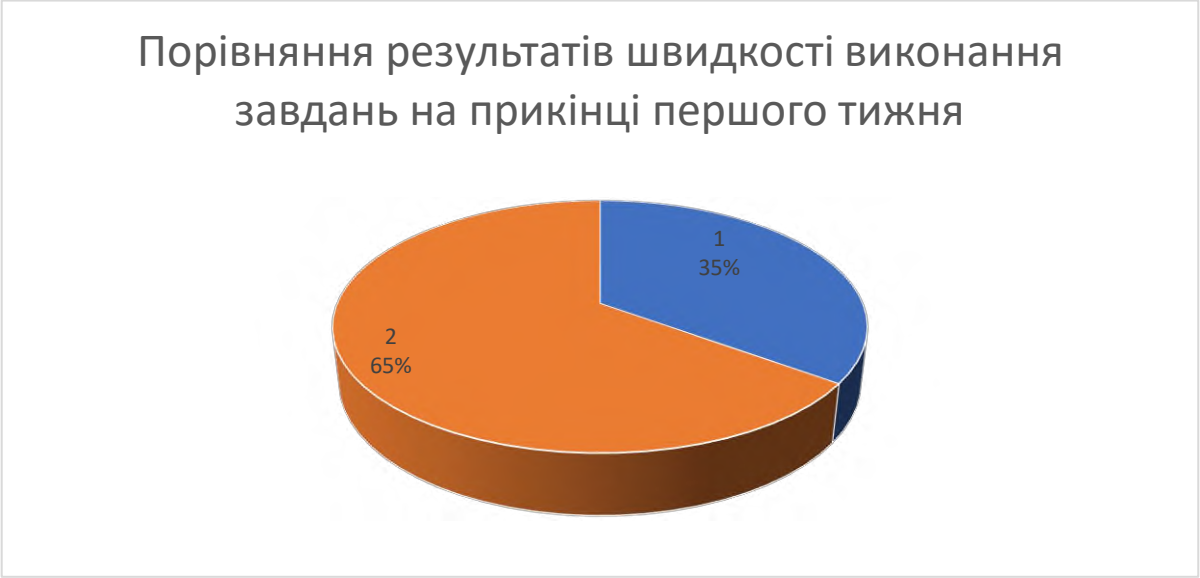


Рисунок 1.23 Діаграма Порівняння результатів швидкості виконання завдань на при кінці першого тижня

З аналізу кругової діаграми видно, що є значна різниця у швидкості виконання завдань між двома групами. Група, яка використовувала симулятор (позначена як 2), показала значно кращі результати. Це свідчить про те, що використання симулятора позитивно впливає на швидкість та ефективність виконання завдань. Такий результат дозволяє зробити висновок, що інтеграція симулятора у процес навчання або виконання завдань є корисною і може бути рекомендована для подальшого застосування з метою підвищення продуктивності.

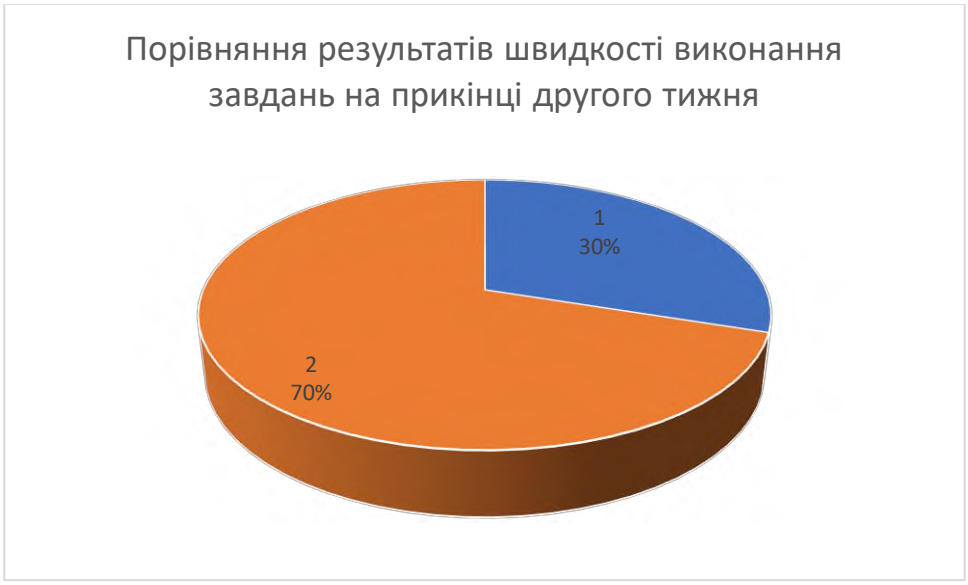


Рисунок 1.23 Діаграма Порівняння результатів швидкості виконання завдань на при кінці другого тижня

На основі діаграми, яка показує порівняння результатів швидкості виконання завдань на при кінці другого тижня, можна зробити висновок, що група, яка використовувала симулятор (70%), виконувала завдання значно швидше, ніж група без симулятора (30%). Це свідчить про те, що використання симулятора суттєво покращило ефективність та швидкість роботи користувачів, що вказує на його корисність у навчальному процесі для підвищення продуктивності та освоєння навичок роботи з Git.

1.8 Тестування та відладка помилок у симуляторі системи контролю версій

Після завершення розробки проекту я вирішив провести базове тестування проекту та виявив декілька дефектів.

Перший дефект це коли об’єкти виходять за границі вікна. Цей дефект не є критичним але його було необхідно виправити. Тому при появі ми проводимо перевірку на те чи заходять границі нового об’єкту за формую. Нижче наведенно код що вирішує нашу проблему.

```
foreach (Bubble bubble in bubbles)
{
   if (bubble.IsOutOfBounds(form))
   {
      Console.WriteLine($"Bubble at ({bubble.X}, {bubble.Y}) with diameter {bubble.Diameter} is out of bounds!");
   }
   else
   {
      bubble.PrintInfo();
   }
}
```

Також дефектом що був знайдений в процесі тестів це те що BranchPlase іноді накладувався на інші об’єкти для цього ми вводимо перевірку та якщо на місце де повинен відображатися об’єкт BranchPlase, зайнято іншим об’єктом тоді ми відображаймо його у протилежному напрямку. Далі я проведу порівняння функцій

Ці дві функції, `CreateBranchPlace` і `CreateBranchPlaceLeft`, мають схожу функціональність, але з деякими ключовими відмінностями у позиціонуванні

елементів на формі. У функції `CreateBranchPlace` квадрат (`blackSquarePanel`) розміщується праворуч від поточного елемента масиву `TextBoxValidator.Bubbles[TextBoxValidator.Number]`, тобто за координатою `Right` поточного елемента плюс ширина квадрата, а лінія (`linePictureBox`) починається справа від поточного елемента масиву і йде вправо. У функції `CreateBranchPlaceLeft` квадрат розміщується ліворуч від поточного елемента масиву за координатою `Left` поточного елемента плюс ширина квадрата, а лінія починається зліва від поточного елемента масиву і йде вліво. Обидва методи додають квадрат, мітку і лінію на форму; квадрат має фіксований розмір 50x50 пікселів, білий колір фону і чорну межу, мітка має текст "new", білий колір фону, червоний колір тексту і автоматичний розмір, а лінія має чорний колір і розмір 50x2 пікселів. Таким чином, основна відмінність між цими функціями полягає у розміщенні квадрата і лінії відносно поточного елемента масиву `TextBoxValidator.Bubbles[TextBoxValidator.Number]`: одна функція розміщує їх праворуч, а інша — ліворуч.

Також було виявлено низьку інших дефектів які не були тут описані, але вони не великі та їх рішення було легким. Таким чином, хоча можуть виникати деякі невеликі проблеми з симуляторами Git, вони, як правило, не є серйозними та легко вирішуються, надаючи можливість продовжувати вивчати та вдосконалювати свої навички в користуванні Git.

2 ОХОРОНА ПРАЦІ

2.1 Вступ

Охорона праці є невід'ємною частиною виробничого процесу. Це не просто комплекс правил і наказів на робочому місці, а повноцінний комплекс заходів для забезпечення безпечної та комфортної роботи на підприємстві.

Охорона праці містить безліч аспектів таких як норми освітлення, рівня шуму, мікроклімату, розмір робочого місця і безліч інших, не менш важливих пунктів.

Нині комфорт та безпеку робітників забезпечують нормативно- правові акти, закони та положення. Адже, якщо працівник працюватиме у негативній для нього обстановці або під впливом негативних факторів і на робочому місці не забезпечуватиметься належні умови праці, тоді цей працівник буде швидше втомлюватися, допускати помилки, що може стати причиною розвитку професійної хвороби або виробничої травми.

В дипломному проєкті охорона праці розглядається з точки зору безпеки на робочому місці охоронця, що працює в приватному закладі для фіксації відвідувачів закладу з використанням персонального комп'ютеру.

2.2 Аналіз та безпека умов праці працівника на робочому місці

2.2.1 Організація робочого місця

Дипломним проектом розглядається праця охоронця у навчальному закладі приватного типу. Ця робота не вимагає будь-яких фізичних навантажень і належить до категорії 1а. Під час виконання роботи використовується персональний комп’ютер.

2.2.2 Аналіз шкідливих та небезпечних чинників

Шкідливими факторами на робочому місці цього робітника можуть стати чинникик що приведенні у таблиці 2.1:

Таблиця 2.1 Чинники шкідливих факторів

Джерело	Чиники	
	Щкідливі	Небезпечні
1.Робота за персональним комп'ютером. 2. Тестування роботи пристрою	1.Підвищена або знижена температура повітря робочої зони 2.Підвищений рівень шуму на робочому місці. 3.Підвищена або знижена рухливість повітря 4.Прямий та відбитий відблиск 5.Нервово-психічні перевантаження	1.Електричний струм

Для кожного з цих факторів повинні бути проведенні заходи для захисту робітника.

2.2.3 Організація робочого місця.

Розмір робочої поверхні столу повинен бути: для жінок 630 мм, для чоловіків 680 мм. Висота сидіння: для жінок 400 мм, для чоловіків 430мм. Ширина і глибина повинна забезпечувати виконувати роботу у зоні моторного поля інакше 600-1400мм, глибина 800-1000 мм. Простір для ніг повинен буди не менше ніж в висоту 600мм у ширину не менше 500мм, глибина не менше 450мм(на рівні коліна) на рівні витягнутої ноги не менше ніж 650мм.

Робоче сидіння повинно мати можливості повороту та підйому висоті сидіння. Висота регулювання сидіння повинна бути не менше 400-500, а глибина та ширини 400мм. Кут нахилу вперед п'ятнадцять градусів а назад п'ять градусів.

Висота спинки крісла повинна бути ненижче 300мм. Ширина не менше ніж 380мм. Кут нахилу спинки має регулюватися в межах 1-30 градусів від стандартного положення. Відстань від краю поверхні не більше ніж 400мм та не нижче ніж 260 мм.

Для забезпечення більш комфортної та здорової роботи слід використовувати підлокітники довжиною 250мм в ширину 50 мм в висоту від сидіння 230 мм а відстань між підлокітниками не менше ніж 350мм.

Сидіння та спинка повинні бути з напівм'якого матеріалу та не ковзкою. Також не повинно електризуватися.

Робоче місце має бути обладнане підставка для ніг. Ширина не менше 300 а

глибина 400мм. Та регулюватися в межах 150 мм і кутом нахилу 20 градусів. Повинна бути рифленою та з бортиками.

Природне світло на робочу поверхню повинно падати переважно з лівого боку. Монітор повинен бути розташований до очей користувача на відстані у 600-700 мм. Розташування монітору повинно бути зручним, 30 градусів від стандартного рівня зору.

Клавіатуру слід розташувати на відстані 100-150 мм від краю робочої поверхні. Клавіатура повинна бути під нахилом 5-15 градусів до робітника.

2.2.4 Мікроклімат

Згідно до ГОСТ 12.1.005-88, СН 4088-86. мікроклімат повинен відповідати нормативам, які приведені у таблиці 2.2:

Таблиця 2.2 Нормативи мікроклімату

Пора року	Температура повітря у градусах Цельсія.	Відносна вологість повітря у відсотках.	Швидкість руху повітря у метрах за секунду.
Холодна	22-24	40-60	0,1
Тепла	23-25	40-60	0,1

Для підтримки в приміщенні нормального, що відповідає гігієнічним вимогам, складу повітря, видалення з нього шкідливих речовин використовують вентиляцію. При природній вентиляції (за допомогою вікон) повітря надходить у приміщення і видаляється внаслідок різниці температур. Але вона має низку недоліків. Тому у приміщенні застосовується штучна, загально обмінна вентиляція, яка очищає повітря і направляє його до робочого місця. Повітря, перед його споживанням можна піддати обробці: підігріти, зволожити, охолодити тощо.

Рівень іонів у повітрі що повинні відповідати санітарно-гігієнічним нормам № 2152-80. Наведені у таблиці 2.3:

Таблиця 2.3 Стандарти рівня іонів у повітрі

Рівні	Кількість на сантиметр кубічний.	
	n+	n-
Мінімальне	400	600
Оптимальне	1500-300	3000-5000
Максимальне	50000	50000

Для підтримки мікроклімату слід використовувати кондиціонери та зволожувачі повітря або інші прилади.

2.2.5 Освітлення

У приміщенні де буде застосовуватися персональний комп'ютер повинно бути штучне та природне освітлення. Природне освітлення повинно бути переважно з північної сторони та забезпечувати коефіцієнт освітлення не нижче ніж 1,5%. Штучне освітлення повинно бути системою загального рівномірного освітлення.

Освітлення на робочій поверхні повинно бути 300-500лк. Якщо загальна система не може забезпечити цей показник то додаткова освітлення може забезпечити місцевими світильниками. Вони повинні не створювати відблиск та освітлення екранну не повинно бути вище ніж 300лк.

У разі використовувані штучного освітлення слід використовувати люмінесцентні лампи. Допускається використання металогалогенних ламп потужністю 250Вт. Також допускається використовування ламп розжарювання для місцевих світильників. Використовування будь яких світильників без розсіювачів заборонено.

2.2.6 Електробезпека

Повинен використовуватися нульовий захисний провідник для заземлення. Усі провідники повинні відповідати параметрам мережі та навантаження. Якщо у приміщенні використовується 5 або більше комп'ютерів або периферійних пристроїв, тоді повинен бути встановлений на помітному та легкодоступному місці аварійний вимикач. Розетки повинні бути справні та відповідати певним нормам та характеристикам.

2.3 Пожежна безпека

Пожежна безпека входить в комплекс заходів з охорони праці, і організаційна робота в цій сфері на об'єктах господарювання включає широкий спектр заходів, а саме:

- створення умов для безпечної праці,
- мінімізації ризику виникнення пожеж,

- своєчасне і повноцінне забезпечення технічними засобами для запобігання займання та усунення самих пожеж та їх наслідків,
- контроль дотримання протипожежних вимог і норм законодавства,
- розробка і впровадження регламентів по гасінню пожеж, евакуації та порятунку з місць пожежі й задимлення людей і майна (матеріальних цінностей),
- внутрішнє і зовнішнє навчання співробітників.

Первинні засоби пожежогасіння застосовуються для боротьби з пожежами на початковій стадії. До них належать: пожежні кран-комплекти, вогнегасники, пожежний інвентар (резервуари з водою, ящики з піском, пожежні відра, лопати), а також різний переносний пожежний інструмент (кирки, сокири, багри, ломи і т. ін.).

Для гасіння пожеж промисловість випускає різні вогнегасники. Найбільшого поширення набули водопінні, водяні, газові (вуглекислотні) і порошкові. За ефективністю пожежогасіння гасіння, економічністю та іншими показниками більш перспективними вважаються порошкові вогнегасники.

Первинні засоби пожежогасіння розміщують на пожежних щитах, які встановлюють на виробничій території з розрахунку один щит на 5000 м2. Вони фарбуються у червоний колір. Вогнегасники маркують буквами, що означає їх вид, та цифрами, що означають їх об’єм

2.4 Висновки

Охорона праці - це не маловажна частина при забезпечені безпеки праці на даний час. Для того щоб працівники були здорові та мали комфортне місце роботи створено велику кількість документів, норм, заходів та інших, але не менш важливих засобі для забезпечення охорони праці. З кожним роком на робочому місці кількість загиблих або постраждавши зменшується. Але до нульових показників ще далеко. Тому всі заходи змінюються та знаходиться багато нових рішень для забезпечення комфортної та безпечної праці.

ВИСНОВКИ

В результаті виконання бакалаврської кваліфікаційної роботи були досягнуті ключові цілі, що були визначені на етапі постановки проекту. Розроблене програмне забезпечення виявилося дієвим і вдало забезпечує процес навчання користувачів взаємодії з Git-системами. Це відчутно підвищує їхні професійні навички та ефективність у вирішенні завдань, пов'язаних з управлінням версіями коду та спільною роботою над проектами.

Застосування платформи GitHub в ролі інструменту для спільної роботи та автоматизації процесів стало ключовим чинником в оптимізації процесу розробки програмного забезпечення. Це дозволило ефективно управляти проектом, відстежувати зміни в коді, забезпечувати якість та прозорість у співпраці між учасниками команди розробників. Висока якість кінцевого продукту, яка досягається завдяки використанню GitHub, відображається в його надійності та готовності до використання.

Отже, ця бакалаврська кваліфікаційна робота підкреслює важливість освоєння сучасних інструментів розробки програмного забезпечення та надає конкретні рекомендації для підвищення рівня навичок у галузі IT. Вона демонструє, що використання відповідних інструментів може значно поліпшити якість роботи розробників та якість створеного продукту.

ПЕРЕЛІК ВИКОРИСТАНИХ ІНФОРМАЦІЙНИХ ДЖЕРЕЛ

C# 9.0 in a Nutshell: The Definitive Reference - Автори: Joseph Albahari, Ben Albahari

Pro C# 9 with .NET 5: Foundational Principles and Practices in Programming - Автор: Andrew Troelsen, Philip Japikse

C# 8.0 and .NET Core 3.0 – Modern Cross-Platform Development - Автор: Mark J. Price

Head First C#: A Learner's Guide to Real-World Programming with C# and .NET Core - Автори: Andrew Stellman, Jennifer Greene

C# Programming for Absolute Beginners - Автор: Radek Vystavěl

CLR via C# - Автор: Jeffrey Richter

C# 7.0 in a Nutshell: The Definitive Reference - Автори: Joseph Albahari, Ben Albahari

Essential C# 8.0 - Автор: Mark Michaelis

Pro Git - Автор: Scott Chacon, Ben Straub

Git Pocket Guide: A Working Introduction - Автор: Richard E. Silverman

Version Control with Git: Powerful tools and techniques for collaborative software development - Автори: Jon Loeliger, Matthew McCullough

Git for Teams: A User-Centered Approach to Creating Efficient Workflows in Git - Автор: Emma Jane Hogbin Westby

Git in Practice: Includes 66 Techniques - Автори: Mike McQuaid

Learn Git in a Month of Lunches - Автор: Rick Umal

Pragmatic Version Control Using Git - Автор: Travis Swicegood

ДОДАТОК А

Класи Bublle, Branch, TextBoxValidator

```
using System;
using System.Drawing;
using System.Drawing.Drawing2D;
using System.Windows.Forms;
using thirdTRY;

public class Bublle : Button
{
   public int bublenumber;
   public Color ButtonColor;
   public int EndThisBranch;
   public string NameBranch;
   public bool isHere;
   public Branch brranch = new Branch();
   public Bublle()
   {
      this.FlatStyle = FlatStyle.Flat;
      this.FlatAppearance.BorderSize = 0;
   }

   public Bublle(string text, Color backgroundColor, Color textColor, Size size, string branch, int endbranch, int number,
Form form)
   {
      EndThisBranch = endbranch;
      ChangeEndBranche(endbranch);
      bublenumber = number;
      NameBranch = branch;
      this.Text = "A" + bublenumber;
      this.BackColor = backgroundColor;
      this.ForeColor = textColor;
      this.Size = size;
      this.FlatStyle = FlatStyle.Flat;
      this.FlatAppearance.BorderSize = 0;
      //brranch.CreateBranchPlace(form);
   }

   public void CreatePlace(Form form)
   {
      brranch.CreateBranchPlace(form);
   }

   protected override void OnPaint(PaintEventArgs pevent)
   {
      GraphicsPath graphicsPath = new GraphicsPath();
      graphicsPath.AddEllipse(0, 0, ClientSize.Width, ClientSize.Height);
      this.Region = new Region(graphicsPath);
      base.OnPaint(pevent);
   }

   public void Visiblee()
   {
      brranch.enablePanel();

   }

   public void UnVisible()
   {
      brranch.disablePanel();
```

```
    }

    public void ChangeEndBranche(int isEnd)
    {
        EndThisBranch = isEnd;
        if (EndThisBranch == 1)
        {
            Visiblee();
        }
        else
        {
            UnVisible();
        }
    }
}
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace thirdTRY
{
    public class Branch
    {
        private static PictureBox linePictureBox =  new PictureBox();
        private static Label label = new Label();
        private static Panel blackSquarePanel = new Panel();
        //Form activeForm = Form.ActiveForm;
        public void CreateBranchPlace(Form activeForm)
        {
            blackSquarePanel.Size = new Size(50, 50);
            blackSquarePanel.BackColor = Color.White; // Цвет квадрата
            blackSquarePanel.BorderStyle = BorderStyle.FixedSingle;
            //MessageBox.Show("kjkdjf{}" + TextBoxValidator.Number);
            blackSquarePanel.Location = new Point(TextBoxValidator.Bubbles[TextBoxValidator.Number].Right +
blackSquarePanel.Width, TextBoxValidator.Bubbles[TextBoxValidator.Number].Top);
            label.Text = "new";
            label.BackColor = Color.White;
            label.ForeColor = Color.Red;
            label.AutoSize = true;
            label.Location = new Point(blackSquarePanel.Right - 42, blackSquarePanel.Top + ((blackSquarePanel.Height -
label.Height) + 5) / 2);

            linePictureBox.BackColor = Color.Black; // Цвет линии
            linePictureBox.Size = new Size(50, 2); // Размер линии
            linePictureBox.Location = new Point(TextBoxValidator.Bubbles[TextBoxValidator.Number].Right,
TextBoxValidator.Bubbles[TextBoxValidator.Number].Bottom -
TextBoxValidator.Bubbles[TextBoxValidator.Number].Height / 2);
            activeForm.Controls.Add(linePictureBox);
            activeForm.Controls.Add(label);
            activeForm.Controls.Add(blackSquarePanel);
        }
        public void CreateBranchPlaceLeft(Form activeForm)
        {
            blackSquarePanel.Size = new Size(50, 50);
            blackSquarePanel.BackColor = Color.White; // Цвет квадрата
            blackSquarePanel.BorderStyle = BorderStyle.FixedSingle;
            blackSquarePanel.Location = new Point(TextBoxValidator.Bubbles[TextBoxValidator.Number].Left +
blackSquarePanel.Width, TextBoxValidator.Bubbles[TextBoxValidator.Number].Top);
            label.Text = "new";
            label.BackColor = Color.White;
```

```
            label.ForeColor = Color.Red;
            label.AutoSize = true;
            label.Location = new Point(blackSquarePanel.Right - 42, blackSquarePanel.Top + ((blackSquarePanel.Height -
label.Height) + 5) / 2);
            linePictureBox.BackColor = Color.Black; // Цвет линии
            linePictureBox.Size = new Size(50, 2); // Размер линии
            linePictureBox.Location = new Point(TextBoxValidator.Bubbles[TextBoxValidator.Number].Left - 50,
TextBoxValidator.Bubbles[TextBoxValidator.Number].Bottom -
TextBoxValidator.Bubbles[TextBoxValidator.Number].Height / 2);
            activeForm.Controls.Add(linePictureBox);
            activeForm.Controls.Add(label);
            activeForm.Controls.Add(blackSquarePanel);
        }
        public void DeleteBranchPlace()
        {
            Form activeForm = (Form)TextBoxValidator.Bubbles[TextBoxValidator.Number].Parent;

            // Проверяем и удаляем элементы, если они существуют
            if (linePictureBox != null)
            {
                activeForm.Controls.Remove(linePictureBox);
                linePictureBox.Dispose();
                linePictureBox = null;
            }

            if (label != null)
            {
                activeForm.Controls.Remove(label);
                label.Dispose();
                label = null;
            }

            if (blackSquarePanel != null)
            {
                activeForm.Controls.Remove(blackSquarePanel);
                blackSquarePanel.Dispose();
                blackSquarePanel = null;
            }
        }

        public void enablePanel()
        {
            linePictureBox.Visible = true;
            label.Visible = true;
            blackSquarePanel.Visible = true;
        }
        public void disablePanel()
        {
            linePictureBox.Visible = false;
            label.Visible= false;
            blackSquarePanel.Visible = false;
        }

        public void CreateBranchPlace_GitBranch()
        {
            label.Text += "\n" + TextBoxValidator.variableText;
            MessageBox.Show($"Триггер найден. Переменная часть текста: {TextBoxValidator.variableText}");
        }

    }
}
using System;
using System.Collections.Generic;
using System.Drawing;
```

```
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace thirdTRY
{
   public class TextBoxValidator
   {
      public static string variableText = "";
      public static int Number { get; set; } = 0;
      public static List<Bublle> Bubbles { get; set; } = new List<Bublle>();
      public static string activbranch = "Main";
      struct ArrayArrayev
      {
         public string MyString;
         public int MyInt;
      }
      public static void IncrementNumber(Form form, KeyPressEventArgs e, string BranchName)
      {
         Number++;
         TextBoxValidator.CreateBublle(form);
         GitCommit.GitCommitCheck(Bubbles[Number], Bubbles[Number - 1], e, BranchName, Number + 1,form);
         //TextBoxValidator.Bubbles[TextBoxValidator.Number].CreatePlace(form);
      }
      public static void CreateBublle(Form form)
      {
         Bubbles.Add(new Bublle("A1", Color.Blue, Color.White, new Size(50, 50), activbranch, 1, Number, form));
      }


      public static void AttachValidation(TextBox textBox, Button button, Form form)
      {
         string BranchName = activbranch;
         textBox.KeyPress += (sender, e) =>
         {
            // Проверяем нажатую клавишу
            if (e.KeyChar == (char)Keys.Enter)
            {
               string text = textBox.Text;
               // Проверяем текст на определенное условие
               if (text.Contains("git commit"))
               {
                  Bubbles[Number].ChangeEndBranche(0);
                  TextBoxValidator.IncrementNumber(form, e, BranchName);
                  //GitCommit.GitCommitCheck(Bubbles[Number], Bubbles[Number - 1], e, BranchName, Number +
1,form);
                  // Передаем кнопку в GitCommitCheck
                  // MessageBox.Show("Нельзя использовать это слово!");
                  textBox.Text = ""; // Очищаем текст в TextBox
               }
               /*if (text.Contains("GitBranch"))
               {

                  GitBranch.GitBranchDo(); // Передаем кнопку в GitCommitCheck
                                                                  // MessageBox.Show("Нельзя использовать это
слово!");
                  textBox.Text = ""; // Очищаем текст в TextBox
               }*/

               if (text.Contains("git branch "))
               {
                  // Выделите остальную часть текста в качестве переменной
                  variableText = text.Replace("git branch ", "");
```

```
                GitBranch.GitBranchDo();
            }
            if (text.Contains("git checkout "))
            {
                // Выделите остальную часть текста в качестве переменной
                variableText = text.Replace("git checkout ", "");
                GitBranch.GitBranchDo();
            }
        }
    };
  }
 }


}


using System;
using System.Drawing;
using System.Windows.Forms;
using System.Threading.Tasks;
using System.Security.Cryptography.X509Certificates;
using System.Runtime.InteropServices;

namespace thirdTRY
{
  public static class GitCommit
  {
    public static async void GitCommitCheck(object secondbutton, object firstbutton, EventArgs e, string
BranchName,int ButtonNumber, Form form)
    {
      Button ButtonFirst = (Button)firstbutton;
      Button Buttonsecond = (Button)secondbutton;
      Buttonsecond.Location = new Point(ButtonFirst.Left, ButtonFirst.Bottom); // Помещаем вторую кнопку внизу
первой
      Buttonsecond.Visible = false; // Устанавливаем видимость в false для начала анимации
      ((Form)ButtonFirst.Parent).Controls.Add(Buttonsecond); // Добавляем кнопку "A2" на форму

      await SmoothAppearance((Button)secondbutton);

      // Рисуем линию между кнопками "A1" и "A2" с помощью PictureBox
      PictureBox linePictureBox = new PictureBox();
      linePictureBox.BackColor = Color.Black; // Цвет линии
      linePictureBox.Size = new Size(2, Buttonsecond.Top - ButtonFirst.Bottom); // Размер линии
      linePictureBox.Location = new Point(ButtonFirst.Left + ButtonFirst.Width / 2, ButtonFirst.Bottom); // Начало
линии в центре нижней стороны первой кнопки
      ((Form)ButtonFirst.Parent).Controls.Add(linePictureBox); // Добавляем линию на форму

      async Task SmoothAppearance(Button button)
      {
        button.Visible = true; // Показываем кнопку

        // Плавное перемещение кнопки вниз
        for (int i = 0; i <= button.Height; i += 5)
        {
          button.Top += 1;
          await Task.Delay(1); // Задержка для плавности анимации
        }
      }
      TextBoxValidator.Bubbles[TextBoxValidator.Number].CreatePlace(form);
      //Function.DeleteBranchPlace();
      //Function.CreateBranchPlace();
      //TextBoxValidator.Bubbles[TextBoxValidator.Number - 1].ThisIsEndThisBranch = 0;
```

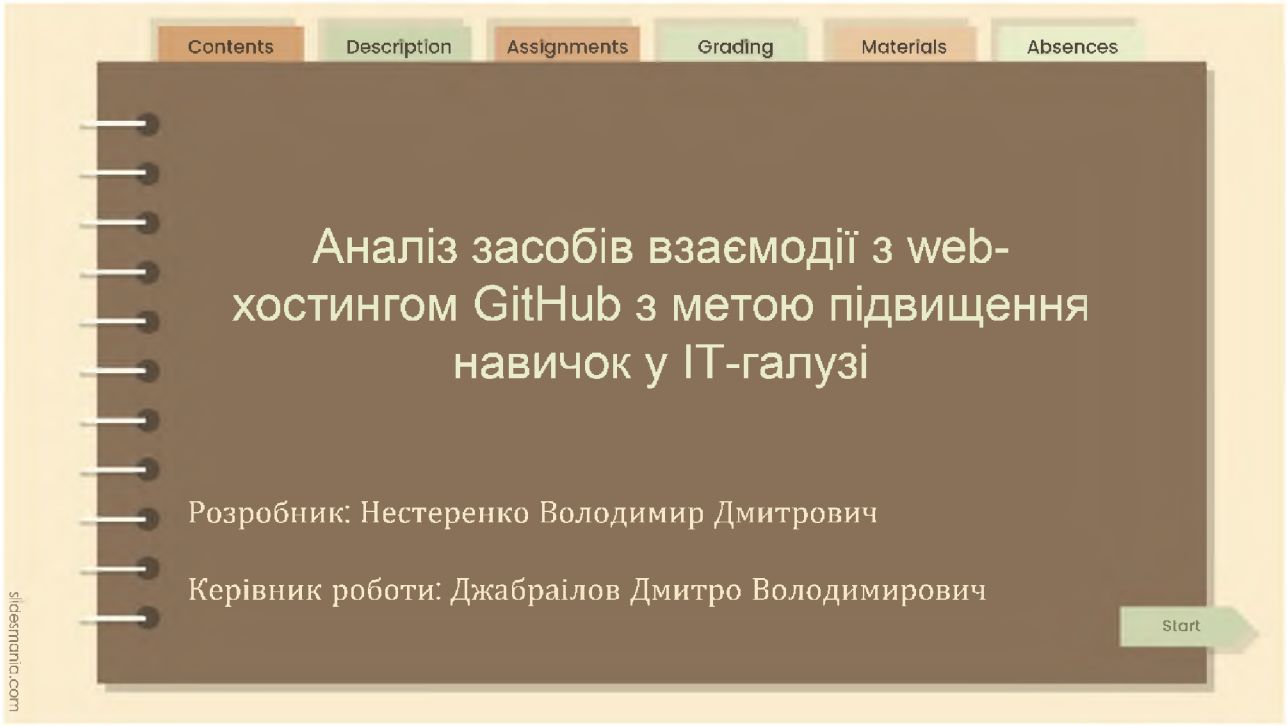
```
        }


    }
}

using System;
using System.Collections.Generic;
using System.Drawing;
using System.Linq;
using System.Reflection.Emit;
using System.Threading.Tasks;
using System.Windows.Forms;
using Label = System.Windows.Forms.Label;

namespace thirdTRY
{
    public class Function
    {


        private static PictureBox linePictureBox;
        private static Label label;
        private static Panel blackSquarePanel;

        public static void CreateBranchPlace()
        {
            Form activeForm = (Form)TextBoxValidator.Bubbles[TextBoxValidator.Number].Parent;

            // Инициализируем элементы управления и сохраняем ссылки
            linePictureBox = new PictureBox();
            label = new Label();
            blackSquarePanel = new Panel();

            blackSquarePanel.Size = new Size(50, 50);
            blackSquarePanel.BackColor = Color.Black; // Цвет квадрата
            blackSquarePanel.Location = new Point(TextBoxValidator.Bubbles[TextBoxValidator.Number].Right +
blackSquarePanel.Width, TextBoxValidator.Bubbles[TextBoxValidator.Number].Top);

            label.Text = TextBoxValidator.activbranch + "*";
            label.ForeColor = Color.Red;
            label.AutoSize = true;
            label.Location = new Point(blackSquarePanel.Right - 50, blackSquarePanel.Top + (blackSquarePanel.Height -
label.Height) / 2);

            linePictureBox.BackColor = Color.Black; // Цвет линии
            linePictureBox.Size = new Size(50, 2); // Размер линии
            linePictureBox.Location = new Point(TextBoxValidator.Bubbles[TextBoxValidator.Number].Right,
TextBoxValidator.Bubbles[TextBoxValidator.Number].Bottom -
TextBoxValidator.Bubbles[TextBoxValidator.Number].Height / 2);

            // Добавляем элементы на форму
            activeForm.Controls.Add(linePictureBox);
            activeForm.Controls.Add(label);
            activeForm.Controls.Add(blackSquarePanel);
        }

        public static void DeleteBranchPlace()
        {
            Form activeForm = (Form)TextBoxValidator.Bubbles[TextBoxValidator.Number].Parent;

            // Проверяем и удаляем элементы, если они существуют
            if (linePictureBox != null)
            {
```

```
            activeForm.Controls.Remove(linePictureBox);
            linePictureBox.Dispose();
            linePictureBox = null;
        }

        if (label != null)
        {
            activeForm.Controls.Remove(label);
            label.Dispose();
            label = null;
        }

        if (blackSquarePanel != null)
        {
            activeForm.Controls.Remove(blackSquarePanel);
            blackSquarePanel.Dispose();
            blackSquarePanel = null;
        }
    }
    public static void CreateBranchPlace_GitBranch()
    {
        label.Text += "\n"+ TextBoxValidator.variableText ;
        MessageBox.Show($"Триггер найден. Переменная часть текста: {TextBoxValidator.variableText}");
    }


    public static void SearchEndPoint(string BranchName)
    {
    foreach(Bublle s in TextBoxValidator.Bubbles)
        {
            if (s.NameBranch == BranchName)
            {
                if(s.EndThisBranch == 1)
                {
                    s.isHere = true;

                }
            }
        }
    }
  }



}
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Linq;
using System.Reflection.Emit;
using System.Threading.Tasks;
using System.Windows.Forms;
using Label = System.Windows.Forms.Label;

namespace thirdTRY
{
  internal static class Program
  {
    private static Button buttonA2; // Объявляем кнопку A2 как глобальную переменную
    private static PictureBox linePictureBox; // Объявляем PictureBox для линии
    /// <summary>
    /// Главная точка входа для приложения.
    /// </summary>
    [STAThread]
```

```
        static void Main()
        {


            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            PictureBox linePictureBox = new PictureBox();
            Form1 form = new Form1(); // Создаем экземпляр формы

            // Создаем экземпляр текстового поля (TextBox)
            TextBox textBox = new TextBox();
            textBox.Size = new Size(200, 20); // Устанавливаем размер текстового поля
            textBox.Location = new Point(10, form.ClientSize.Height - textBox.Height - 10); // Устанавливаем
расположение текстового поля
            form.Controls.Add(textBox); // Добавляем текстовое поле на форму

            // Применяем валидацию к текстовому полю
            // Применяем валидацию к текстовому полю
            //List<Bublle> bubbless = new List<Bublle>();
            System.Windows.Forms.Label label = new Label();
            TextBoxValidator.CreateBublle(form);
            TextBoxValidator.Bubbles[TextBoxValidator.Number].Location = new Point((form.ClientSize.Width -
TextBoxValidator.Bubbles[TextBoxValidator.Number].Width) / 2, 10);
            form.Controls.Add(TextBoxValidator.Bubbles[0]); // Добавляем кнопку на форм
            TextBoxValidator.AttachValidation(textBox, TextBoxValidator.Bubbles[0], form);
            TextBoxValidator.Bubbles[TextBoxValidator.Number].CreatePlace(form);
            //Function.CreateBranchPlace();

            Application.Run(form); // Запускаем приложение с этой формой

        }


    }
}
```

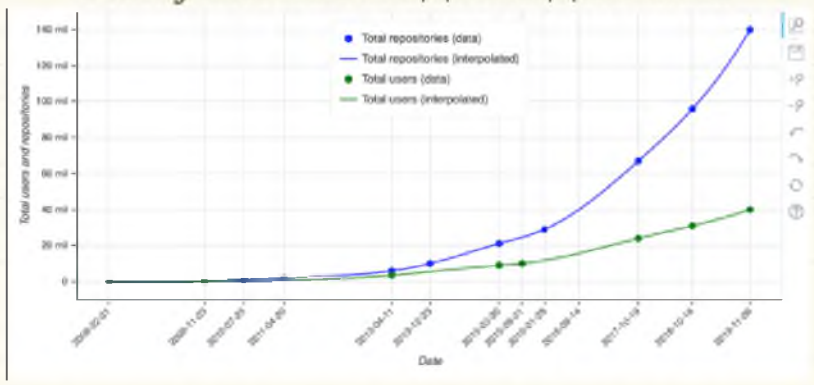
ДОДАТОК Б

Слайди мультимедійної презентації



Contents | Description | Assignments | Grading | Materials | Absences

Актуальність дослідження



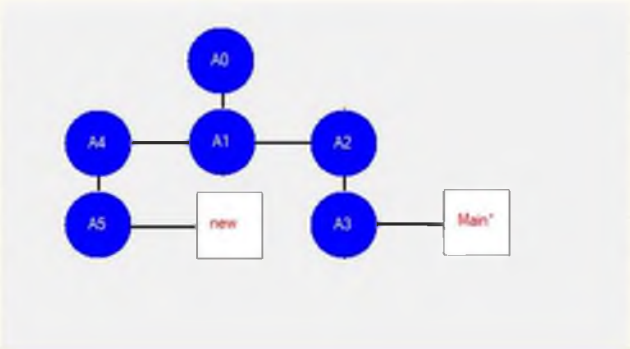
Графік зростання кількості користувачів GitHub

Contents | Description | Assignments | Grading | Materials | Absences

Огляд технологій

slidesmania.com

Розробка програмного забезпечення



Contents Description Assignments Grading Materials Absences

Тестування та налагодження – Завдання

Для тестування було обрано 6 студенті та розділені на дві групи. Ті що вивчали Git-системи за допомогою тренежера, та ті що без нього

slidesmania.com

Тестування та налагодження – Хід тестування

Після завершення експерименту було проведено аналіз результатів та оцінка ефективності використання симулятора Git у навчанні студентів.

Contents Description Assignments Grading Materials Absences

Аналіз ефективності використання ПЗ

Порівняння результатів швидкості виконання завдань на прикінці першого тижня

Порівняння результатів швидкості виконання завдань на при кінці другого тижня

slidesmania.com

Відео презентація проєкту

Contents Description Assignments Grading Materials Absences

Висновки

У висновках хочу підсумувати виконану роботу. Ми досягли основних цілей та завдань, поставлених на початку проекту. Розроблене програмне забезпечення дозволяє ефективно навчати користувачів взаємодії з Git-системами, що сприяє підвищенню їх професійних навичок та продуктивності. Використання GitHub як платформи для спільної роботи та автоматизації процесів дозволяє оптимізувати розробку програмного забезпечення, забезпечуючи високу якість кінцевого продукту. Ця робота підтверджує важливість освоєння сучасних інструментів для розробки програмного забезпечення та надає конкретні рекомендації для покращення навичок роботи в IT-галузі

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

ВІДГУК

керівника на кваліфікаційну роботу бакалавра здобувача (здобувачки) освіти відділення комп'ютерних систем

Нестеренко Володимира Дмитровича

(прізвище, ім`я та по батькові)

Спеціальність: 123 "Комп'ютерна інженерія"

Освітня програма: «Комп'ютерна інженерія»

Тема дипломного проекту: Аналіз засобів взаємодії з web-хостингом GitHub з метою підвищення навичок у IT-галузі

ХАРАКТЕРИСТИКА ДИПЛОМНОГО ПРОЕКТУ

а) обсяг і якість виконання проекту (графічного матеріалу і розрахунково-пояснювальної записки) Кваліфікаційну роботу бакалавра виконано відповідно технічному завданню. Пояснювальна записка містить 61 сторінок. У пояснювальній записці виконано опис предметної області, проведено аналіз засобів взаємодії з web-хостингом GitHub. Проведено проектування та реалізація симулятора роботи з GitHub. Графічна частина складається з 10 слайдів мультимедійної презентації, які передбачені технічним завданням. Якість виконання пояснювальної записки та графічної частини добра, розробку виконано в повному обсязі.

б) самостійність роботи над проектом: Протягом всього строку дипломного проектування та переддипломної практики здобувач освіти Нестеренко В.Д. поступово та послідовно виконував всі етапи розробки. Всі роботи здобувач освіти виконував самостійно, з оглядом на рекомендації керівника.

в) теоретична підготовка випускника (випускниці): Здобувач освіти Нестеренко В.Д. під час роботи над дипломним проектом вивчив достатню кількість літературних джерел та матеріалів за даною тематикою.

Вважаю, що теоретична підготовка дипломника добра і він готовий до захисту дипломного проекту

г) вміння розв'язувати виробничі та конструкторські питання

Під час дипломного проектування здобувач освіти Нестеренко В.Д. мав змогу самостійно приймати рішення з реалізації симулятор роботи із GitHub, питань тестування його ефективності, та показав вміння організовано працювати над поставленим завданням, складати схеми та проводити розробку коду за допомогою актуальних для теми комп'ютерних програмних засобів.

Оцінка розрахункової частини Відмінно

Оцінка графічної частини Відмінно

Загальна оцінка Відмінно

Прізвище, ім'я, по батькові керівника дипломного проекту

Джабраілов Дмитро Володимирович

Місце роботи і посада керівника дипломного проекту

ВСП "Одеський технічний фаховий коледж ОНТУ", викладач кафедри комп'ютерної інженерії

Підпис

«13» 06 2024 р.

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

РЕЦЕНЗІЯ

на кваліфікаційну роботу бакалавра здобувача освіти

відділення комп'ютерних систем

Нестеренко Володимира Дмитровича

(прізвище, ім'я та по батькові)

Спеціальність 123 "Комп'ютерна інженерія"

Освітня програма «Комп'ютерна інженерія»

Керівник дипломного проекту (роботи) Джабраілов Дмитро Володимирович

(прізвище, ім'я та по батькові)

Тема дипломного проекту (роботи) *Аналіз засобів взаємодії з web-хостингом GitHub з метою підвищення навичок у IT-галузі*

Обсяг розрахунково-пояснювальної записки 61 сторінок

Обсяг графічної (презентаційної) частини 10 аркушів (слайдів)

ХАРАКТЕРИСТИКА ДИПЛОМНОГО ПРОЕКТУ (РОБОТИ)

а) заключення про ступінь відповідності виконаного дипломного проекту (роботи) завданню

Представлена на рецензію кваліфікаційна робота бакалавра повністю відповідає меті проектування та технічному завданню. Тематика дипломного проекту є актуальною для своєї галузі та присвячена питанням засобів взаємодії з web-хостингом GitHub.

б) характеристика виконання кожного розділу дипломного проекту (роботи)

Кваліфікаційна робота складається зі вступу, двох розділів, висновків, переліку використаних джерел. У основному розділі розглянуті питання проблематики взаємодії з web-хостингом GitHub, сформовано вимоги до тестування та аналізу програми для симуляції роботи з GitHub, виконано проектування основних аспектів розробляемого ПЗ. За допомогою відповідного програмного забезпечення реалізовано тестування якості впливу програми-симулятора на навички користування GitHub.

в) оцінка якості виконання пояснювальної записки та графічної частини дипломного проекту (роботи) Графічна частина виконана на достатньо високому рівні у вигляді презентації із використанням офісного пакету Microsoft PowerPoint та Visio. Пояснювальна записка виконана охайно та у відповідності до норм оформлення документів із використанням офісного пакету Microsoft Word. Загальна якість виконання документації – добра, академічного плагіату у роботі не виявлено

г) перелік позитивних якостей дипломного проекту (роботи)

1. Детально описано мету та цілі аналізу;
2. Виконано проектування та реалізовано симулятор роботи із web-хостингом GitHub;
3. Проведено експериментальне тестування груп добровольців для визначення впливу швидкості роботи із GitHub, після використання програми симулятора.

д) основні недоліки дипломного проекту (роботи)

1. Програма-симулятор могла б реалізовувати більший набір функцій;
2. Не дуже зрозумілими є умови тестування впливу симулятора на якість використання Git-систем

Оцінка розрахункової частини Відмінно

Оцінка графічної частини Відмінно

Загальна оцінка Відмінно

Прізвище, ім'я, по батькові рецензента Царьов Роман Юрійович

Місце роботи і посада рецензента Державний університет інтелектуальних технологій і зв'язку, ст. викладач, зав. кафедри комп'ютерної інженерії та інформаційних систем

Підпис:

«___» 06 2024 р.

UNICHECK
by Turnitin

Ім'я користувача:
Катерина Григоріївна Краснокутська

ID перевірки:
1016355625

Дата перевірки:
13.06.2024 10:26:01 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
13.06.2024 10:28:36 EEST

ID користувача:
100011688

Назва документа: **2БКС-28_Нестеренко**

Кількість сторінок: **50** Кількість слів: **10742** Кількість символів: **80849** Розмір файлу: **1.57 MB** ID файлу: **1016159766**

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

1.04%
Схожість

Найбільша схожість: 0.18% з Інтернет-джерелом (https://ir.library.knu.ua/server/api/core/bitstreams/3603b454-73fa-428.

1.04% Джерела з Інтернету | 56 ..... Сторінка 52

Не знайдено джерел з Бібліотеки

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Підозріле форматування | 10 сторінок

ДОЗВІЛ
НА РОЗМІЩЕННЯ
ВИПУСКНОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ БАКАЛАВРА
В ЕЛЕКТРОННОМУ РЕПОЗИТАРІЇ ВСП «ОТФК ОНТУ»

Ми, що нижче підписалися,

Нестеренко Володимир Дмитрович,
здобувач освіти гр. 2БКС-28, та

Джабраілов Дмитро Володимирович,
керівник дипломного проекту,

не заперечуємо щодо розміщення електронного варіанту пояснювальної записки до випускного кваліфікаційної роботи бакалавра:

«Аналіз засобів взаємодії з web-хостингом GitHub з метою підвищення навичок у IT-галузі» (автор роботи – Нестеренко В.Д., керівник роботи – Джабраілов Д.В.)

виконаного у ВСП «Одеський технічний фаховий коледж Одеського національного технологічного університету» в 2024 році, у повному обсязі в електронному репозитарії ВСП «ОТФК ОНТУ» для вільного доступу через мережу Інтернет.

Несемо відповідальність за ідентичність електронного та друкованого варіантів випускної кваліфікаційної роботи бакалавра, і даємо згоду на обробку персональних даних.

Виконавець _____ / Нестеренко В.Д. /

Керівник _____ / Джабраілов Д.В./

«13» 06 2024 р.