НАЦІОНАЛЬНИЙ ТРАНСПОРТНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ТРАНСПОРТНИХ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА «ІНФОРМАЦІЙНО-АНАЛІТИЧНОЇ ДІЯЛЬНОСТІ ТА ІНФОРМАЦІЙНОЇ БЕЗПЕКИ»

Пояснювальна записка
до кваліфікаційної роботи бакалавра

на тему
«ТРАДИЦІЙНІ СУЧАСНІ ЗАСОБИ ТА ТЕХНОЛОГІЇ ПРОГРАМУВАННЯ»

Виконав: студент IV курсу, групи ІБК– IV – 1
спеціальності 122 «Комп’ютерні науки»
освітньої програми «Інформаційна безпека в комп’ютеризованих системах»

_____ Нікіта КОЛЕСНИКОВ
(підпис) (прізвище та ініціали)

Керівник: _____ Ірина ПРОКУДІНА
(підпис) (прізвище та ініціали)

Завідувач кафедри: д.т.н., проф. _____ Алі АЛЬ-АММОРІ
(науковий ступінь, вчене звання) (підпис) (прізвище та ініціали)

Київ – 2023 року

НАЦІОНАЛЬНИЙ ТРАНСПОРТНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ «ТРАНСПОРТНІ ТА ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ»
КАФЕДРА «ІНФОРМАЦІЙНО-АНАЛІТИЧНОЇ ДІЯЛЬНОСТІ ТА ІНФОРМАЦІЙНОЇ БЕЗПЕКИ»

Ступінь – бакалавр
Спеціальность 122 «Комп'ютерні науки»
Освітньо- професійна програми «Інформаційна безпека в комп'ютеризованих системах»

ЗАТВЕРДЖУЮ
Завідувач кафедри
«Інформаційно-аналітичної
діяльності та інформаційної безпеки»
_____/ Алі Аль-Амморі /
« 16 » березня 2023 року

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА

КОЛЕСНИКОВА НІКІТИ МИКОЛАЙОВИЧА

1. Тема роботи «Традиційні сучасні засоби та технології програмування»
_____
керівник проекту __Прокудіна Ірина Іванівна_____
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом закладу вищої освіти від *«16» березня 2023 року №62*
2. Строк подання студентом проекту «20 червня 2023 року»
3. Вихідні дані до проекту Глобальна та локальна аналітика даних сучасного ринку програмування.
4. Зміст пояснювальної записки (перелік питань, які потрібно розробити):
1).Розвиток технологій та засобів програмування: від минулого до сьогодення; 2).Огляд сучасних засобів та технологій програмування; 3).Порівняння сучасних засобів та технологій програмування
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень):

1).Статистика найпопулярніших фреймворків; 2).Аналіз мобільної розробки; 3).Популярність мов програмування в різних сферах використання; 4).Популярність мов програмування серед новачків.
6. Дата видачі завдання 16.01.2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Збір вихідних даних	17.01-03.02	виконано
2	Уточнення теми бакалаврської кваліфікаційної роботи, написання вступу, визначення мети роботи та постановка задач для її вирішення	03.02-24.02	виконано
3	Написання першого розділу бакалаврської кваліфікаційної роботи	25.02-12.03	виконано
4	Написання другого розділу бакалаврської кваліфікаційної роботи	13.03-04.04	виконано
5	Написання третього розділу бакалаврської кваліфікаційної роботи	05.04-18.05	виконано
6	Написання четвертого розділу бакалаврської кваліфікаційної роботи	25.05-31.05	виконано
7	Написання висновків та переліку посилань	25.05-31.05	виконано
8	Оформлення бакалаврської кваліфікаційної роботи	01.06-07.06	виконано
9	Оформлення ілюстративного матеріалу до бакалаврської кваліфікаційної роботи	07.06-12.06	виконано
10	Підготовка доповіді	12.06-16.06	виконано

Студент _____ Нікіта КОЛЕСНИКОВ

Керівник проекту _____ Ірина ПРОКУДІНА.

РЕФЕРАТ

Метою цієї роботи є дослідження та аналіз традиційних сучасних засобів та технологій програмування, які використовуються у сучасному програмуванні. Робота спрямована на вивчення основних характеристик, особливостей та використання цих засобів та технологій, а також їх впливу на процес розробки програмного забезпечення.

Задачі дослідження:

- Вивчити основні традиційні сучасні засоби та технології програмування, такі як Java, C++, Python, JavaScript, Ruby, PHP та інші.
- Дослідити синтаксис, основні принципи та особливості кожної з мов програмування.
- Визначити сфери застосування та основні переваги кожного засобу або технології.
- Проаналізувати розвиток та популярність цих засобів та технологій протягом останніх років.
- Оцінити перспективи та майбутні тренди використання цих засобів та технологій у сфері програмування.

Об'єктом дослідження є традиційні сучасні засоби та технології програмування, які широко застосовуються у сучасному програмуванні.

Предметом дослідження є характеристики, принципи роботи, сфери використання, розвиток та популярність традиційних сучасних засобів та технологій програмування.

Обсяг та структура роботи. Бакалаврська кваліфікаційна робота включає пояснювальну записку, яка виконана на 98 аркушах. Пояснювальна записка складається із 3 основних розділів та містить 6 рисунків, 9 діаграм та 1 таблицю, кількість посилань у списку використаних джерел – 40.

Ключові слова: технології програмування, база даних, розробка, засоби програмування, інформаційна безпека.

ABSTRACT

The method of this work is research and analysis of traditional modern programming tools and technologies used in modern programming. The work is aimed at studying the main characteristics, features and use of these tools and technologies, as well as their impact on the software development process.

Research objectives:

- Learn basic traditional modern programming tools and technologies such as Java, C++, Python, JavaScript, Ruby, PHP and others.
- To study the syntax, basic principles and features of each programming language.
- Identify the applications and main benefits of each or technology.
- Analyze the development and popularity of these tools and technologies in recent years.
- Assess the prospects and future trends of using these tools and technologies in the field of programming.

The object of research is traditional modern programming tools and technologies used in modern broad programming.

The subject of the study is the characteristics, principles of operation, use of the field, development and popularity of traditional modern programming tools and technologies.

Scope and structure of work. Bachelor qualification work includes an explanatory note, which is made on 78 sheets. The explanatory note consists of 3 main sections and contains 6 drawings, 9 diagrams and 1 table, the number of references in the list of used sources is 40.

Keywords: programming technologies, Database, development, programming tools.

ЗМІСТ

ВСТУП

Сучасний світ безперервно розвивається, і програмування вже давно стало невід'ємною складовою цього розвитку. Програмування забезпечує можливість створення інноваційних програмних рішень, що впливають на різні сфери життя, починаючи від побутових пристроїв і закінчуючи складними бізнес-додатками.

Традиційні засоби та технології програмування постійно еволюціонують, адаптуючись до зростаючих потреб сучасного програміста та вимог ринку. Швидкість розвитку програмування змушує фахівців постійно вивчати нові інструменти та підходи, щоб бути конкурентоспроможними і досягти успіху у своїй галузі.

Мета цієї бакалаврської кваліфікаційної роботи полягає в дослідженні та аналізі традиційних сучасних засобів та технологій програмування, що використовуються в сучасному програмуванні. Дослідження цієї теми дозволить розкрити сутність та особливості цих засобів та технологій, а також їх вплив на процес розробки програмного забезпечення.

Актуальність теми зумовлена необхідністю ознайомлення зі сучасними засобами та технологіями програмування, оскільки це допомагає програмістам підвищити ефективність своєї роботи, забезпечує високу якість створюваного програмного продукту та дозволяє швидше впроваджувати нові рішення на ринку.

Обмеження та складність дослідження полягають у швидкому розвитку сфери програмування, що вимагає постійного оновлення знань та вивчення нових засобів і технологій. Проте, аналіз та систематизація інформації про традиційні сучасні засоби та технології програмування дозволять нам отримати глибше розуміння процесу розробки програмного забезпечення і визначити найбільш оптимальні підходи та інструменти для роботи.

РОЗДІЛ 1: РОЗВИТОК ТЕХНОЛОГІЙ ТА ЗАСОБІВ ПРОГРАМУВАННЯ: ВІД МИНУЛОГО ДО СЬОГОДЕННЯ

ОГЛЯД СУЧАСНИХ ЗАСОБІВ ТА ТЕХНОЛОГІЙ ПРОГРАМУВАННЯ

1.1 Розвиток технологій програмування

Технології програмування – це сукупність методів та інструментів, що використовуються під час розроблення програмного забезпечення. Вони містять у собі як технічні аспекти (мови програмування, фреймворки та бібліотеки), так і методологічні (моделі життєвого циклу розробки, архітектурні шаблони тощо). Основна мета технологій програмування – забезпечення швидкого та якісного створення програмного забезпечення.

Технології та засоби програмування почали активно розвиватися в середині XX століття. Ключовим моментом у цьому процесі був постійний розвиток і впровадження нових технологій, які покращували процес програмування і давали змогу створювати складніші та масштабніші проєкти.

Спочатку програми писалися мовами низького рівня, такими як асемблер. У той час основним завданням було написання програм для наукових і математичних розрахунків. Надалі з’явилися мови високого рівня, як-от Fortran і COBOL, які давали змогу програмістам писати програми зрозумілішою людині мовою. Пізніше з’явилися більш високорівневі мови програмування, такі як C і Pascal, які дали змогу спростити процес написання коду.

А вже сьогоднішній етап розвитку технології програмування пов’язаний з появою нових технологій, таких як штучний інтелект, машинне навчання, блокчейн і багато інших. Також стали активно використовуватися відкриті вихідні коди і технології, такі як Linux, Git, Docker тощо. Ці технології дають змогу розробникам швидко й ефективно створювати програмне забезпечення, підвищуючи тим самим продуктивність і якість проєктів.

Основні етапи:

1. Перше покоління програмування, яке також відоме як машинний код, охоплює період з початку 1940-х до середини 1950-х років. В цей період були вперше створені електронні цифрові комп'ютери, і розробники стикалися з необхідністю створення програмного забезпечення для керування цими пристроями.

Ось деякі ключові аспекти першого покоління програмування:

- Машинний код: В першому поколінні програмування програми писалися безпосередньо у вигляді двійкових кодів, які розумів комп'ютер. Це були набори інструкцій, які могли керувати операціями комп'ютера, такими як арифметичні операції та збереження даних.
- Використання машини: Програми були написані для конкретних моделей комп'ютерів, і вони працювали безпосередньо на цих машинах. Кожна машина мала свій власний набір інструкцій та формати даних, що ускладнювало перенесення програм між різними системами.
- Планування асемблерів: Для полегшення написання програм у машинних кодах були створені асемблери. Асемблери дозволяли програмістам використовувати символьні мнемоніки для представлення машинних інструкцій та даних, що полегшувало розробку програм та зрозуміння коду.
- Обмежена функціональність: У порівнянні з сучасними мовами програмування перше покоління мало обмежений функціонал. Програми могли виконувати основні арифметичні операції, зберігати та завантажувати дані, а також здійснювати прості операції з введенням-виведенням.
- Ручна оптимізація: Програмісти першого покоління часто проводили ручну оптимізацію програм, оскільки комп'ютери того часу мали обмежені ресурси. Це включало вибір оптимальних алгоритмів та структур даних, розміщення даних у пам'яті та інші прийоми для поліпшення продуктивності програм.

Перше покоління програмування було важливим етапом у розвитку комп'ютерних технологій, оскільки воно створило основу для подальшого розвитку високорівневих мов програмування та інших засобів, що полегшують розробку програмного забезпечення.

2. Друге покоління програмування включає в себе період з середини 1950-х по початок 1960-х років і характеризується значними змінами в області програмування і технологій комп'ютерів. У цей період відбулися важливі події і досягнення, що вплинули на розвиток програмування. Ось деякі ключові аспекти другого покоління програмування:

- Високорівневі мови програмування: З'явилися нові високорівневі мови програмування, які були спрямовані на більш зрозуміле та зручне програмування. Серед них були Fortran II, Algol та COBOL. Ці мови використовували абстракції, такі як підпрограми, умовні оператори та цикли, що полегшувало розробку складних програм.
- Магнітні стрічки: В цей період стали широко використовуватися магнітні стрічки як засіб зберігання і передачі даних. Це відкрило нові можливості для збереження програм та даних, а також для обміну інформацією між комп'ютерами.
- Багатозадачність: З'явилися перші комп'ютерні системи з підтримкою багатозадачності, що дозволяли виконувати кілька програм одночасно. Це відкрило шлях до розробки складних систем та додатків, що працюють паралельно.
- Збільшення продуктивності: Комп'ютери другого покоління стали швидшими, потужнішими і більш надійними. Це дало змогу розробникам створювати більш складні програми та обробляти великі обсяги даних.
- Розвиток операційних систем: Були створені операційні системи, які спрощували управління ресурсами комп'ютера та дозволяли розробникам працювати в зручному середовищі. Прикладами таких операційних систем були IBM OS/360 та Unix.

Друге покоління програмування принесло значні поліпшення у виразності мов програмування, ефективності комп'ютерів та розширенні можливостей розробників. Цей період відіграв важливу роль у подальшому розвитку програмування та комп'ютерних технологій.

3. Третє покоління програмування охоплює період з 1960-х років до середини 1970-х років і є продовженням розвитку програмування та комп'ютерних технологій. У цей період були введені нові концепції та інструменти, які значно поліпшили ефективність та продуктивність розробки програмного забезпечення. Ось деякі ключові аспекти третього покоління програмування:

- Високорівневі мови програмування: Третє покоління характеризується подальшим розвитком високорівневих мов програмування. Мови, такі як Pascal, C та ALGOL 68, були створені з метою полегшення процесу програмування та зростання продуктивності розробників. Вони мали більш розширену функціональність, включали конструкції управління потоком, підтримували складні структури даних та надавали засоби для створення більш ефективних програм.
- Компілятори та інтерпретатори: З'явилися потужні компілятори та інтерпретатори, які дозволяли перетворювати вихідний код програми на машинний код або виконувати його безпосередньо. Це покращило продуктивність програмістів та ефективність виконання програм.
- Структурне програмування: В цей період виникло поняття структурного програмування, яке передбачало використання блоків коду, умовних операторів та циклів для полегшення розуміння та обслуговування програм. Це допомогло уникнути проблеми спагеті-коду (програмного коду, в якому важко зрозуміти логіку через зламані та складні перехресні залежності) та зробило програми більш читабельними та обслуговуваними.
- ООП та модульне програмування: У третьому поколінні з'явилися концепції об'єктно-орієнтованого програмування (ООП) та модульного програмування. ООП дозволяло розробляти програми з використанням

об'єктів, які містять дані та методи для їх обробки. Це забезпечувало більшу структурованість та повторне використання коду. Модульне програмування дозволяло розбити програму на окремі модулі, що сприяло полегшенню розробки, тестування та підтримки програмного забезпечення.

- Віртуальна машина: Третє покоління також відзначається впровадженням віртуальних машин, які дозволяли виконувати програми в середовищі, що імітує реальну апаратну платформу. Це забезпечувало переносимість програм між різними комп'ютерами та оптимізацію виконання програм.

4. Четверте покоління програмування відноситься до періоду з 1970-х років до нашого часу і характеризується використанням високорівневих мов програмування, побудовою комп'ютерних систем високої рівня, заснованих на архітектурі мікропроцесорів, та розвитком нових технологій програмування. Ось деякі ключові аспекти четвертого покоління програмування:

- Високорівневі мови програмування: В цьому поколінні були створені та розроблені високорівневі мови програмування, такі як C, C++, Java, Python, Ruby і багато інших. Ці мови мають розширений функціонал, легкість використання та більш високий рівень абстракції, що дозволяє розробникам писати програми більш ефективно та швидко.
- Об'єктно-орієнтоване програмування (ООП): ООП стало однією з найважливіших концепцій у четвертому поколінні програмування. Цей підхід дає змогу створювати програми, які складаються з об'єктів, що містять дані та функції для їх обробки. ООП сприяє повторному використанню коду, поліпшенню модульності та підтримці складних програмних систем.
- Розподілене програмування: Четверте покоління також відзначається розробкою та використанням розподілених систем, які дозволяють програмам працювати на різних комп'ютерах або в мережах. Це включає розробку веб-додатків, мобільних додатків та хмарних рішень, які розширюють можливості програмного забезпечення.

- Інтернет та веб-технології: Четверте покоління відкрило шлях до розвитку Інтернету та веб-технологій. Розробники отримали можливість створювати веб-додатки, які працюють в браузерах, спілкуються з серверами та надають різноманітні послуги, такі як соціальні мережі, електронна комерція, онлайн-банкінг та багато іншого.
- Автоматизація та інструменти розробки: В четвертому поколінні з'явилися потужні інструменти розробки програмного забезпечення, такі як інтегровані середовища розробки (IDE), системи керування версіями, тестування та налагодження програм. Ці інструменти сприяють підвищенню продуктивності розробників та якості програмного забезпечення.

Четверте покоління програмування відкрило безліч нових можливостей для розробників програмного забезпечення і поклали основу для подальшого розвитку інформаційних технологій. З його допомогою було створено багато іноваційних продуктів та рішень, які суттєво змінили наш спосіб життя та роботи.

5. На даний момент, п'яте покоління програмування ще не визначене чітко і широко застосовуване, оскільки це поняття використовується для опису майбутніх технологій та напрямків розвитку програмування. Однак, існують деякі концепції та напрями, які асоціюються з п'ятим поколінням програмування:

- Штучний інтелект (AI): П'яте покоління може бути пов'язане з розробкою програмного забезпечення, яке здатне до самостійного навчання, аналізу даних та прийняття розумних рішень. AI включає в себе такі підходи, як машинне навчання, нейронні мережі, генетичні алгоритми та інші технології, що дозволяють програмам розуміти, аналізувати та вирішувати складні завдання.
- Квантове програмування: Ще одним напрямком, пов'язаним з п'ятим поколінням, є розвиток квантового програмування. Квантові комп'ютери пропонують значно більшу обчислювальну потужність порівняно з класичними комп'ютерами, що відкриває нові можливості для обробки

даних та вирішення складних проблем, зокрема в областях криптографії, медицини, матеріалознавства та інших галузях.

- Розширена реальність (AR) та віртуальна реальність (VR): З розвитком AR та VR технологій з'являються нові можливості для створення програмного забезпечення, що взаємодіє з реальним світом або створює іммерсивне віртуальне середовище. Це охоплює галузі, такі як ігри, симуляції, тренування, візуалізація даних та інші сфери.
- Інтернет речей (IoT): П'яте покоління також може охоплювати розробку програмного забезпечення для підключених пристроїв та систем, які спілкуються між собою через Інтернет. IoT включає в себе розумних пристроїв, датчики, мережі зв'язку та аналітику даних, що відкриває нові можливості для автоматизації та взаємодії з фізичним світом.

Ці напрямки вказують на можливі шляхи розвитку програмування в майбутньому, але важливо зауважити, що п'яте покоління програмування все ще є предметом активних досліджень та розвитку, і точне визначення та обсяг цього покоління можуть змінюватися з часом.

Кожен етап розвитку технології програмування не означав повного заміщення попередніх технологій новими. Більш того, багато з них й досі використовуються в різних проєктах. Кожен етап являється наступним кроком у розвитку та вдосконаленні засобів та технологій програмування.

1.2 Засоби програмування: загальний огляд та роль у розробці

Сучасні засоби програмування є невід'ємною частиною розробки програмного забезпечення. Їх розвиток та постійне вдосконалення відіграють важливу роль у підтримці та полегшенні процесу програмування.

Засоби програмування включають широкий спектр інструментів та технологій, які допомагають розробникам створювати, тестувати, відлагоджувати та підтримувати програмний код. Вони забезпечують різні функціональні можливості, спрощуючи процес розробки та підвищуючи продуктивність розробників.

1. Одним з найважливіших засобів програмування є інтегровані середовища розробки (IDE). Integrated Development Environment (IDE) або інтегроване середовище розробки - це програмний інструмент, який надає розробникам зручне та комплексне середовище для написання, редагування, компіляції та налагодження програмного коду. IDE об'єднує різні інструменти, які допомагають полегшити роботу розробника та забезпечують продуктивність та ефективність процесу розробки програмного забезпечення.

Основні складові інтегрованого середовища розробки включають:

- Редактор коду: IDE надає розробнику зручне середовище для написання та редагування коду. Редактор коду зазвичай має функції автодоповнення, підсвічування синтаксису, автоматичне вирівнювання та інші корисні функції, що допомагають розробнику писати код швидше та з меншими помилками.
- Компілятор або інтерпретатор: IDE часто включає компілятор або інтерпретатор, який дозволяє перетворити вихідний код на виконуваний формат. Це дозволяє розробнику перевірити правильність синтаксису та логіки програми та виконати її для перевірки працездатності.
- Відладчик: IDE надає вбудований відладчик, який допомагає розробнику відстежувати та вирішувати помилки в програмному коді. Відладчик дозволяє ставити точки зупинки, виконувати програму по крокам, спостерігати значення змінних та аналізувати стек викликів функцій.
- Управління проектом: IDE надає інструменти для управління проектом, такі як створення, видалення та перейменування файлів, організація файлів у структурах папок, інтеграція з версійними системами та інші функції, які допомагають організувати та керувати проектом.
- Підтримка мов програмування: IDE підтримує різні мови програмування та надає спеціалізовані функції для кожної мови. Це включає синтаксичне підсвічування, автодоповнення, довідку по функціях, перевірку синтаксису та багато інших інструментів, які полегшують розробку в конкретній мові програмування.

Інтегроване середовище розробки дозволяє розробникам працювати в єдиному, зручному та продуктивному середовищі, що сприяє поліпшенню якості та ефективності розробки програмного забезпечення. Крім того, IDE часто мають розширення та плагіни, які розширюють їх функціональність та дозволяють налаштувати робоче середовище під потреби кожного розробника.

2. Текстовий редактор - це програмний інструмент, який дозволяє розробникам створювати, редагувати та форматувати текстові файли, включаючи програмний код. Текстові редактори широко використовуються розробниками програмного забезпечення для написання та редагування коду, а також для роботи з текстовими документами.

Основні особливості текстових редакторів включають:

- Редагування коду: Текстові редактори надають зручне середовище для введення та редагування коду. Вони підтримують основні функції, такі як вставка, видалення, копіювання, переміщення тексту, а також можуть мати функцію автодоповнення, яка допомагає розробникам швидко вписувати фрагменти коду.
- Синтаксичне підсвічування: Текстові редактори часто надають можливість синтаксичного підсвічування, що дозволяє розрізняти ключові слова, коментарі, рядки коду та інші елементи мови програмування. Це полегшує читання та редагування коду, а також допомагає виявляти синтаксичні помилки.
- Функції форматування: Деякі текстові редактори мають функції форматування, які допомагають автоматично вирівнювати код, додавати відступи, виправляти форматування для поліпшення зрозумілості коду.
- Підтримка багатьох мов програмування: Текстові редактори зазвичай підтримують багато різних мов програмування, що дозволяє розробникам працювати з кодом у своїй улюбленій мові.
- Розширення та налаштування: Багато текстових редакторів підтримують розширення та плагіни, які дозволяють розширити їх

функціональність та налаштувати робоче середовище під потреби розробника.

Текстові редактори можуть бути простими та легкими, або більш потужними з розширеними функціями. Деякі популярні текстові редактори включають Sublime Text, Visual Studio Code, Atom, Notepad++, Vim, Emacs та багато інших. Кожен розробник може обрати той, який найкраще відповідає його потребам та персональним вподобанням.

3. Компілятори та інтерпретатори є ключовими інструментами в процесі виконання програмного коду. Вони використовуються для перетворення вихідного коду, написаного розробником, на машинний код, який комп'ютер може розуміти та виконувати. Хоча обидва інструменти служать для виконання коду, вони мають деякі основні відмінності у своєму підході.

Компілятор - це програмний інструмент, який перетворює вихідний код цілком у машинний код перед його виконанням. Процес компіляції включає кілька етапів: лексичний аналіз, синтаксичний аналіз, семантичний аналіз, генерацію проміжного коду та оптимізацію. Компілятор перетворює вихідний код в проміжний код або машинний код, який може бути виконаний безпосередньо комп'ютером. Приклади відомих компіляторів включають GCC (GNU Compiler Collection), Clang, Microsoft Visual C++ Compiler та інші.

Інтерпретатор - це програмний інструмент, який зчитує вихідний код рядок за рядком і безпосередньо виконує його. Він не перетворює код на машинний код перед виконанням. Інтерпретатор аналізує, розуміє та виконує інструкції програми покроково. Це означає, що код виконується по мірі його інтерпретації. Деякі популярні інтерпретовані мови програмування включають Python, Ruby, JavaScript та PHP. Крім того, існують також інтерпретатори, які можуть виконувати байт-код, такі як Java Virtual Machine (JVM) для мови Java або Common Language Runtime (CLR) для мови C#.

Основна відмінність між компіляторами та інтерпретаторами полягає у їхньому підході до виконання коду. Компілятор перетворює вихідний код у

машинний код, що забезпечує швидке виконання програми, але вимагає попереднього компілювання перед запуском. З іншого боку, інтерпретатор виконує код безпосередньо, що забезпечує більшу гнучкість та спрощує розробку, але може бути менш ефективним з точки зору швидкодії.

Обидва підходи мають свої переваги та недоліки, і вибір між компілятором та інтерпретатором залежить від мови програмування, проекту та специфічних вимог розробки.

4. Версійна система (Version Control System або VCS) є інструментом, що дозволяє відстежувати та керувати змінами в файловій системі проекту. Вона зберігає всі версії файлів, що дозволяє розробникам простіше співпрацювати, відновлювати попередні версії та контролювати розвиток проекту. Основні типи версійних систем включають локальні, централізовані та розподілені системи керування версіями.

- Локальна версійна система: Локальна VCS зберігає всі версії файлів безпосередньо на локальному комп'ютері розробника. Коли розробник створює нову версію файлу, система зберігає його копію з повною історією змін. Це простий підхід, але не надає можливості спільної роботи між розробниками.
- Централізована версійна система: У централізованій VCS всі версії файлів зберігаються на центральному сервері. Розробники спільно працюють з центральним репозиторієм, зберігаючи та отримуючи оновлення змін. Вони можуть здійснювати зміни та зберігати їх на сервері, а також отримувати зміни інших розробників. Приклади централізованих VCS включають Subversion (SVN) та Perforce.
- Розподілена версійна система: Розподілена VCS зберігає всі версії файлів не тільки на центральному сервері, але й на комп'ютерах кожного розробника. Кожен розробник має повну копію репозиторію, що дозволяє їм працювати незалежно від мережевого з'єднання з сервером. Розподілені VCS, такі як Git і Mercurial, надають більшу гнучкість та забезпечують ефективну спільну роботу навіть при розробці великих проектів.

Версійні системи дозволяють розробникам керувати змінами, відстежувати прогрес розробки, співпрацювати з іншими розробниками, відновлювати попередні версії та вирішувати конфлікти. Вони є важливим інструментом у сучасній розробці програмного забезпечення, що сприяє покращенню ефективності та якості роботи розробників.

5. Спеціалізовані бібліотеки, фреймворки та інструменти є невід'ємною частиною сучасного програмування. Вони надають розробникам готові компоненти, модулі та інструменти, що спрощують розробку програмного забезпечення і забезпечують високу продуктивність. Ось деякі загальні відомості про ці поняття:

- Бібліотеки: Бібліотека - це набір готових функцій, класів, модулів або компонентів, які можна використовувати для розв'язання певних завдань або функціональностей. Вони надають розробникам певний набір інструментів для вирішення конкретних завдань. Приклади бібліотек включають NumPy для наукових обчислень, Pandas для обробки та аналізу даних, Requests для взаємодії з веб-серверами тощо. Розробник може імпортувати бібліотеку в свій проект та використовувати її функції для полегшення роботи з конкретними завданнями.
- Фреймворки: Фреймворк - це комплексна структура або платформа, що надає шаблони, бібліотеки та інструменти для розробки програмного забезпечення. Фреймворки допомагають розробникам створювати програми швидше і з меншими зусиллями, надаючи готові компоненти та структуру для роботи. Вони встановлюють правила та стандарти для організації проекту та взаємодії з бібліотеками. Приклади фреймворків включають Django для веб-розробки, Ruby on Rails для швидкої розробки програм, TensorFlow для машинного навчання тощо. Розробник використовує фреймворк, щоб створити програму за допомогою готових модулів, забезпечуючи швидкість розробки та використання найкращих практик.

- Інструменти: Інструменти програмування - це програми або сервіси, що допомагають розробникам у виконанні різних задач, таких як управління версіями, налагодження коду, автоматичне тестування та багато іншого. Інструменти можуть бути спеціалізованими для певної мови програмування або платформи, або ж загального призначення. Приклади включають Git для керування версіями, IntelliJ IDEA для розробки Java, Visual Studio для розробки .NET, JUnit для тестування Java-коду тощо. Інструменти допомагають розробникам підтримувати якість коду, ефективно працювати та покращувати процес розробки.

Ці спеціалізовані бібліотеки, фреймворки та інструменти значно полегшують розробку програмного забезпечення, дозволяють розробникам використовувати готові компоненти та швидше досягати результату. Вибір конкретного інструментарію залежить від типу проекту, мови програмування та інших вимог розробки.

Сучасні засоби програмування відіграють важливу роль у підвищенні продуктивності розробників та якості програмного забезпечення. Вони спрощують процес програмування, допомагають зберігати та керувати кодом, надають засоби для швидкої розробки та тестування програм, а також дозволяють використовувати сучасні технології та практики, такі як DevOps, Continuous Integration та інші.

Усі ці засоби програмування постійно розвиваються і оновлюються, щоб задовольняти зростаючі потреби розробників та відповідати вимогам швидко змінюючого IT-середовища. Процес програмування стає все більш швидким та ефективним завдяки сучасним засобам, що веде до поліпшення якості програмного забезпечення та забезпечує успішну реалізацію проектів у різних галузях.

1.3 Висновок до 1 розділу

У сучасному світі, де технології швидко розвиваються, засоби та технології програмування відіграють ключову роль у створенні програмного

забезпечення. Розробники постійно шукають нові інструменти та методики, щоб полегшити процес програмування, підвищити продуктивність та забезпечити високу якість розробленого програмного забезпечення.

У цьому розділі було розглянуто різні аспекти сучасних засобів та технологій програмування. Було розкрито роль засобів програмування в процесі розробки програмного забезпечення, їхній вплив на продуктивність та якість коду, а також описано різні типи засобів, такі як інтегровані середовища розробки (IDE), текстові редактори, компілятори та інтерпретатори, системи контролю версій, фреймворки та бібліотеки.

Також було розглянуто розвиток програмування з ранніх періодів до сучасності, включаючи покоління програмування, такі як перше, друге, третє та четверте покоління. Кожне покоління мало свої особливості та внесло вагомий внесок у розвиток програмування.

Оглянувши основні засоби та технології програмування, можна зробити висновок, що сучасне програмування надає безліч інструментів та можливостей для розробників. Ці засоби допомагають підвищити продуктивність, створити якісне програмне забезпечення та забезпечити ефективну роботу розробників у команді.

Однак, необхідно зазначити, що технології та засоби програмування продовжують еволюцію і змінюються з часом. Розробники повинні бути в курсі останніх трендів та навичок, адаптуватися до нових вимог та викликів і навчатися протягом усього свого професійного шляху.

РОЗДІЛ 2. СУЧАСНІ ЗАСОБИ ТА ТЕХНОЛОГІЇ ПРОГРАМУВАННЯ

2.1 Веб-розробка та фреймворки

2.1.1 Веб-розробка

Веб-розробка є одним із найбільш швидкозростаючих сегментів сфери програмування. Завдяки інтернету та розширенню онлайн-сервісів, веб-додатки та сайти стали необхідними для бізнесу, комунікації та розваг. Веб-розробка охоплює різні аспекти, включаючи клієнтську та серверну сторони, бази даних, дизайн та безпеку. Щоб полегшити процес розробки та підвищити продуктивність, розробники використовують фреймворки - набори інструментів та бібліотек, які надають готові рішення та структуру для веб-додатків.

Веб-розробка має велике значення як для бізнесу, так і для користувачів. За допомогою веб-сайтів та веб-додатків компанії можуть презентувати свої послуги та товари, взаємодіяти з клієнтами та забезпечувати їм зручний доступ до інформації. Користувачі, у свою чергу, отримують можливість отримувати потрібну інформацію, здійснювати покупки, спілкуватися та виконувати різноманітні завдання через веб-інтерфейс.

Одним з основних аспектів веб-розробки є мови програмування та технології, що використовуються для розробки веб-додатків. HTML (HyperText Markup Language) використовується для створення структури веб-сторінок, CSS (Cascading Style Sheets) відповідає за оформлення та вигляд веб-елементів, а JavaScript надає можливості для розробки інтерактивності та динамічного контенту. Додатково, для розробки більш складних веб-додатків використовуються різні серверні мови програмування, такі як PHP, Python, Ruby, Java та інші.

Фреймворки є важливими інструментами веб-розробки, оскільки вони надають структуру та ряд готових компонентів для розробки веб-додатків. Фреймворки спрощують роботу розробників, дозволяючи їм швидше

створювати функціональні веб-додатки з меншими зусиллями. Деякі популярні фреймворки включають Ruby on Rails, Django, Laravel, Angular, React та інші. Кожен фреймворк має свої особливості та набір інструментів, які допомагають розробникам ефективно створювати веб-додатки.

Окрім мов програмування та фреймворків, веб-розробка також включає в себе роботу з базами даних, веб-серверами, протоколами передачі даних та іншими технологіями. Для зберігання та управління даними використовуються реляційні бази даних, такі як MySQL та PostgreSQL, а також нереляційні бази даних, наприклад MongoDB та Cassandra. Веб-сервери, такі як Apache та Nginx, забезпечують обробку запитів та доставку веб-сторінок користувачам.

Найпопулярніші мови для веб-розробки:

- **JavaScript.** Одна з найпоширеніших мов, її сфера використання практично безмежна. JS - не одне й те саме, що Java. JavaScript використовується для написання комп'ютерних, серверних та мобільних додатків. Усі сценарії, написані на JS, виконуються у браузері.
- **PHP.** Серверна мова, що використовується для програмування програм. З її допомогою можна вирішувати різні обчислювальні операції, але створити сайт з нуля не вдасться. Популярність PHP обумовлена гнучкістю, продуктивністю та простотою.
- **ASP.** Є скриптовою мовою VBScript. ASP – середовище розробки, запропоноване корпорацією Microsoft. Її суть полягає у можливості впровадження елементів коду на звичайну сторінку HTML.
- **PERL.** Являє собою універсальну серверну мову загального призначення. З її допомогою можна розробляти CGI-програми, що використовуються для роботи з веб-серверами. PERL дозволяє обробляти великий обсяг даних, характеризується кросплатформністю та вільним синтаксисом.

- **Python.** Основною перевагою цієї мови розробки є простота. Особливість Python – економія часу розробника. Ця мова пропонує безліч бібліотек з готовими моделями конструкцій. Одним із недоліків Python є динамічна типізація. Код можна писати швидко, не оголошуючи типи змінних, але це може призвести до помилок.
- **Ruby.** Універсальна мова програмування для роботи з веб-проєктами та програмами. Спочатку Ruby не отримав широкої поширеності через повільність і неможливість масштабування великих сайтів. Але оновлення допомогли вирішити проблему. Час відгуку програми, написаної на Ruby, залежить від правильності побудови архітектури.
- **CSS.** Таблиця стилів, що відповідає за зовнішній вигляд веб-сторінок. CSS можна лише умовно назвати мовою програмування. За допомогою CSS3 ви зможете створювати анімації та втілювати у життя будь-які дизайнерські ідеї.
- **C#**. Має багато схожого з C++ та Java. C# спочатку була задумана як інструмент розробки сайтів. Сьогодні ця мова розвивається швидкими темпами, регулярно виходять доповнення та оновлення. Розробники можуть використовувати різні асинхронні методи та динамічні зв'язування.
- **Java**. Використовується переважно для розробки мережевого софту та мобільних додатків. Java - основна мова програмування для Андроїд. Для неї характерна надійність, безпека та простота синтаксису.
- **SQL**. Оптимальний варіант розробки систем управління базами даних. SQL може використовуватися як розробниками, так і адміністраторами БД. Ця мова є основною для роботи з базами даних, але вона досить складна.

2.1.2 Фреймворки

Фреймворки є невід'ємною частиною сучасного програмування. Вони забезпечують розробникам набір інструментів, бібліотек та шаблонів, які допомагають створювати програмне забезпечення ефективніше та швидше. Фреймворки відповідають за структуру та архітектуру проекту, дозволяючи розробникам сконцентруватися на реалізації функціональності, замість написання великої кількості загальних кодових частин.

Одним з основних переваг використання фреймворків є полегшення розробки програмного забезпечення. Замість того, щоб писати всю необхідну функціональність з нуля, розробники можуть скористатися готовими компонентами, модулями та рішеннями, що прискорює розробку. Фреймворки забезпечують структурований підхід до розробки, де заздалегідь визначені правила та стандарти, що допомагають підтримувати чистоту та організацію коду.

На Рис.2.1.2.1 зображено ряд найпопулярніших фреймворків серед розробників.

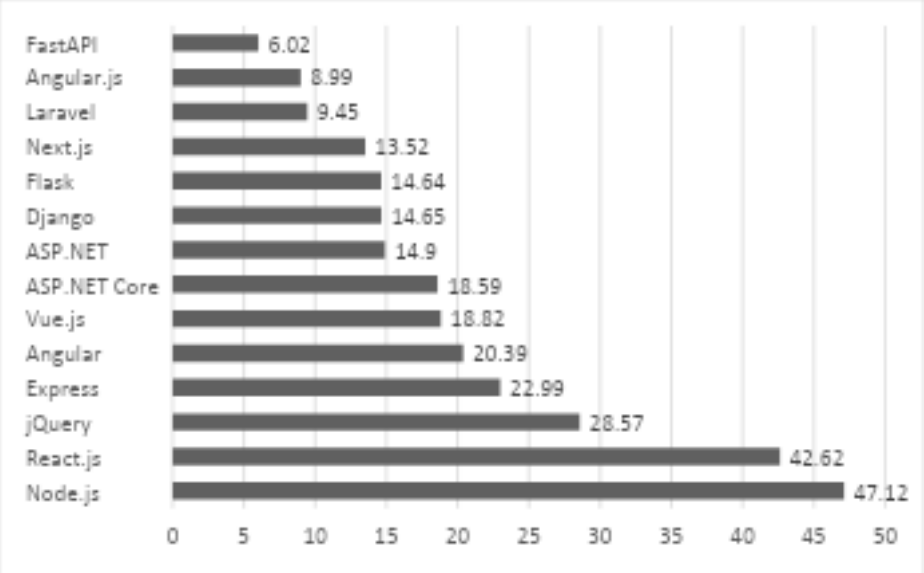

Рис.2.1.2.1 – Статистика найпопулярніших фреймворків

Іншою вагомою перевагою фреймворків є підтримка та розширюваність. Вони зазвичай мають активну спільноту розробників, яка стежить за оновленнями та вносить внески у розвиток фреймворку. Це означає, що розробники можуть легко отримувати підтримку, документацію та оновлення фреймворку. Крім того, багато фреймворків мають розширення та плагіни, які дозволяють додавати нову функціональність та розширювати можливості фреймворку.

Фреймворки також сприяють підвищенню якості програмного забезпечення. Завдяки використанню стандартизованої архітектури та компонентів, фреймворки допомагають запобігати появі помилок, забезпечують більшу безпеку та надійність програмного забезпечення. Крім того, вони надають інструменти для автоматизованого тестування, що сприяє виявленню помилок та поліпшенню якості коду.

На сьогоднішній день існує велика кількість фреймворків для різних мов програмування та галузей розробки. Наприклад, в сфері веб-розробки популярними фреймворками є Ruby on Rails для Ruby, Django для Python, Laravel для PHP та інші. Кожен з цих фреймворків має свої особливості та переваги, але загальна мета є спрощення та прискорення процесу розробки програмного забезпечення.

2.2 Мобільна розробка

Мобільна розробка стала надзвичайно актуальною в сучасному світі, оскільки мобільні пристрої стали неодмінною частиною нашого повсякденного життя. За допомогою мобільних додатків ми можемо отримувати інформацію, спілкуватися з іншими людьми, проводити операції з банківськими рахунками, покупати товари та послуги та багато іншого. Мобільна розробка включає створення програмного забезпечення для мобільних платформ, таких як iOS і Android, з метою надання користувачам зручного та ефективного досвіду.

Одним з ключових аспектів мобільної розробки є розробка мобільних додатків. Мобільні додатки можуть бути класифіковані як нативні, гібридні або веб-перехідні. Нативні додатки розробляються спеціально для конкретної платформи, використовуючи мови програмування та інструменти, специфічні для цієї платформи. Нативні додатки мають високий рівень продуктивності та можуть максимально використовувати функціональність пристрою. Гібридні додатки комбінують код з веб-технологіями, такими як HTML, CSS та JavaScript, щоб розробляти додатки, які працюють на кількох платформах. Веб-перехідні додатки розробляються з використанням веб-технологій та можуть бути запаковані в контейнер для доступу до нативних функцій пристрою.

Одним з основних інструментів у мобільній розробці є розробка інтерфейсу користувача. Зручний та естетичний дизайн інтерфейсу допомагає користувачам ефективно взаємодіяти з мобільним додатком. Для розробки інтерфейсу користуються спеціальними засобами, такими як Xcode для iOS та Android Studio для Android, які надають набір елементів і шаблонів для створення візуально привабливих та функціональних інтерфейсів.

Мобільна розробка також включає тестування, оптимізацію та розгортання мобільних додатків. Тестування допомагає виявляти помилки та проблеми в додатку, забезпечуючи якість та стабільність роботи. Оптимізація додатків спрямована на поліпшення продуктивності, швидкості роботи та ефективного використання ресурсів пристрою.

Основні завдання розробника мобільних додатків:

- створення ТЗ (технічного завдання) на розробку мобільного додатку;
- обговорення з замовником етапів і процесу роботи проекту;
- побудова архітектури додатку;
- безпосередньо програмування;
- робота з дизайнерами;
- підтримка мобільних додатків;
- робота з тестувальниками над налагодженням і тестуванням додатків;

- допомога в створенні інструкцій по роботі з готовим додатком;
- оформлення документації;
- розміщення додатків в AppStore і Google Play Market, Amazon Appstore, Opera Mobile Store і інших магазинах мобільних додатків.

Мобільна розробка постійно розвивається, і нові технології та інструменти постійно з'являються, щоб полегшити роботу розробників та покращити досвід користувачів. За останні роки з'явилися такі тренди, як розробка з використанням фреймворків, хмарні технології, штучний інтелект та машинне навчання, які активно використовуються в мобільній розробці.

За останні десятиліття ми спостерігаємо значне зростання популярності мобільних пристроїв, таких як смартфони та планшети, що впливає на попит на мобільні додатки. Розробники мобільних додатків зробили великий крок уперед, пропонуючи нові технології та інноваційні рішення, з якими швидко зростає кількість додатків для потенційних користувачів

Згідно з Рис 2.2.1 ми можемо спостерігати постійно зростаючу динаміку на кількість завантажень мобільних додатків. Це вказує на те , що кількість попиту на мобільні додатки зростає з кожним роком незалежно від типу та сфери використання цих додатків. За загальною статистикою на кожного користувача припадає 42 завантажені ним додатки.

Рис.2.2.1 – Глобальна статистика скачуваних додатків

Статистика витраченого часу користувачами у додатках може значно варіюватися в залежності від типу додатків, їх функціональності та специфіки використання. Однак, якщо говорити в загальному, то на Рис.2.2.2 ми знову можемо спостерігати зростаючий темп на витрачний у додатках час.

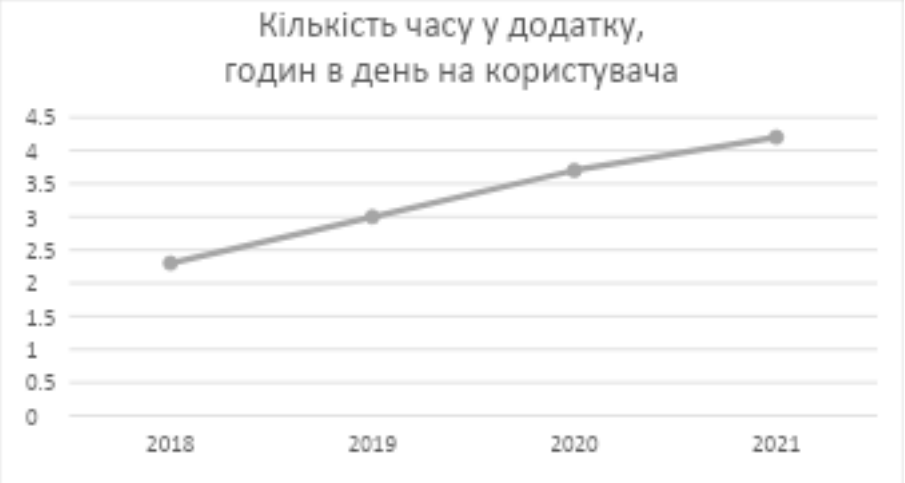

Рис.2.2.2 – Глобальна статистика витраченого часу у дадатку

Розвиток мобільної розробки продовжує набирати темп, як для користувачів, так і для розробників, що призводить до зростання витрат на це напрямок. На Рис. 2.2.3 можна помітити швидке збільшення витрат на мобільну розробку, а також на її подальше оновлення та супровід. Це свідчить про зростаючу важливість мобільних додатків у сучасному світі та залучення додаткових ресурсів для їх розробки та підтримки.

Оскільки мобільна розробка стає все більш конкурентним і швидкозростаючим сектором, важливо мати достатні фінансові засоби і ресурси, щоб забезпечити якість та успіх мобільних додатків.

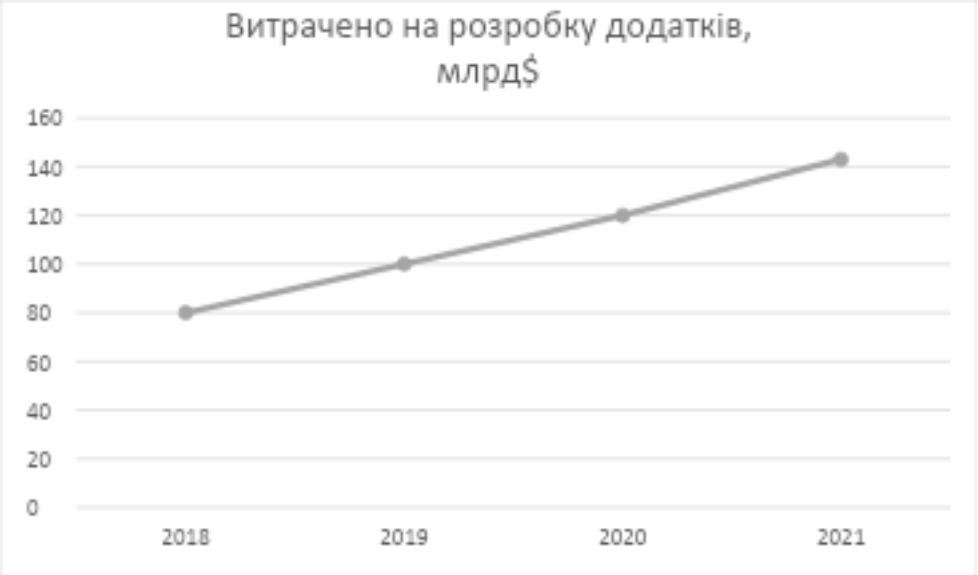

Рис.2.2.2 – Статистика витрачених коштів на розробку додатків

2.3 Хмарні технології

Хмарні технології є однією з найважливіших тенденцій в сучасному інформаційному та технологічному світі. Вони змінюють спосіб, яким люди, компанії та організації працюють з даними, зберігають їх і отримують доступ до них. Хмарні технології надають гнучкість, ефективність і доступність, що робить їх популярними серед користувачів та бізнес-середовищ.

Основною ідеєю хмарних технологій є збереження та обробка даних на віддалених серверах, які називаються хмарою. Користувачі можуть звертатися до своїх даних через Інтернет, використовуючи різноманітні пристрої, такі як комп'ютери, смартфони чи планшети. Це дозволяє отримувати доступ до своїх даних з будь-якого місця та в будь-який час, що робить роботу більш гнучкою та мобільною.

Одним з основних переваг хмарних технологій є масштабованість. Великі хмарні платформи мають значні ресурси, які можуть бути розширені або зменшені залежно від потреб користувачів. Це дозволяє ефективно використовувати ресурси, зменшуючи витрати на обладнання та інфраструктуру.

Хмарні технології також забезпечують високий рівень безпеки. Провайдери хмарних послуг забезпечують захист даних, використовуючи

шифрування, аутентифікацію та інші методи. Вони також регулярно оновлюють свої системи, щоб захистити користувачів від потенційних загроз.

Окрім збереження та обробки даних, хмарні технології також надають доступ до різноманітних сервісів. Наприклад, користувачі можуть використовувати хмарні сервіси для зберігання, синхронізації та обміну даними, створення та спільної роботи над документами, виконання обчислювальних задач і багато іншого.

Хмарні технології знаходять застосування в різних галузях, включаючи бізнес, освіту, медицину, розваги та багато інших. Вони сприяють покращенню ефективності, зниженню витрат та забезпеченню доступності до інформації та сервісів.

Нижче наведений список найпопулярніших хмарних технологій, що мають значний вплив на сучасне інформаційне середовище:

- Amazon Web Services (AWS): AWS є найбільш популярною платформою хмарних послуг, яка надає широкий спектр сервісів, включаючи зберігання даних, обчислювальні ресурси, машинне навчання та інше. AWS є надійним та масштабованим рішенням для багатьох компаній та розробників.
- Microsoft Azure: Azure від Microsoft є іншою популярною хмарною платформою, яка надає широкий спектр послуг, включаючи хмарне зберігання, віртуальні машини, аналітичні сервіси та інтеграцію з інструментами Microsoft.
- Google Cloud Platform (GCP): GCP від Google пропонує хмарні послуги, такі як обчислення, зберігання даних, штучний інтелект та аналітика. Він володіє широким функціоналом та добре підходить для розробки масштабних проектів.
- IBM Cloud: IBM Cloud є хмарною платформою, яка надає послуги хмарного обчислення, зберігання даних, штучного інтелекту та інтеграції. Вона володіє потужними функціональними можливостями та підходить для різноманітних сценаріїв розробки.

- Oracle Cloud: Oracle Cloud пропонує послуги хмарного зберігання, бази даних, обчислення та аналітику. Вона спеціалізується на розробці індустріальних рішень та надійності.

Вибір цих хмарних платформ базується на їх популярності, широкому спектрі послуг та надійності. Вони забезпечують гнучкість та масштабованість, що дозволяє розробникам ефективно працювати з різними типами проектів та обсягами даних. Крім того, ці хмарні платформи надають розширені функціональні можливості, такі як штучний інтелект, аналітика та інструменти розробки, що дозволяє розробникам будувати потужні та інноваційні додатки.

Хмарні технології можуть бути безпечними, але вони також мають свої потенційні ризики, які потрібно враховувати. Ось декілька аспектів безпеки, які пов'язані з хмарними технологіями:

- Захист даних: Хмарні постачальники повинні забезпечувати ефективні механізми захисту даних, такі як шифрування, контроль доступу та захист від несанкціонованого доступу. Важливо переконатися, що постачальники хмарних послуг мають належні механізми безпеки, які відповідають найвищим стандартам та регуляціям.
- Приватність даних: Користувачі хмарних технологій мають право на захист приватності своїх даних. Важливо переконатися, що постачальники хмарних послуг дотримуються відповідних нормативних вимог та політик щодо захисту приватності.
- Надійність та доступність: Хмарні сервіси повинні бути надійними та доступними для користувачів. Це означає, що інфраструктура хмари має бути стійкою до відмов та має бути забезпечена можливість резервного копіювання та відновлення даних.
- Сертифікація та стандартизація: Існування сертифікаційних організацій та стандартів допомагає забезпечити безпеку хмарних технологій.

Користувачі можуть спиранняться на сертифікати безпеки та відповідність стандартам, які випускаються такими організаціями.

- Свідомість користувачів: Нарешті, свідомість та освіченість користувачів щодо безпеки хмарних технологій важлива. Користувачі повинні бути обізнані з базовими принципами безпеки, такими як сильні паролі, оновлення програмного забезпечення та використання двофакторної аутентифікації.

Існує кілька заходів та механізмів, які використовуються для забезпечення безпеки даних та інфраструктури хмарних послуг.

Ось декілька з них:

1. Шифрування даних: Використання шифрування є одним з основних засобів захисту даних в хмарному середовищі. Дані можуть бути шифровані під час передачі та зберігання, що забезпечує їх конфіденційність та цілісність.

2. Аутентифікація та авторизація: Для забезпечення безпеки важливо використовувати ефективні методи аутентифікації та авторизації. Це включає використання сильних паролів, двофакторної аутентифікації та керування доступом до ресурсів.

3. Захист мережі: Хмарні постачальники повинні застосовувати заходи безпеки на рівні мережі, такі як мережеві брандмауери, виявлення та запобігання вторгненням (IDS/IPS), щоб захистити мережевий трафік від несанкціонованого доступу та зловживань.

4. Резервне копіювання та відновлення даних: Важливо мати механізми резервного копіювання та відновлення даних, щоб забезпечити можливість відновлення в разі втрати або пошкодження даних.

5. Аудит та моніторинг: Регулярний аудит безпеки та моніторинг дозволяють виявляти потенційні загрози та аномальну активність в хмарному середовищі. Це допомагає вчасно виявляти і реагувати на можливі безпекові проблеми.

6. Сертифікація та відповідність: Деякі хмарні постачальники можуть отримувати сертифікати безпеки та відповідність стандартам безпеки, таким як ISO 27001. Це свідчить про те, що вони виконують високі стандарти безпеки.

Ці заходи та механізми, використовуючи сучасні технології та передові практики безпеки, спрямовані на забезпечення безпеки даних та інфраструктури в хмарному середовищі. Проте, важливо враховувати, що безпека хмарних технологій є спільною відповідальністю постачальника та користувача, і обидва мають приймати необхідні заходи для забезпечення безпеки своїх даних та ресурсів.

2.4 Машинне навчання та штучний інтелект

2.4.1 Загальний огляд штучного інтелекту та машинного навчання

Штучний інтелект (ШІ) є однією з найбільш захоплюючих та швидкорозвиваючих галузей сучасної науки і технологій. Він вивчає та моделює інтелектуальну поведінку та здатність до розуміння, навчання та прийняття рішень, яку виявляють люди. Штучний інтелект заснований на використанні комп'ютерних алгоритмів та моделей, які дозволяють системам "мислити" та "розуміти" інформацію, а також виконувати складні завдання, які раніше вважалися привілеєм людей.

Одним з ключових понять в штучному інтелекті є поняття "машинного навчання". Це підполе ШІ, яке вивчає алгоритми та методи, що дозволяють комп'ютерним системам навчатися на основі даних та здійснювати прогнози, робити висновки та приймати рішення без явного програмування. Машинне навчання використовується в багатьох сферах, таких як медицина, фінанси, автоматизація виробництва, розпізнавання образів та багато інших.

Штучний інтелект також включає в себе інші підполя, такі як обробка природної мови (Natural Language Processing), комп'ютерний зір (Computer Vision), розпізнавання мови (Speech Recognition) та експертні системи (Expert Systems). Ці підполя займаються вивченням та розробкою алгоритмів, які

дозволяють комп'ютерним системам розуміти та взаємодіяти з людьми через мову, зображення та звук.

Машинне навчання один з розділів AI, алгоритми, що дозволяють комп'ютеру робити висновки на підставі даних, не слідуючи жорстко заданим правилами. Тобто машина може знайти закономірність у складних і багато-параметричних завданнях (які мозок людини не здатен вирішити), таким чином знаходячи більш точні відповіді. Як результат – правильне прогнозування.

Машинне навчання використовується в різних сферах, таких як медицина, фінанси, маркетинг, автоматизація виробництва та багато інших. Його застосування можна знайти у розпізнаванні образів, обробці природної мови, рекомендаційних системах, робототехніці та інших областях. Завдяки машинному навчанню комп'ютерні системи здатні адаптуватися до змінних умов, виявляти складні залежності в даних та покращувати свою продуктивність з досвідом.

У машинному навчанні ключову роль відіграють алгоритми навчання. Вони дозволяють системі "навчитися" на основі великого обсягу даних, а потім використовувати ці знання для вирішення нових завдань або роботи з новими даними. До популярних алгоритмів належать нейронні мережі, дерева рішень, метод опорних векторів та багато інших.

Процес машинного навчання включає кілька етапів, таких як збір та підготовка даних, вибір моделей та алгоритмів, навчання, валідацію та тестування моделей. Важливим аспектом є також питання оцінки точності та ефективності моделей, а також управління перенавчанням (overfitting) та недонавчанням (underfitting).

При використанні машинного навчання також важливо враховувати етичні та правові аспекти. Питання приватності даних, справедливості алгоритмів та можливості появи системного упередження потребують уваги та відповідального підходу. Необхідно розробляти моделі та алгоритми, які

будуть враховувати соціальні та етичні норми, а також забезпечувати прозорість та відповідальність у прийнятті рішень.

За ознакою наявності вчителя, навчання ділиться на навчання з учителем (Supervised Learning), без вчителя (Unsupervised Learning) та з підкріпленням (Reinforcement Learning):

- Навчання з учителем застосовують коли потрібно навчити машину розпізнавати об'єкти або сигнали. Загальний принцип навчання з учителем це "дивись, ось це двері і це теж двері, і ось це теж двері"
- Навчання без вчителя використовує принцип "ця річ така ж як інші". Алгоритми вивчають подібності і можуть виявити відмінність і виконати виявлення аномалій, розпізнаючи, що є незвичайним або несхожим.
- Навчання з підкріпленням використовують там, де перед машиною стоїть завдання - правильно виконати поставлені завдання у зовнішньому середовищі маючи безліч можливих варіантів дії Наприклад в комп'ютерних іграх, трейдингових операціях, для безпілотної техніки.

За типом застосовуваних алгоритмів можна виділити два види:

А) класичне навчання - відомі і добре вивчені алгоритми навчання, розроблені в основному більше 50-ти років тому для статистичних бюро. Підходить, насамперед, під завдання роботи з даними: класифікація, кластеризація, регресія і т.п. Застосовують для прогнозування, сегментації клієнтів і так далі.

Б) нейронні мережі і глибоке навчання - найбільш сучасний підхід до машинного навчання. Нейронні мережі застосовуються там, де потрібні розпізнавання або генерація зображень і відео, складні алгоритми управління або прийняття рішень, машинний переклад і подібні складні завдання.

2.4.2 Дослід використання штучного інтелекту та машинного навчання

Як для прикладу можна зробити короткий дослід про змогу машинного навчання та штучного інтелекту відрізнити кота від собаки. Штучний

інтелект може бути навчений розрізняти об'єкти, включаючи собак і котів, за допомогою методів комп'ютерного зору і машинного навчання. За допомогою великої кількості зображень собак і котів, система штучного інтелекту може навчитися розпізнавати унікальні риси кожного виду тварини, такі як форма тіла, розмір, функції обличчя та інші характеристики.

Однак, точність розпізнавання залежить від якості навчання моделі, доступності репрезентативних даних та використовуваних алгоритмів. Якщо система навчена на великому наборі якісних зображень, вона зможе розпізнавати собак і котів з високою точністю. Однак, існують випадки, коли система може зробити помилку і неправильно класифікувати об'єкти.

Таким чином, штучний інтелект може відрізняти собаку від кота з певною ймовірністю і залежно від якості навчання моделі та характеристик зображень. Точність розпізнавання може покращуватися з використанням більш складних алгоритмів, більш репрезентативних даних та додаткової обробки інформації.

Важливо зауважити, що штучний інтелект розробляється вперше для виконання певних завдань і не має здатності до загального розуміння, як у людей. Тому, хоча штучний інтелект може відрізняти собаку від кота на підставі візуальних ознак, він не має глибокого розуміння поняття собаки та кота, їх поведінки, контекстуального та візуального значення.

2.4.3 Користь впровадження штучного інтелекту та машинного навчання

Ефективно управляти бізнесом без впровадження сучасних комп’ютерних технологій вже неможливо. Але частина компаній боїться додаткових витрат на сучасне програмне забезпечення, розроблене на основі штучних нейронних мереж з використанням технологій штучного інтелекту, великих даних і машинного навчання, працює по-старому і втрачає ефективність. Компанії ж, які впроваджують новітні досягнення науки, отримують конкурентні переваги, за рахунок швидкого прийняття найбільш ефективних рішень.

Щоб в програми побудовані на застарілих принципах ввести нові алгоритми, потрібно кожен раз запрошувати фахівця і оплачувати його працю. Програмне забезпечення з функцією машинного навчання створює ці алгоритми самостійно.

Разова інвестиція в такі програмні рішення швидко окупиться за рахунок прийому правильних рішень і скорочення витрат на послуги програмістів. Нові товари і послуги з новими споживчими характеристиками надходять на ринок безперервно. Умови ведення бізнесу швидко змінюються. Потреба в нових алгоритмах з'являється постійно.

Технологія машинного навчання може бути застосована у всіх областях ведення бізнесу, в державному управлінні, всюди де потрібне швидке прийняття рішень в умовах мінливої обстановки:

- На основі штучних нейронних мереж можна, створити високоякісні системи:
- Управління бізнесом, незалежно від конкретного його напрямки.
- Управління транспортом і перевезеннями, що підвищують пропускну здатність доріг, що збільшують швидкість доставки і скорочують транспортні витрати.
- Розпізнавання осіб і образів, здатні не тільки виявляти необхідний об'єкт за результатами відеофіксації, а й, наприклад, читати по губах. Це відкриває дорогу використанню такого програмного забезпечення в поліції та інших державних відомствах і органах.
- Проєктування і моделювання, як матеріальних об'єктів, так і ситуацій.
- Діагностики та лікування різних захворювань.

Фактично, побудовані за цим принципом, системи можуть бути використані в будь-якій сфері людської діяльності. Функція машинного навчання може бути вбудована в деякі чинні програмні рішення, які раніше її не мали.

2.5 Блокчейн та криптовалюти

Блокчейн - це розподілена технологія збереження та передачі інформації, що базується на концепції ланцюга блоків. Вона стала відомою завдяки криптовалюті Bitcoin, але зараз використовується в багатьох інших сферах.

Основна ідея блокчейна полягає в тому, що дані зберігаються в блоках, які поступово додаються до ланцюга. Кожен блок містить хеш попереднього блока, що гарантує безпеку та цілісність даних. Коли новий блок додається до ланцюга, він стає постійною та необоротною частиною системи.

За вимогу до хешів блоків відповідає спеціальний параметр, званий «складність».Оскільки обчислювальні потужності мережі непостійні, цей параметр перераховується клієнтами мережі через кожні 2016 блоків таким чином, щоб підтримувати середню швидкість формування розподіленої БД на рівні 2016 блоків за два тижні. Таким чином 1 блок повинен створюватися приблизно раз на десять хвилин. На практиці, коли обчислювальна потужність мережі зростає — відповідні часові проміжки коротші, а коли знижується — довші

Одна з основних переваг блокчейна - це децентралізація. Вона дозволяє уникнути потреби в посередниках та сторонніх учасниках, оскільки кожен вузол мережі має копію ланцюга блоків і може перевірити та підтвердити транзакції самостійно. Це забезпечує безпеку та надійність системи, оскільки зламати або змінити дані в кожному блоку вимагає великої обчислювальної потужності та контролю більшості мережі.

Блокчейн також має можливості для "розумних контрактів". Це програми, які автоматично виконують умови, зазначені в контракті, коли певні умови виконуються. Це дозволяє створювати автоматизовані та безпечні транзакції без необхідності в довіреній стороні.

Блокчейн знаходить застосування у різних галузях, включаючи фінанси, логістику, ланцюг постачання, охорону здоров'я, громадську безпеку та

багато інших. Він дозволяє створювати довірчі, безпечні та ефективні системи обміну даними та виконання транзакцій.

Проте, варто враховувати, що блокчейн не є універсальним рішенням для всіх проблем. Він має свої обмеження, такі як швидкодія та масштабованість, які потрібно враховувати при розгляді використання цієї технології.

Смарт-контракти - це автоматизовані програмні коди, які запускаються на блокчейні і мають можливість виконувати, перевіряти та контролювати угоди або транзакції без потреби в посередниках. Вони є одним з ключових компонентів технології блокчейн.

Смарт-контракти базуються на принципі "якщо-тоді" (if-then). Вони містять умови, які визначають, які дії повинні бути виконані, якщо визначені умови стають правдивими. Наприклад, якщо виконується певна умова, наприклад, отримання платежу, то виконується певна дія, наприклад, передача товару.

Смарт-контракти дозволяють сторонам укладати угоди та виконувати їх автоматично, без необхідності в довірених посередниках або централізованих організаціях. Вони забезпечують безпеку, недоступність для змін, прозорість та недискримінацію в угодах.

Однією з відомих платформ для розробки смарт-контрактів є Ethereum, але існують й інші блокчейн-платформи, які підтримують смарт-контракти. Важливо враховувати, що смарт-контракти мають свої обмеження і вимагають ретельного проектування та аудиту для забезпечення їх безпеки та правильної функціональності.

Написання смарт-контрактів може здійснюватися різними мовами програмування: Solidity, Serpent, Mutan та інші. Але найпопулярнішим з них все ж таки є Solidity.

Приклади використання блокчейну та смарт-контрактів включають:

- Фінансові послуги: Блокчейн може використовуватись для швидкого, безпечного та безпосереднього переказу грошей або цифрових

активів між користувачами. Смарт-контракти дозволяють автоматично виконувати умови платежів, уникнувши потреби в посередниках.

- Логістика та ланцюг постачання: Блокчейн може відстежувати переміщення товарів від виробника до кінцевого споживача. Це дозволяє забезпечити прозорість, автентичність та ефективність у логістичних процесах. Смарт-контракти можуть автоматично перевіряти умови доставки та оплати.
- Охорона здоров'я: Блокчейн може забезпечувати безпечний обмін медичних даних між різними провайдерами охорони здоров'я. Це дозволяє пацієнтам контролювати свої дані та забезпечувати конфіденційність. Смарт-контракти можуть автоматично виконувати угоди щодо обробки медичних послуг або страхових виплат.
- Голосування: Блокчейн може забезпечити безпечне та прозоре голосування. Кожен голос може бути записаний в блокчейні, що гарантує неможливість зміни результатів. Смарт-контракти можуть автоматично перевіряти правомірність голосів та підраховувати результати.
- Інтелектуальна власність: Блокчейн може забезпечувати захист та управління правами інтелектуальної власності, такої як авторські права або патенти. Смарт-контракти можуть автоматично управляти доступом та ліцензуванням інтелектуальних активів.

Ці приклади лише деяких з багатьох можливостей використання блокчейну та смарт-контрактів. Вони демонструють потенціал цих технологій для створення безпечних, ефективних та децентралізованих систем у різних сферах життя.

До переваг смарт-контрактів відносять:

- Автономність. Для затвердження договору не потрібна юридична особа, тож сторони можуть обійтися без посередництва. Контракт сам ухвалює рішення на підставі закладених у нього умов. Він або завершить угоду та видасть сторонам необхідне, або накладе на учасників санкції чи заблокує кошти.

- Мінімізація витрат. Оскільки присутність третьої сторони не потрібна, смарт-контракти вкладати вигідніше, ніж звертатися до спеціалістів.
- Швидкість. Смарт-контракти створюються за допомогою коду, що дозволяє уникнути бюрократичної тяганини. Це сприяє швидшій обробці інформації та швидшому виконанню угоди.
- Безпека. Децентралізоване управління зменшує ризик маніпуляцій, оскільки вся інформація записується в блокчейн та не може бути звідти видалена або відформатована.
- Автоматизація. Розумні контракти можуть застосовуватися для автоматизації рутинних процесів, оскільки вони виконуються та підтверджуються самостійно.
- Точність. Результат роботи контракту завжди буде однаковим незалежно від того, хто виконує прописані умови. Головне – щоб ці умови було виконано.

Попри те, що смарт-контракти самі по собі чітко виконують прописані в них умови, все ще існує загроза шахрайства. Нижче наведені ризики, з якими ризиками можна зіткнутися:

- Надійність договору залежить від розробника, який його пише. Дві та більше сторони можуть домовитися про одне, але розробник може прописати у коді трохи інші умови. Тому краще самому перевірити ще раз код контракту або звернутися до фахівця, а не безумовно покладатися на автора смарт-контракту. Також деякі компанії пропонують послуги аудиту розумних контрактів та їх тестування.
- Не існує центрального органу, який би регулював смарт-контракти. Тому у випадках шахрайства мало що можна зробити, щоб скасувати дію контракту.
- Також розумні контракти не регулюються державою, тому вона теж не зможе втрутитися, якщо користувача ошукають.

2.6 Віртуальна та доповнена реальність

Віртуальна та доповнена реальність (VR та AR) є одними з найцікавіших технологій програмування, які змінюють спосіб, яким ми сприймаємо і взаємодіємо з цифровим світом. Ці технології надають можливість створювати іммерсивні та захоплюючі віртуальні середовища, де користувач може взаємодіяти з об'єктами та інформацією в реальному часі.

Віртуальна реальність передає користувачеві враження про перебування в іншому світі, повністю відокремленому від фізичної реальності. Віртуальна реальність (VR) - це технологія, що створює іммерсивне віртуальне середовище, яке відокремлює користувача від реального світу і переносить його в цифровий простір. За допомогою спеціальних VR-пристроїв, таких як шоломи або віртуальні очки, користувач може побачити та взаємодіяти з віртуальними об'єктами і сценами.

Одним з ключових аспектів VR є створення віртуальної реалістичності та іммерсії для користувача. Це досягається за допомогою потужного обчислювального обладнання та передових графічних можливостей. VR-пристрої відстежують рухи голови та тіла користувача, щоб забезпечити відповідну взаємодію з віртуальним світом.

Віртуальна реальність застосовується у різних галузях. У галузі розваг VR дозволяє користувачам зануритися у віртуальні ігрові світи, де вони можуть взаємодіяти з персонажами та об'єктами. Також VR використовується в медицині для тренування хірургів, лікування фобій та психічних захворювань, архітектурі для візуалізації будівельних проектів, освіті для інтерактивного навчання та багатьох інших сферах.

Розробка VR додатків вимагає спеціалізованих інструментів та технологій. Одні з найпопулярніших платформ розробки VR включають Unity та Unreal Engine, які надають потужні інструменти для створення віртуальних світів.

Важливими аспектами в розробці VR є оптимізація продуктивності та забезпечення безпеки. Оскільки VR вимагає великого обсягу обчислювальних

ресурсів, важливо забезпечити плавну роботу додатків на різних пристроях. Крім того, слід враховувати фізичну безпеку користувачів, забезпечуючи правильну інтеракцію зі віртуальними об'єктами та запобігаючи можливим травмам.

Доповнена реальність, з свого боку, поєднує віртуальний світ з реальними об'єктами та оточенням.

Доповнена реальність (AR) - це технологія, що дозволяє накладати віртуальні об'єкти та інформацію на реальний світ. Вона поєднує реальність і віртуальність, дозволяючи користувачам сприймати і взаємодіяти з віртуальними елементами у реальному середовищі.

Основною особливістю AR є те, що вона додає додаткову інформацію до реального світу, збагачуючи сприйняття користувача. Це може бути відображення тексту, зображень, анімації, моделей або навіть інтерактивних об'єктів. AR-технології використовуються на мобільних пристроях, спеціальних пристроях AR та навіть на окулярах.

Доповнена реальність знаходить широке застосування у різних галузях. В сфері маркетингу та реклами AR використовується для створення інтерактивних рекламних кампаній, де користувачі можуть взаємодіяти з продуктами або переглядати віртуальні каталоги. У медицині AR використовується для тренування хірургів, діагностики та планування операцій. AR також застосовується в галузі освіти, дизайну, туризму, ремонту та обслуговування техніки.

Доповнена реальність відкриває безліч можливостей для інтерактивного взаємодії зі світом, дозволяючи створювати захоплюючі інтерфейси та перетворюючи спосіб сприйняття інформації. Вона знаходить все більше застосувань у різних галузях, пропонуючи нові способи комунікації, навчання та розваг.

Що стосується програмування для VR та AR, існують спеціальні платформи та мови програмування, які дозволяють розробляти додатки для цих технологій. Наприклад, Unity та Unreal Engine - популярні інтегровані

середовища розробки, що підтримують розробку VR та AR додатків. Мови програмування, такі як C#, C++, JavaScript, Python та інші, використовуються для реалізації функціональності додатків та взаємодії зі середовищем.

Програмування віртуальної та доповненої реальності вимагає використання спеціалізованих інструментів та платформ, які допомагають розробникам створювати віртуальні середовища та інтерактивні додатки.

Основні етапи програмування включають наступне:

- Вибір платформи: Для розробки віртуальної та доповненої реальності можна використовувати різні платформи, такі як Unity, Unreal Engine, ARKit (для iOS) або ARCore (для Android). Вибір платформи залежить від цільової платформи, функціональних потреб та вимог проекту.
- Моделювання та створення віртуальних об'єктів: Віртуальна реальність передбачає створення тривимірних об'єктів та їх візуалізацію. Для цього використовуються спеціалізовані інструменти для моделювання, такі як Blender або Autodesk Maya. Розробники створюють моделі об'єктів, текстури, освітлення та анімацію.
- Розробка логіки та функціональності: Після створення віртуальних об'єктів розробники програмують логіку та функціональність додатків. Вони використовують мови програмування, такі як C#, JavaScript або C++, для створення скриптів, які контролюють поведінку об'єктів, взаємодію з користувачем та обробку даних.
- Візуалізація та рендеринг: Рендеринг є важливою частиною програмування віртуальної реальності. Розробники використовують графічні движки та бібліотеки, які допомагають відображати віртуальні об'єкти з реалістичною графікою. Це включає освітлення, текстури, ефекти та анімацію.
- Тестування та оптимізація: Після програмування додатку важливо провести тестування та оптимізацію для забезпечення плавної та ефективної роботи. Розробники перевіряють взаємодію з об'єктами, функціональність, якість графіки та продуктивність.

- Розгортання та випуск: Після успішного тестування розробник готовий розгорнути додаток на відповідних платформах, таких як віртуальні навушники або мобільні пристрої, і надати доступ користувачам.

Програмування для VR та AR вимагає розуміння особливостей цих технологій, а також врахування аспектів, пов'язаних з відтворенням 3D-графіки, взаємодією зі сценою, детектуванням руху та багатьма іншими. Крім того, потрібно мати на увазі обмеження, пов'язані з обладнанням, продуктивністю та ефективним використанням ресурсів.

2.7 Інформаційна безпека. Криптографія та шифрування

2.7.1 Загальний огляд

Якщо говорити про вищесказані засоби та технології програмування, можна дійти висновку, що сучасний світ розвивається з неймовірними темпами. Але аналізуючи дані сфери, можна стверджувати що одним з найважливіших аспектів такого швидкого темпу розвитку є інформаційна безпека. Зростання обсягу цифрових даних, їх обмін і зберігання створюють значні виклики для забезпечення конфіденційності, цілісності та доступності інформації. Крім того, з появою нових технологій, таких як хмарні обчислення, штучний інтелект (AI) і блокчейн, з'являються нові вектори загроз і нові вимоги до захисту інформації.

Інформаційна безпека як технологія програмування зосереджується на застосуванні набору методів, практик і механізмів для захисту конфіденційності, цілісності та доступності інформаційних ресурсів. Ось декілька основних методів забезпечення безпеки інформації:

- Аутентифікація: Цей метод використовується для перевірки та підтвердження ідентичності користувачів, пристроїв або систем. Включає в себе використання паролів, біометричних даних, токенів або інших методів для перевірки прав доступу.

- Авторизація: Цей метод контролює права доступу користувачів або систем до різних ресурсів. Використовується для обмеження доступу до конфіденційної інформації та функціональності системи.
- Шифрування: Шифрування використовується для захисту конфіденційності даних шляхом перетворення їх у незрозумілу форму за допомогою шифрувального алгоритму. Для доступу до даних потрібен ключ розшифрування.
- Відновлення даних: Регулярне створення резервних копій даних та відновлення їх у разі втрати або пошкодження є важливим методом забезпечення доступності та цілісності інформації.
- Фізична безпека: Цей метод включає в себе захист фізичного середовища, в якому знаходяться інформаційні ресурси. Включає в себе захист серверних кімнат, обмеження фізичного доступу до обладнання та контроль відеоспостереження.
- Мережева безпека: Захист мережі та комунікаційних каналів від несанкціонованого доступу, перехоплення даних або атак з метою злому системи.
- Аудит безпеки: Метод аудиту безпеки включає систематичне перевіряння та аналіз системи з метою виявлення потенційних вразливостей, недоліків в безпеці та моніторингу подій для виявлення аномалій.
- Навчання та свідомість: Освіта та навчання користувачів про потенційні загрози, техніки соціальної інженерії та кращі практики щодо забезпечення безпеки є важливими елементами забезпечення безпеки інформації.

Забезпечення інформаційної безпеки вимагає постійного оновлення і адаптації до змінних загроз та ризиків. Розробники програмного забезпечення відіграють важливу роль у цьому процесі, використовуючи сучасні засоби та технології програмування для створення безпечних і надійних програмних продуктів. Вони враховують принципи безпеки в кожній стадії розробки,

використовуючи надійні алгоритми шифрування, перевіряючи вразливості, встановлюючи заходи захисту і забезпечуючи безпеку передачі та зберігання даних.

Застосування інформаційної безпеки є критично важливим у сучасному світі, де обмін і зберігання інформації в електронному форматі стає все більш поширеним. Забезпечення безпеки інформації включає різні аспекти і методи, спрямовані на захист конфіденційності, цілісності та доступності інформаційних ресурсів. Ось деякі ключові області застосування інформаційної безпеки:

- Комп'ютерна безпека: Це включає застосування заходів для захисту комп'ютерних систем від несанкціонованого доступу, злому та крадіжки інформації. Це включає в себе використання міцних паролів, аутентифікацію, шифрування даних, вогнезахисних систем та антивірусного програмного забезпечення.
- Мережева безпека: Забезпечення безпеки мережі включає захист мережевих інфраструктур, таких як мережеві пристрої, комутатори, маршрутизатори, а також безпечне підключення і передачу даних через мережі. Це включає в себе використання мережевих фаєрволів, віртуальних приватних мереж (VPN), інтранетів та інших методів захисту мережевої інфраструктури.
- Криптографія: Криптографія використовується для шифрування даних з метою забезпечення їх конфіденційності та недоступності несанкціонованим особам. Вона використовує різні алгоритми шифрування, такі як симетрична криптографія (один ключ для шифрування та розшифрування) та асиметрична криптографія (публічний та приватний ключі). Криптографія використовується в електронному цифровому підписі, VPN, електронній комерції та багатьох інших аспектах інформаційної безпеки.
- Фізична безпека: Фізична безпека включає захист фізичних ресурсів, таких як комп'ютери, сервери, дата-центри, системи зберігання

даних тощо. Це включає в себе контроль доступу до приміщень, використання систем відеоспостереження, біометричні системи та інші фізичні заходи безпеки.

- Соціальна інженерія: Соціальна інженерія відноситься до маніпулювання людьми з метою отримання несанкціонованого доступу до інформації або систем. Застосування соціальної інженерії включає навчання персоналу щодо розпізнавання шахрайства, фішингових атак, викрадання даних тощо.

Захист інформації стає все більш важливим в сучасному цифровому світі, де зловмисники постійно шукають нові способи зламу інформаційних систем. Інформаційна безпека як технологія програмування допомагає забезпечити захист інформації та забезпечити надійність та безпеку програмного забезпечення в цьому постійно змінюючомуся середовищі.

2.7.2 Приклади застосування

В даному розділі буде розглянуто наявно кілька прикладів застосування сучасних технологій для забезпечення інформаційної безпеки. Першим прикладом буде застосування криптографії.

Сучасна криптографія є невід'ємною складовою інформаційної безпеки і використовується для захисту конфіденційності, цілісності та автентичності даних. Вона займається шифруванням та розшифруванням інформації з метою забезпечення її конфіденційності, а також створенням та перевіркою електронних підписів для забезпечення автентичності та цілісності даних.

Основні принципи та методи сучасної криптографії включають:

- Симетричне шифрування: Використовується один ключ для шифрування та розшифрування повідомлень. Прикладами алгоритмів симетричного шифрування є AES (Advanced Encryption Standard) і DES (Data Encryption Standard).

- Асиметричне шифрування: Використовується пара ключів - публічний та приватний. Публічний ключ використовується для шифрування повідомлень, а приватний ключ - для розшифрування. Асиметричне шифрування також використовується для створення та перевірки електронних підписів. Найпоширеніші алгоритми асиметричного шифрування - RSA, DSA та ECC (еліптичні криві).
- Хеш-функції: Використовуються для перетворення повідомлень будь-якої довжини в фіксовану довжину хеш-коду. Хеш-функції використовуються для перевірки цілісності даних та створення цифрових підписів. Прикладами хеш-функцій є SHA-256 та MD5.
- Протоколи обміну ключами: Використовуються для безпечного обміну секретними ключами між сторонами комунікації. Найвідоміші протоколи обміну ключами - Diffie-Hellman та RSA.
- Цифрові підписи: Використовуються для перевірки автентичності та цілісності повідомлень. Вони базуються на асиметричних шифрах та хеш-функціях. Цифровий підпис підтверджує, що повідомлення було створене відповідною особою і не було змінене.
- Криптографічні протоколи: Використовуються для забезпечення безпеки при обміні даними між сторонами, наприклад, протоколи SSL/TLS для безпечного з'єднання між веб-сервером і браузером.

Для дослідження цього напрямку було розроблено демонстраційну програму, що показує застосування сучасних криптографічних алгоритмів. Вона використовує криптографічну бібліотеку Bouncy Castle (включає в себе широкий спектр криптографічних алгоритмів, таких як шифри блочного шифрування (AES, Blowfish, DES), шифри потокового шифрування, алгоритми хешування (SHA-1, SHA-256, MD5), цифрові підписи (RSA, DSA, ECDSA), генерація ключів, розподілені протоколи ключів та багато іншого.) для реалізації алгоритмів шифрування та розшифрування повідомлень.

```
class CryptographyDemo {

    public void main(String[] args) throws Exception {
        // Згенеруємо секретний ключ для шифрування
        KeyGenerator keyGenerator = KeyGenerator.getInstance("AES");
        keyGenerator.init( keysize: 256);
        SecretKey secretKey = keyGenerator.generateKey();
```

Рис.2.7.2.1 – демонстрація генерації ключа

Наведений код є прикладом використання криптографічних функцій в мові програмування Java для шифрування та розшифрування повідомлень за допомогою алгоритму AES (Advanced Encryption Standard) у режимі CBC (Cipher Block Chaining).

Створення секретного ключа показано на Рис.2.7.2. За допомогою класу KeyGenerator створюється секретний ключ для шифрування. У даному випадку використовується алгоритм AES з довжиною ключа 256 біт.

В наступному блоці програми на Рис.2.7.2.2 показано шифрування повідомлення.

```
public byte[] encrypt(String message, SecretKey secretKey) throws Exception {
    Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
    cipher.init(Cipher.ENCRYPT_MODE, secretKey);

    byte[] encryptedBytes = cipher.doFinal(message.getBytes(StandardCharsets.UTF_8));

    // Отримаємо вектор ініціалізації з шифрувальника
    byte[] iv = cipher.getIV();

    // Комбінуємо вектор ініціалізації та зашифровані дані
    byte[] encryptedMessage = new byte[iv.length + encryptedBytes.length];
    System.arraycopy(iv, srcPos: 0, encryptedMessage, destPos: 0, iv.length);
    System.arraycopy(encryptedBytes, srcPos: 0, encryptedMessage, iv.length, encryptedBytes.length);

    return encryptedMessage;
}
```

Рис.2.7.2.2 – Шифрування повідомлення

Шифрування повідомлення: Функція encrypt приймає повідомлення та секретний ключ. Створюється екземпляр Cipher з вказаною конфігурацією AES/CBC/PKCS5Padding та ініціалізується для шифрування з використанням секретного ключа. Повідомлення перетворюється в масив байтів, який потім шифрується. Додатково, отримується вектор ініціалізації (IV), який комбінується з зашифрованими даними.

```
public String decrypt(byte[] encryptedMessage, SecretKey secretKey) throws Exception {
    byte[] iv = new byte[16];
    byte[] encryptedBytes = new byte[encryptedMessage.length - iv.length];
    System.arraycopy(encryptedMessage, srcPos: 0, iv, destPos: 0, iv.length);
    System.arraycopy(encryptedMessage, iv.length, encryptedBytes, destPos: 0, encryptedBytes.length);

    Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
    cipher.init(Cipher.DECRYPT_MODE, secretKey, new IvParameterSpec(iv));

    byte[] decryptedBytes = cipher.doFinal(encryptedBytes);
    return new String(decryptedBytes, StandardCharsets.UTF_8);
}
```

Рис.2.7.2.3 – Дешифрування повідомлення

Розшифрування повідомлення: Функція decrypt приймає зашифроване повідомлення та секретний ключ. Розшифрування включає розбиття зашифрованого повідомлення на вектор ініціалізації (IV) та зашифровані байти. За допомогою Cipher розшифровуються дані з використанням секретного ключа та вектора ініціалізації.

Зашифроване повідомлення має вигляд такого типу: l+6SfrwXCVe9RkU4IRK6Z6lQmePTy8V7RAFz9SJ8br4=, а оригінальне та розшифроване будуть однакові.

В даній програмі було продемонстровано простий приклад застосування сучасної криптографії для зашифрування та розшифрування повідомлень з використанням алгоритму AES (Advanced Encryption Standard) в режимі CBC (Cipher Block Chaining).

Програма починається з генерації секретного ключа за допомогою KeyGenerator. Потім вона шифрує задане повідомлення за допомогою методу

encrypt та розшифровує його за допомогою методу decrypt, використовуючи секретний ключ.

Ця програма демонструє простий, але ефективний спосіб застосування криптографії для забезпечення конфіденційності повідомлень. Застосування сучасних криптографічних алгоритмів, таких як AES, сприяє захисту важливої інформації від несанкціонованого доступу.

Для наступного прикладу було обрано одслідження електронного цифрового підпису.

Електронний цифровий підпис (Electronic Digital Signature) є одним з ключових елементів інформаційної безпеки та криптографії. В сучасному цифровому світі, де обмін електронною інформацією стає все поширенішим, важливо мати засоби, що дозволяють підтвердити автентичність та цілісність цифрових документів. Електронний цифровий підпис забезпечує цю можливість, гарантуючи, що дані не були змінені під час передачі та що вони походять від певної особи чи організації.

Електронний цифровий підпис базується на принципах криптографії, де використовуються симетричні або асиметричні криптографічні алгоритми. Симетрична криптографія використовує спільний секретний ключ для підпису та перевірки підпису, тоді як асиметрична криптографія використовує пару ключів: приватний ключ для підпису та публічний ключ для перевірки підпису.

Процес створення електронного цифрового підпису включає кілька кроків. Спочатку, особа або організація, що бажає підписати документ, використовує криптографічний алгоритм для обчислення хеш-функції документа, що представляє собою унікальний числовий ідентифікатор цього документа. Потім, вони застосовують свій приватний ключ до хеш-функції для отримання цифрового підпису, який є унікальним для даного документа та особи, яка його підписала.

Перевірка електронного цифрового підпису включає зворотній процес. Отримувач документа використовує публічний ключ особи, яка підписала

документ, для перевірки підпису. Він обчислює хеш-функцію документа та порівнює його з отриманим підписом, використовуючи публічний ключ. Якщо хеш-значення співпадає з підписом, то це свідчить про те, що документ не був змінений під час передачі та що він був підписаний вірною особою.

Електронний цифровий підпис забезпечує довіру, автентичність та неденійність електронних документів. Він застосовується в різних сферах, таких як електронна комерція, електронний документообіг, електронний банкінг, електронний уряд тощо. Електронний цифровий підпис є важливим інструментом для забезпечення безпеки та довіри в цифровому середовищі.

Отже, електронний цифровий підпис є ефективним засобом забезпечення безпеки та автентичності цифрових документів. Він забезпечує перевірку цілісності та автентичності даних, що передаються в електронній формі, і гарантує, що вони походять від відповідної особи чи організації. Використання електронного цифрового підпису сприяє безпечному обміну інформацією та забезпечує довіру в цифровому світі.

Основні кроки програми наступні:

- Генерація ключової пари: У методі generateKeyPair() використовується KeyPairGenerator, який ініціалізується для алгоритму RSA з ключовою довжиною 2048 біт. Потім генерується ключова пара (приватний та публічний ключі), показано на Рис2.7.2.4

```
public KeyPair generateKeyPair() throws NoSuchAlgorithmException {
    // Генеруємо ключову пару RSA
    KeyPairGenerator keyGen = KeyPairGenerator.getInstance("RSA");
    keyGen.initialize( keysize: 2048);
    return keyGen.generateKeyPair();
}
```

Рис.2.7.2.4 – Генерація ключової пари

- Підпис повідомлення: У методі sign() створюється об'єкт Signature з алгоритмом SHA256withRSA. Приватний ключ ініціалізується для підписування, а потім повідомлення конвертується у байтовий масив і оновлюється в об'єкті підпису. Після цього викликається метод sign(), який повертає підписаний байтовий масив, показано на Рис.2.7.2.5.

```
public byte[] sign(String message, PrivateKey privateKey) throws Exception {
    // Створюємо об'єкт для підпису
    Signature signature;
    signature = Signature.getInstance("SHA256withRSA");
    signature.initSign(privateKey);
    signature.update(message.getBytes());

    // Отримуємо підпис
    return signature.sign();
}
```

Рис.2.7.2.5 – Підпис повідомлення

- Перевірка підпису: У методі verify() створюється об'єкт Signature з алгоритмом SHA256withRSA. Публічний ключ ініціалізується для перевірки підпису, а повідомлення оновлюється в об'єкті перевірки підпису. Потім викликається метод verify(), який повертає значення true, якщо підпис перевірено успішно, або false, якщо перевірка не пройшла, показано на Рис.2.7.2.6.

```
public boolean verify(String message, byte[] signature, PublicKey publicKey) throws Exception {
    // Створюємо об'єкт для перевірки підпису
    Signature verifier = Signature.getInstance("SHA256withRSA");
    verifier.initVerify(publicKey);
    verifier.update(message.getBytes());

    // Перевіряємо підпис
    return verifier.verify(signature);
}
```

Рис.2.7.2.6 – Перевірка підпису

2.8 Висновок до 2 розділу

У сучасному світі програмування, наявність правильних засобів та технологій визначає успіх проектів та якість програмного забезпечення. Огляд сучасних засобів та технологій програмування наголошує на їхній важливості для розробників. Мови програмування, інтегровані розробних середовища, фреймворки, бази даних, методології розробки, тестування програмного забезпечення, алгоритми та структури даних, технології веб-розробки, засоби контролю версій та безпеки – всі ці компоненти грають важливу роль у розробці програмного забезпечення.

Сучасні засоби та технології програмування надають розробникам широкі можливості для творчого підходу до створення програм. Вони дозволяють швидко реалізовувати ідеї, забезпечують ефективну роботу з кодом та ресурсами, забезпечують високу продуктивність та якість програмного забезпечення.

Огляд сучасних засобів та технологій програмування також підкреслює необхідність постійного оновлення та вдосконалення знань розробників. Технології швидко розвиваються, і нові інструменти з'являються на ринку. Тому, для досягнення успіху в програмуванні, важливо бути в курсі останніх тенденцій та активно вивчати нові можливості, які пропонуються в цій галузі.

Загалом, огляд сучасних засобів та технологій програмування підкреслює важливість правильного вибору і використання засобів для досягнення успіху у розробці програмного забезпечення. Розробники повинні знати переваги та особливості різних засобів та технологій, а також уміти їх

грамотно поєднувати та використовувати для досягнення поставлених цілей. Ті, хто вміло використовує сучасні засоби та технології програмування, мають перевагу на ринку та можуть створити інноваційні та конкурентоспроможні програмні продукти.

РОЗДІЛ 3. ПОРІВНЯННЯ ЗАСОБІВ ТА ТЕХНОЛОГІЙ ПРОГРАМУВАННЯ

3.1 Критерії вибору засобів та технологій

Засоби та технології, які використовуються при розробці програмного забезпечення, мають велике значення для успішності проекту. Вибір правильних інструментів може визначити ефективність розробки, функціональність продукту та його майбутню масштабованість. Розглянемо критерії вибору засобів та технологій, які можуть допомогти прийняти обґрунтовані рішення.

Першим критерієм вибору є функціональність. Функціональність є одним із ключових критеріїв вибору засобів та технологій при розробці програмного забезпечення. Вона визначає, наскільки добре вибрані інструменти відповідають вимогам і потребам проекту.

При оцінці функціональності розглядаються такі аспекти, як наявність потрібних функцій і можливостей, зручність використання, гнучкість та розширюваність. Засоби та технології повинні відповідати функціональним вимогам проекту, забезпечувати необхідні можливості для його реалізації та задовольняти потреби користувачів.

При оцінці функціональності необхідно аналізувати можливості інструментів у контексті конкретної задачі. Це можуть бути можливості для роботи з базами даних, обробки даних, взаємодії з користувачем, інтеграції з іншими системами тощо. Також важливо враховувати масштабованість та продуктивність засобів, особливо для проектів, що працюють з великими обсягами даних або мають високі вимоги до продуктивності.

Критерій функціональності визначає, наскільки добре обрані засоби та технології відповідають потребам проекту і спроможні забезпечити його успішне виконання. Врахування цього критерію під час вибору допомагає забезпечити відповідність функціональним вимогам проекту, покращити продуктивність та забезпечити зручність використання програмного забезпечення.

Другим критерієм є масштабованість. Масштабованість є важливим критерієм вибору засобів та технологій при розробці програмного забезпечення. Вона визначає, наскільки добре система може збільшувати свою продуктивність та витримувати навантаження при зростанні обсягу даних або користувацького трафіку.

Масштабованість може бути вертикальною або горизонтальною. Вертикальна масштабованість означає збільшення продуктивності системи за рахунок збільшення обсягу ресурсів на окремому сервері, наприклад, збільшення пам'яті або потужності процесора. Горизонтальна масштабованість, зі свого боку, передбачає збільшення продуктивності системи шляхом додавання нових серверів та розподілу навантаження між ними.

Масштабованість є особливо важливою для систем з великим обсягом даних або високою кількістю користувачів, таких як соціальні мережі, електронна комерція або хмарні платформи. Здатність системи масштабуватися дозволяє забезпечити стабільну продуктивність, швидку обробку запитів та ефективне використання ресурсів.

При виборі засобів та технологій для розробки програмного забезпечення необхідно враховувати потенційні потреби у масштабованості. Розглядайте рішення, які забезпечують легку горизонтальну масштабованість, такі як використання контейнерів, мікросервісної архітектури або розподілених систем управління базами даних. Також слід оцінювати можливості автоматичного масштабування та використання хмарних платформ, які надають гнучкість у відповідності до змінних потреб системи.

Масштабованість важлива для забезпечення ефективності та стабільності системи у майбутньому. Враховуючи цей критерій вибору, розробники можуть забезпечити гнучкість і можливість росту свого програмного забезпечення, що сприяє досягненню успіху у розробці та використанні високоякісних програмних рішень.

Третій критерій - спільнота та підтримка. Критерій вибору спільноти та підтримки відіграє важливу роль при виборі засобів та технологій для програмування. Спільнота - це група розробників, яка працює над певною технологією або мовою програмування, обмінюючись знаннями, досвідом та вирішуючи спільні проблеми.

Наявність активної та підтримуваної спільноти має декілька переваг. По-перше, це надає доступ до багатої бази знань, документації та ресурсів, які можуть значно полегшити розробку програмного забезпечення. Розробники можуть швидко знайти відповіді на свої питання, вирішити проблеми та отримати рекомендації щодо ефективного використання засобів програмування.

По-друге, наявність активної спільноти означає, що технологія або мова програмування мають популярність та підтримку. Це вказує на те, що вони широко використовуються та мають довгострокову перспективу. Підтримка забезпечує оновлення, виправлення помилок та нові функціональності, що робить технологію більш надійною та сучасною.

Для оцінки спільноти та підтримки можна враховувати такі фактори, як активність форумів, списків розсилки, стековерфлоу, наявність офіційної документації, оновлення та патчі для засобів програмування. Також важливо враховувати репутацію та надійність компаній або організацій, які стоять за певними технологіями.

Критерій спільноти та підтримки є важливим при виборі засобів та технологій програмування, оскільки він забезпечує доступ до знань, підтримку та перспективу розвитку. Це допомагає розробникам створювати якісне програмне забезпечення та досягати успіху у своїх проектах.

Четвертий критерій - вартість. Критерій вибору вартості відіграє важливу роль у прийнятті рішення про використання певних засобів та технологій програмування. Вартість може охоплювати витрати на ліцензії, платформи, інструменти, ресурси та інші аспекти, пов'язані з розробкою програмного забезпечення.

Однак, при оцінці вартості важливо враховувати не лише витрати на придбання, але й загальну економічну ефективність і вигоди, які можуть бути отримані від використання конкретної технології. Низька вартість може виглядати привабливо, але якщо це веде до обмежень у функціональності, продуктивності чи масштабованості, це може вплинути на загальний успіх проекту.

Для оцінки вартості можна порівняти витрати на ліцензії або платформи, наявність безкоштовних альтернатив, витрати на навчання персоналу або придбання нового обладнання. Також варто враховувати можливості для масштабування, адаптації до майбутніх потреб та гнучкість технології.

Вибір засобів та технологій програмування з урахуванням вартості допомагає забезпечити оптимальний баланс між витратами та якістю розробки програмного забезпечення. Важливо звернути увагу на загальну ефективність і економічну вигоду, яку може принести вибрана технологія у майбутньому.

П'ятий критерій - сумісність Критерій вибору сумісності відіграє важливу роль при виборі засобів та технологій програмування. Сумісність означає здатність різних компонентів програмного забезпечення або технологій працювати разом без проблем.

При оцінці сумісності необхідно враховувати наступні аспекти:

- Платформна сумісність: Вибраний засіб або технологія повинна бути сумісною з платформою, на якій планується розгортання програмного забезпечення. Наприклад, якщо розробка ведеться для платформи Windows, то важливо використовувати мови програмування та інструменти, що підтримують цю платформу.
- Інтеграція: Вибрані засоби та технології повинні бути здатні інтегруватися з існуючими системами або компонентами програмного забезпечення. Це може бути важливо при розробці систем, які повинні співпрацювати з іншими додатками чи сервісами.

- Взаємодія: Якщо в проекті задіяні різні розробники або команди, важливо, щоб вони могли працювати разом із використанням обраних засобів та технологій. Це може включати спільне використання коду, інтеграцію розробницьких середовищ, спільну роботу з репозиторіями коду тощо.
- Підтримка: Важливо мати на увазі наявність активної спільноти, документації, форумів та інших ресурсів, які можуть надати підтримку та допомогу при виникненні проблем або питань під час розробки. Наявність регулярних оновлень та патчів для виправлення помилок також є важливим фактором.

Вибір засобів та технологій з урахуванням критерію сумісності допомагає забезпечити гладку розробку, інтеграцію та функціонування програмного забезпечення. Це дозволяє уникнути проблем, пов'язаних з неповнотою або невідповідністю між різними компонентами системи і забезпечити ефективну співпрацю між розробниками.

І останній критирій вибору засобів та технологій програмування це безпека. Критерій вибору безпеки є одним з найважливіших при виборі засобів та технологій програмування. Забезпечення безпеки програмного забезпечення є критичним, особливо у сучасному цифровому світі, де загрози кібербезпеки стають все більшими.

При виборі засобів та технологій з урахуванням критерію безпеки необхідно враховувати наступні аспекти:

- Захист від вразливостей: Засоби та технології повинні мати вбудовані механізми захисту від вразливостей, що допомагають запобігати атакам, внедренню шкідливого коду або несанкціонованому доступу до системи. Це можуть бути механізми шифрування, автентифікації, контролю доступу та інші заходи безпеки.
- Аудит та моніторинг: Засоби та технології повинні підтримувати можливості аудиту та моніторингу, що дозволяють виявляти та реагувати на потенційні загрози безпеки. Це можуть бути системи виявлення вторгнень (IDS), системи журналювання (logging) та інші інструменти, що допомагають

відстежувати події та аналізувати їх для виявлення вразливостей чи аномальних поведінок.

- Підтримка оновлень та патчів: Засоби та технології повинні мати активну підтримку з боку розробників, що включає регулярні оновлення та виправлення виявлених проблем безпеки. Важливо, щоб існувала можливість швидко впроваджувати оновлення та патчі для захисту системи від нових загроз.
- Безпека даних: Засоби та технології повинні забезпечувати захист конфіденційності, цілісності та доступності даних. Це може включати механізми шифрування даних, контроль доступу до інформації, резервне копіювання та відновлення даних.
- Відкритий код та спільнота: Засоби та технології з відкритим кодом мають перевагу, оскільки їх можна аудитувати та перевіряти на безпеку спільнотою розробників. Наявність активної та забезпеченої спільноти розробників також гарантує швидше виявлення та виправлення проблем безпеки.

Розглядаючи ці критерії, розробники можуть зробити осмислений вибір засобів та технологій, що допоможе забезпечити високий рівень безпеки їх програмного забезпечення. Безпека повинна бути пріоритетом у будь-якому проекті програмування для захисту від потенційних загроз та збереження конфіденційності та цілісності даних.

Загально кажучи, вибір засобів та технологій є важливим кроком при розробці програмного забезпечення. Критерії вибору, такі як функціональність, масштабованість, спільнота та підтримка, вартість та технічні вимоги, допоможуть прийняти обґрунтоване рішення. Ретельний аналіз цих критеріїв та вивчення доступних варіантів дозволить обрати оптимальні засоби та технології для вашого проекту.

Критерії вибору засобів та технологій програмування є ключовими факторами при розробці програмного забезпечення. Сучасні засоби та технології програмування розширюють можливості розробників і дозволяють

створювати складні та інноваційні програмні рішення. Правильний вибір засобів та технологій забезпечує успішну реалізацію проектів та створення якісного програмного забезпечення.

3.2 Порівняння технологій та засобів програмування

Для реалізації будь-якого продукту незалежно від того, за якою сферою застосування працює компанія, ключовим інструментом залишається використання різноманітних мов програмування.

У сучасному світі програмування існує велика кількість різних мов, кожна з яких має свої особливості та призначення. Вибір правильної мови програмування є важливим аспектом для розробки програмного забезпечення, оскільки різні мови мають різні синтаксиси, функціональні можливості та екосистеми підтримки. Проведемо порівняння сучасних мов програмування за їхніми характеристиками та особливостями:

1. Python.

Серед основних її переваг можна назвати такі:

- чистий синтаксис (для виділення блоків слід використовувати відступи);
- переносність програм (що властиве більшості інтерпретованих мов);
- стандартний дистрибутив має велику кількість корисних модулів (включно з модулем для розробки графічного інтерфейсу);
- можливість використання Python в діалоговому режимі (дуже корисне для експериментування та розв'язання простих задач);
- стандартний дистрибутив має просте, але разом із тим досить потужне середовище розробки, яке зветься IDLE і яке написане мовою Python;
- зручний для розв'язання математичних проблем (має засоби роботи з комплексними числами, може оперувати з цілими

числами довільної величини, у діалоговому режимі може використовуватися як потужний калькулятор);

- відкритий код (можливість редагувати його іншими користувачами).

Python має ефективні структури даних високого рівня та простий, але ефективний підхід до об'єктно-орієнтованого програмування. Елегантний синтаксис Python, динамічна обробка типів, а також те, що це інтерпретована мова, роблять її ідеальною для написання скриптів та швидкої розробки прикладних програм у багатьох галузях на більшості платформ.

Недоліки:

- Низька швидкодія
- Відсутність статичної типізації
- Неможливість модифікації вбудованих класів
- Глобальне блокування інтерпретатора

2. JavaScript.

JavaScript має декілька переваг, серед яких:

- Високий рівень абстракції: JavaScript дозволяє розробникам працювати на високому рівні абстракції, що спрощує роботу з кодом і зменшує його складність.
- Крос-браузерність: JavaScript підтримується майже всіма сучасними веб-браузерами, що дозволяє створювати веб-додатки, які працюють на різних платформах і браузерах.
- Автоматичне приведення типів і збирання сміття: JavaScript автоматично вирішує проблему типізації, що спрощує роботу з даними. Він також володіє механізмом збору сміття, що автоматично вивільняє пам'ять від невикористовуваних об'єктів, спрощуючи управління пам'яттю.

- Обробка винятків: JavaScript надає механізми для обробки винятків, що дозволяє розробникам ефективно управляти помилками і виконувати відповідні дії при виникненні помилок.
- Спрощений синтаксис: Синтаксис JavaScript є простим і зрозумілим, що полегшує розробку програм і покращує читабельність коду.

Проте, JavaScript також має свої недоліки:

- Невисока швидкість роботи: Оскільки JavaScript є інтерпретованою мовою, він може працювати повільніше порівняно з мовами, що компілюються. Це може бути проблемою для вимогливих до продуктивності програм.
- Складність відлагодження: JavaScript може мати складнощі з відлагодженням, особливо в складних програмах, через свою динамічну природу і можливість зміни типів даних під час виконання.
- Виконання програм у браузерах: JavaScript часто виконується у середовищі браузера, що обмежує доступ до деяких операцій і може вплинути на безпеку і продуктивність програм.

3. Java.

Java має декілька переваг, серед яких:

- Незалежність від архітектури і переносимість: Код, написаний на Java, може виконуватися на різних платформах без необхідності внесення змін. Це робить мову Java дуже переносимою і підходить для розробки крос-платформових додатків.
- Безпека від низькорівневих помилок: Java має вбудовану систему безпеки, яка дозволяє уникати низькорівневих помилок, таких як витоки пам'яті або небезпечні виклики методів.
- Висока продуктивність виконання: Java використовує віртуальну машину Java (JVM), яка забезпечує оптимізацію коду та

ефективне виконання програм. Це дозволяє отримати високу продуктивність виконання.

- Автоматичне керування пам'ятю: Java використовує систему збору сміття, яка автоматично вивільняє пам'ять, звільняючи програміста від необхідності вручну керувати пам'яттю.

Проте, Java також має свої недоліки:

- Низька швидкість виконання програм: Хоча JVM надає оптимізацію, виконання Java-програм може бути повільнішим порівняно з мовами, що компілюються безпосередньо в машинний код.
- Значне використання ресурсів пам'яті: Java використовує додаткові ресурси пам'яті для виконання JVM і управління об'єктами, що може призвести до більшої використання пам'яті в порівнянні з деякими іншими мовами.

4. C++.

Мова програмування C++ є статично типізованою і компільованою мовою загального призначення. Вона підтримує різні парадигми програмування, такі як процедурне, об'єктно-орієнтоване та узагальнене програмування. C++ використовується для розробки системного програмного забезпечення, драйверів, серверних та клієнтських програм, а також відеоігор.

Синтаксис мови C++ подібний до мови C, проте вона є розширеною версією мови C, вводячи об'єктно-орієнтовані можливості за допомогою класів, що забезпечують інкапсуляцію, успадкування та поліморфізм. Стандартна бібліотека C++ базується на Стандартній Бібліотеці Шаблонів (STL) і надає корисні інструменти, такі як контейнери та ітератори.

Переваги мови C++ включають:

- Швидкість роботи програм: Мова C++ має майже таку ж високу швидкість роботи, як мова C, що робить її популярним вибором для завдань, де потрібна продуктивність.

- Переносимість: C++ підходить для розробки на різних системах і платформах, що робить його універсальним вибором для багатьох проектів.
- Низькорівневі можливості: Мова C++ дозволяє працювати з адресами, портами і пам'яттю на низькому рівні, що робить її відмінним вибором для системного програмування та розробки вбудованих систем.
- Підтримка різноманітних стилів програмування: C++ дозволяє програмувати в різних стилях, включаючи процедурне, об'єктно-орієнтоване та узагальнене програмування, що дає розробникам більше гнучкості.

Недоліки мови C++ включають:

- Можливості, що знижують безпеку програм: Деякі можливості мови C++ можуть сприяти виникненню помилок, вразливостей і непередбачуваної роботи програми, аналогічно мові C.
- Обмежена підтримка модульності: C++ має обмежену підтримку модульності, що може ускладнювати організацію та управління великими проектами.
- Неінтуїтивне перетворення типів даних: У деяких випадках перетворення типів даних в C++ може бути складним і потребувати додаткової уваги програміста.
- Відсутність підтримки функціонального програмування: Мова C++ не має вбудованої підтримки функціонального програмування, що може бути недоліком для розробників, які більше цінують цей підхід до програмування.

5. C#.

C# є об'єктно-орієнтованою мовою програмування для платформи Microsoft .NET Framework, вона має безпечну систему типізації та синтаксис, подібний до мови C. Мова підтримує строгу статичну типізацію, перевантаження операторів, поліморфізм, делегати, атрибути, події, анонімні

функції та ітератори. Вона поєднує в собі кращі риси мов C і C++, а також використовує можливості .NET Framework, такі як FCL (Framework Class Library) і CLR (Common Language Runtime).

Переваги мови C# включають:

- Об'єктно-орієнтовані можливості: C# є потужною об'єктною мовою, яка підтримує універсалізацію й спадкування, дозволяючи розробляти структуровані та модульні програми.
- Інтеграція з .NET Framework: C# повністю інтегрована з .NET Framework і використовує його функціональні можливості, такі як FCL та CLR, що забезпечує багатий набір інструментів для розробки програм.
- Зручний синтаксис: Синтаксис C# подібний до мови C, що дозволяє програмістам з легкістю переходити від однієї мови до іншої. Це робить мову C# привабливою для програмістів, які вже мають досвід роботи з мовами C або C++.
- Підтримка .NET Framework: Завдяки використанню .NET Framework, програмісти мови C# отримують переваги віртуальної машини, такі як можливість перенесення коду на різні платформи і підтримка веб-служб.
- Багатий набір бібліотек: C# має потужну бібліотеку каркасів, яка допомагає легко будувати різні типи додатків, включаючи веб-служби та доступ до баз даних.

Незважаючи на свої переваги, мова C# також має деякі обмеження і недоліки, які включають обмежену підтримку деяких моделей програмування і відсутність підтримки функціонального програмування, яке може бути важливим для деяких розробників.

6. Ruby:

- Відома своєю простотою та зручним синтаксисом, Ruby є популярною мовою для розробки веб-додатків та веб-сайтів.

- Має фреймворк Ruby on Rails, який спрощує створення веб-додатків та роботу з базами даних.

7. Go:
   - Розроблена компанією Google, Go є ефективною мовою програмування, що спеціалізується на створенні надійних та швидких програм.
   - Підтримує паралельне виконання, вбудовану підтримку мережевих операцій та має компактний бінарний вихідний код.
   - Go є мовою з дуже простим синтаксисом, зрозумілими конструкціями та мінімальною кількістю ключових слів. Вона була спеціально розроблена для спрощення розробки і підвищення продуктивності.
8. Swift:
   - Розроблена компанією Apple, Swift є основною мовою для розробки додатків під iOS, macOS, watchOS та tvOS.
   - Має сучасний синтаксис, підтримку безпеки типів та велику кількість бібліотек для розробки мобільних додатків.
9. PHP:
   - PHP є популярною мовою для розробки веб-додатків та динамічних веб-сайтів.
   - Є багато фреймворків, таких як Laravel та Symfony, які полегшують розробку веб-додатків та спрощують взаємодію з базами даних.
10. Rust:
    - Rust є мовою програмування з високою безпекою та ефективністю.
    - Забезпечує контроль пам'яті на етапі компіляції, що допомагає уникнути багатьох типових помилок.
    - Часто використовується для системного програмування, розробки бібліотек та інших проектів, де важлива швидкість та безпека.

- Має трохи складніший синтаксис, але надає багато інструментів для уникнення помилок та забезпечення безпеки.

Один з головних аспектів порівняння мов програмування - це синтаксис. Різні мови мають різні правила і структури для написання програм. Наприклад, мови з синтаксисом C-подібним, такі як C++, Java або C#, використовують фігурні дужки та точки з комою для розділення коду. У той же час, мови з функціональним синтаксисом, такі як Python або JavaScript, використовують відступи для показу блоків коду. Вибір мови залежить від особистих уподобань розробника та вимог проекту.

Інший важливий аспект - це функціональні можливості мови. Різні мови мають різні набори функцій та можливостей для створення програмного забезпечення. Наприклад, мови з орієнтованими на об'єкти парадигмами, такі як Java або C#, надають можливість створювати класи та об'єкти, що сприяє модульності та повторному використанню коду. У той же час, мови з функціональним підходом, такі як Haskell або Scala, надають багато функцій для роботи з функціями вищого порядку, рекурсії та інших функціональних концепцій. Вибір мови залежить від вимог проекту та стилю програмування, який розробник використовує.

Також варто враховувати екосистему підтримки мови. Деякі мови, такі як Java, C++ або Python, мають великі та активні спільноти розробників, що призводить до наявності багато ресурсів, документації та бібліотек для вирішення різних задач. Інші мови можуть мати меншу спільноту, але при цьому вони можуть бути спрямовані на конкретні галузі або мати високий рівень продуктивності. Вибір мови залежить від потреб проекту та доступних ресурсів.

Зміна популярності мов програмування є невід'ємною частиною швидкого технологічного розвитку. Розробники постійно стикаються з новими викликами та вимогами, що вимагає оновлення інструментарію, якими вони користуються. Отже, мови програмування, які набувають

популярності, можуть варіюватись у залежності від таких факторів, як ефективність, зручність синтаксису, розширені можливості та розмір спільноти розробників.

Один із прикладів зміни популярності мов програмування є зростання популярності Python. Python, з його простим і зрозумілим синтаксисом, став привабливим вибором для багатьох розробників. Його велика спільнота розробників забезпечує широку підтримку та наявність безлічі бібліотек і фреймворків для різних галузей, таких як наукові дослідження, аналіз даних та веб-розробка. Це призвело до зростання використання Python у багатьох сферах.

Інший приклад - TypeScript. Завдяки своїм можливостям статичної типізації, TypeScript отримує все більше популярності серед розробників. Він забезпечує покращену безпеку коду та поліпшує розробку великих проектів. TypeScript є суперсетом JavaScript, що означає, що він побудований на основі JavaScript, що робить його сумісним з існуючим JavaScript кодом.

У той час як деякі мови програмування можуть зазнавати спаду популярності, такі як Java або C++, інші мови, такі як Go або Kotlin, можуть зростати в популярності через свої особливості та специфічні використання.

На Рис.3.2.1, Рис.3.2.2, Рис.3.32.3 ми можемо спостерігати умовну популярність деяких мов програмування в залежності від сфери використання цих мов.

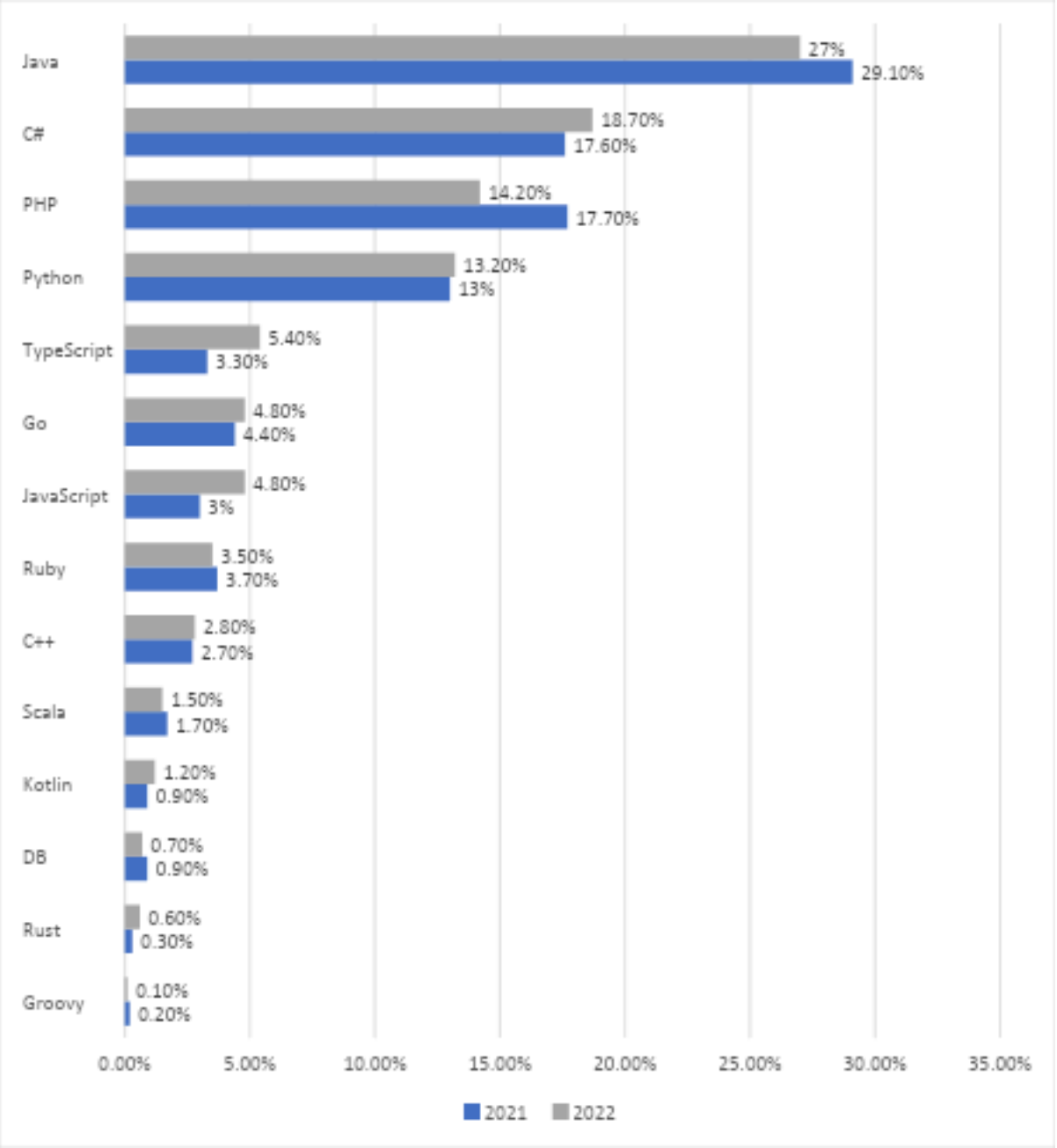

Рис.3.2.1 – Популярність мов програмування в Back-end

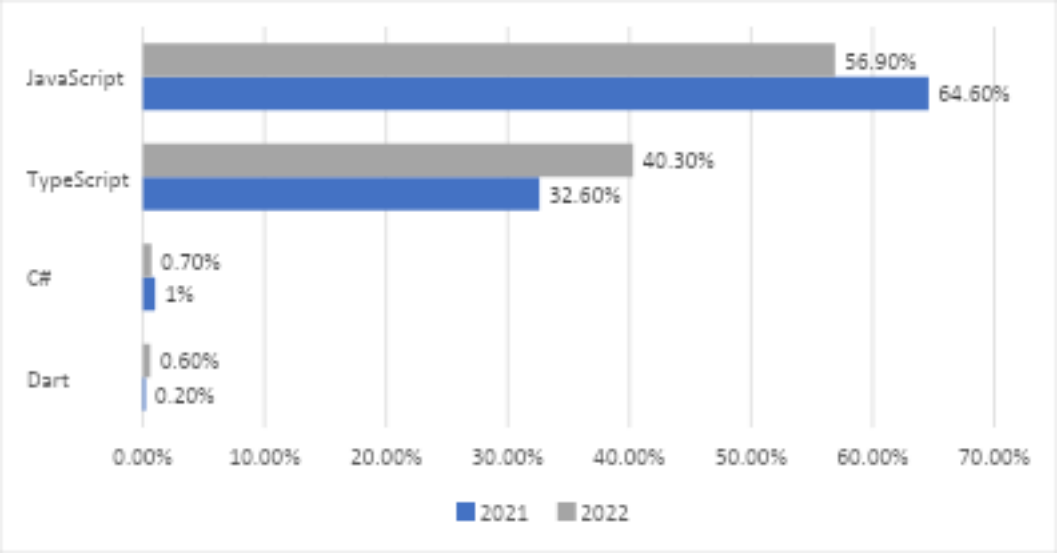

Рис.3.2.2 – Популярність мов програмування в Front-end

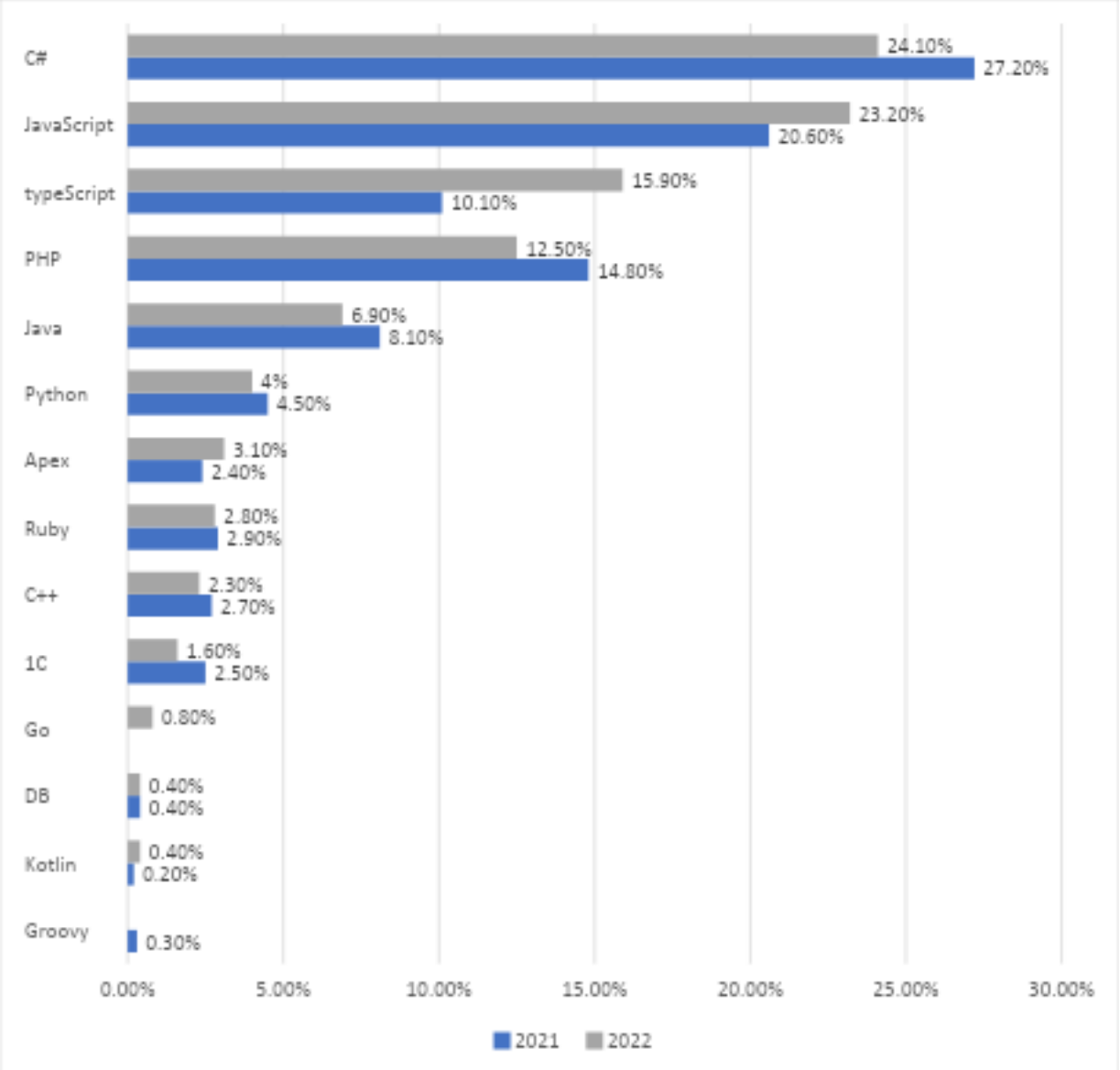

Рис.3.2.3 – Популярність мов програмування в Full Stack

Так як для аналізу береться стандартний вибір мови програмування для роботи або комерційного\власного використання, то було прийнято рішення також провести аналіз серед мов програмування для новачків. Критерії вибору мови програмування вже було розглянуто у попередньому розділі. Вони є універсальними та повністю підходять і для використання новачками у даній сфері.

Новачки в програмуванні - це люди, які тільки починають вивчати і володіти навичками програмування. Вони можуть бути студентами, людьми, які змінюють свою професію на програміста, або просто тими, хто цікавиться цією галуззю і бажає вивчити програмування як хобі.

Новачкам часто властива обмежена або нульова попередня експертиза в програмуванні, і вони потребують введення в основи програмування, вивчення синтаксису мови програмування, а також засвоєння концепцій і принципів програмування.

Новачки можуть вивчати програмування самостійно за допомогою підручників, онлайн-курсів, відеоуроків або через формальне навчання в університетах або навчальних закладах. Вони також можуть приєднатися до спільнот програмістів, де вони можуть отримати підтримку, поради і навіть знайти можливості для співпраці над проектами.

На Рис3.2.4, Рис3.2.5, Рис3.2.6, Рис3.2.7, Рис3.2.8 ми можемо спостерігати динаміку популярності п’яти обраних для дослідження мов програмування.

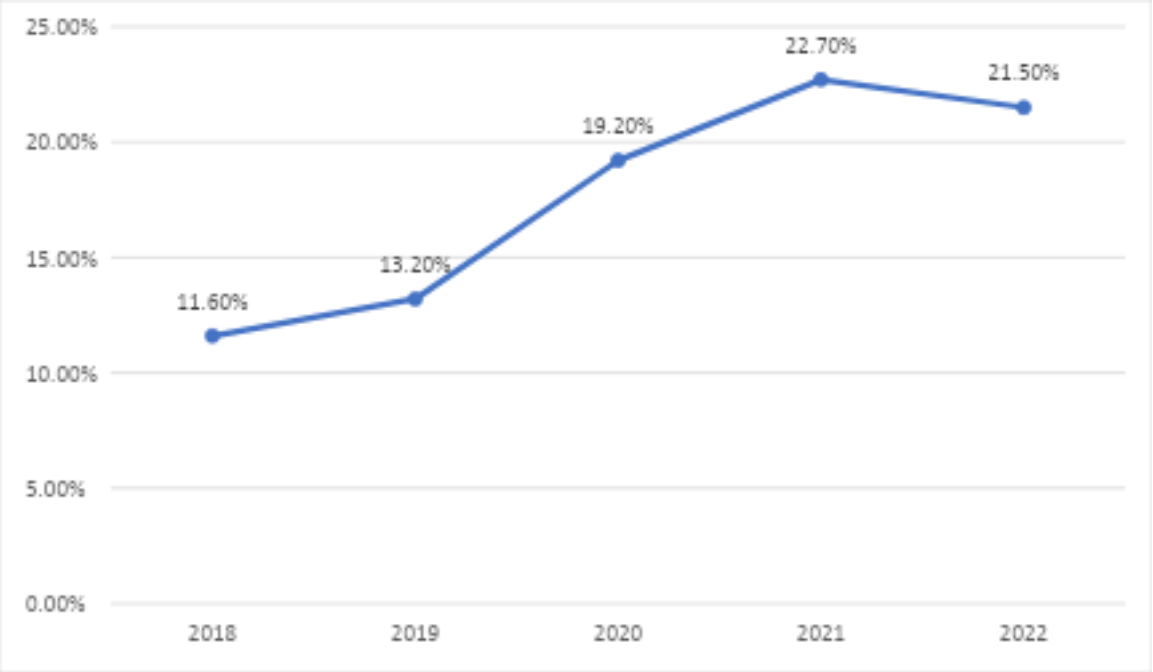

Рис.3.2.4 – Популярність JavaScript

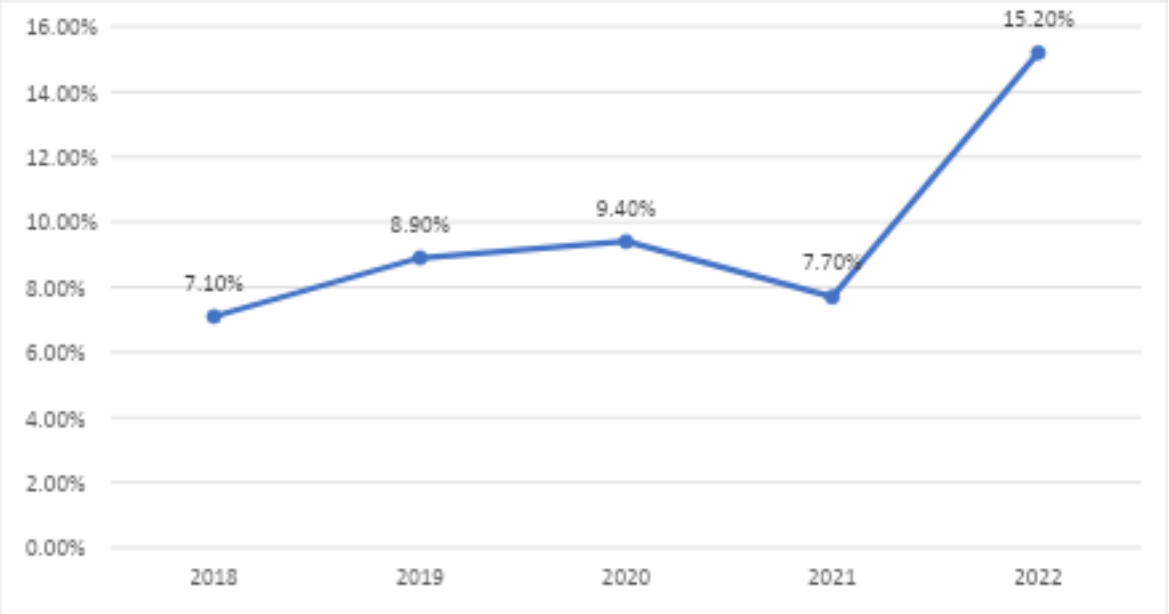

Рис.3.2.5 – Популярність Python

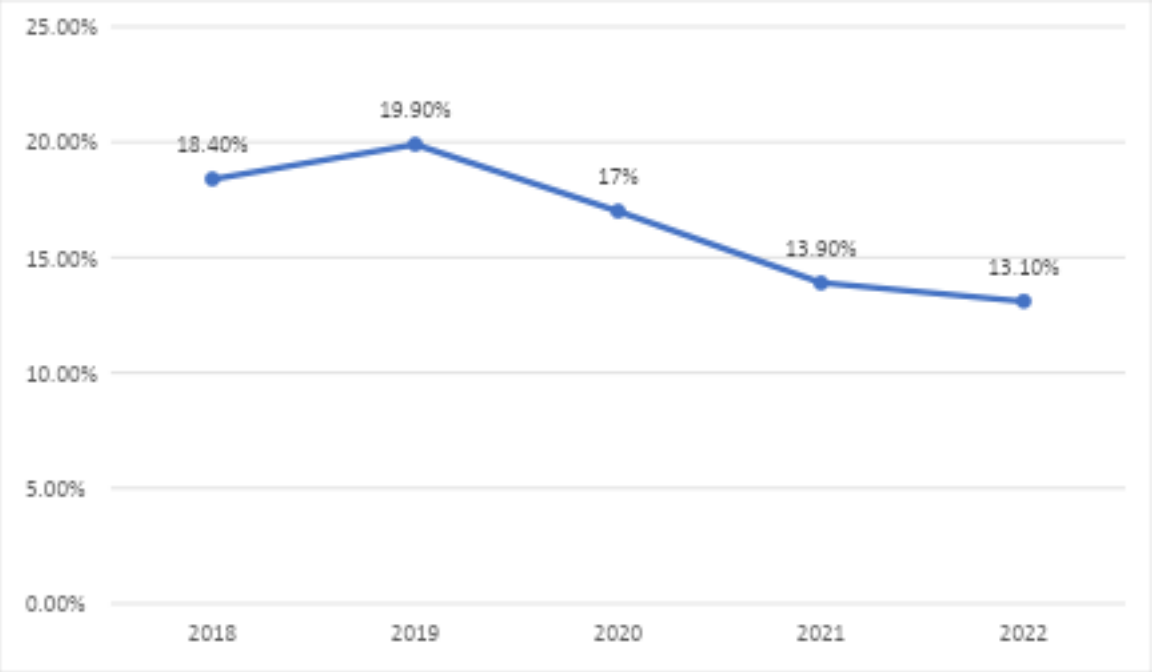

Рис.3.2.6 – Популярність C++

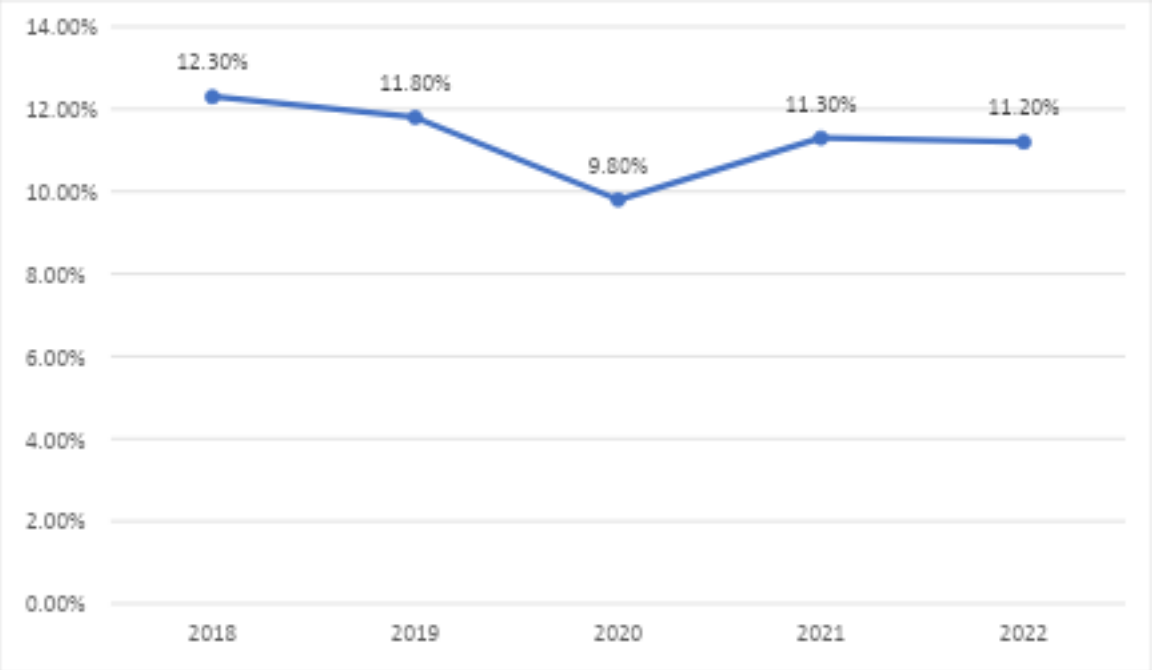

Рис.3.2.7 – Популярність Java

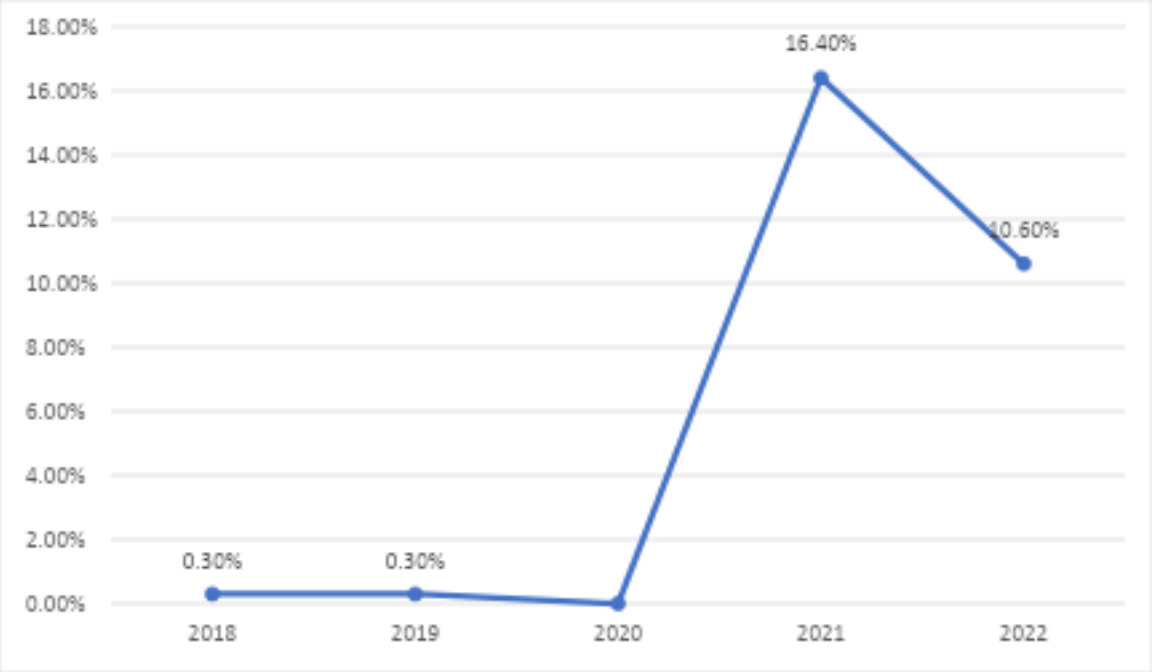

Рис.3.2.8 – Популярність Pascal

Нині у більшості новачків першою мовою програмування є JavaScript, але також помітно зростає роль Python. З одного боку, можливо, це відображення того, що Python більше підходить для навчання, ніж JavaScript, з іншого боку — дані показують, що більшість з тих, для кого JavaScript була першою мовою програмування, нині працює в екосистемі JavaScript/TypeScript, а для кого Python — на Python, тож, можливо, це просто розподіл траєкторій навчання.

Загальна аналітична таблиця містить інформацію про мови програмування, яка включає в себе наступні показники: кількість програмістів, які використовують цю мову у комерційних проектах (виражену як відсоток від загальної кількості програмістів та абсолютне числове значення), зміни у популярності цієї мови порівняно з попереднім роком, кількість програмістів, які використовують цю мову як додаткову, та індекс вподобання цієї мови.

Ця таблиця надає зручну уяву про ситуацію на ринку мов програмування, дозволяючи порівнювати їх популярність та використання. Вона допомагає виявити тренди та зміни у виборі мов програмування серед

програмістів. Це корисний інструмент для розробників, рекрутерів та компаній, які планують проекти та формують команди програмістів.

Аналізуючи цю таблицю, можна отримати важливі усвідомлення щодо популярних мов програмування, їхнього росту або спаду використання, а також розуміння того, які мови залишаються актуальними і мають перспективи для майбутнього. Такий огляд допомагає приймати обґрунтовані рішення щодо вибору мови програмування для конкретних проектів і планування кар'єри в галузі програмування.

Таблиця 3.2.1 – Статистика використання сучасних мов програмування

Мова	Пишуть зараз,%	Зміни	Пишуть зараз, к-ть	Як додаткова мова, к-ть	Індекс вподобання
JavaScript	19,1	+0,56	1695	3186	51,6
Java	14	0,00	1241	851	72,4
Python	13,4	+2,34	1190	1178	73,8
C#	13,3	-1,21	1179	533	88,1
TypeScript	13,3	+3,06	1179	2110	80,5
PHP	7,1	-3,34	627	423	59,0
Kotlin	3,1	-0,11	286	272	88,4
C++	2,8	-0,41	251	321	67,3
DB	2,7	+1,27	237	3669	23,9
Swift	2,5	-0,81	221	151	93,8
Go	1,9	-0,31	165	337	83,6

Ruby	1,8	-0,32	158	117	65,3
Dart	1,1	0,00	96	78	83,7

Згідно з проаналізованими даними можна спостерігати стрімке зростання популярності мови програмування TypeScript. TypeScript починає займати нові ніші у програмуванні, здобуваючи популярність як на бекенді, так і в повному стеку розробки. Це свідчить про його широкий спектр застосування та зростаючу придатність для різних типів проєктів.

Крім того, також варто відзначити різке зростання популярності мови програмування Python. Після певного застою у розвитку Python знову став активно використовуваною мовою завдяки своїм універсальним можливостям та широкій спільноті розробників. Python знаходить застосування у багатьох галузях, включаючи наукові дослідження, машинне навчання, веб-розробку та автоматизацію задач. Його простий і зрозумілий синтаксис, разом з великою кількістю доступних бібліотек та фреймворків, роблять Python привабливим вибором для багатьох розробників.

Зміна популярності мов програмування є невід'ємною частиною швидкого розвитку технологій і відображає сучасні потреби та тенденції у програмуванні. Розробники постійно вибирають мови, які найкраще задовольняють їх потреби та допомагають ефективно створювати програмне забезпечення.

3.3. Висновок до 3 розділу

Було проведено аналіз різних засобів та технологій програмування з різних аспектів та виявлено, що сучасні засоби та технології програмування розширюють можливості розробників і дозволяють створювати складні та інноваційні програмні рішення. Вибір правильних засобів та технологій, врахування критеріїв вибору і забезпечення безпеки є важливими аспектами у програмуванні та створенні якісного програмного забезпечення.

Порівняння засобів та технологій було проведено за такими критеріями: функціональність, масштабованість, спільнота та підтримка, вартість, сумісність і безпека. Ці критерії допомагають розробникам приймати обґрунтовані рішення щодо вибору засобів та технологій програмування відповідно до потреб проекту.

Крім того, було проведено порівняння різних мов програмування за їх характеристиками та перевагами. Було виявлено, що різні мови мають свої особливості і придатні для різних сфер використання. Вибір мови програмування залежить від конкретних потреб проекту, його масштабу, швидкості розробки та інших факторів.

ВИСНОВОК

У бакалаврській кваліфікаційній роботі "Традиційні сучасні засоби та технології програмування" було проведено детальний аналіз різних засобів та технологій програмування, які застосовуються в сучасному програмуванні. Основна мета роботи полягала в порівнянні цих засобів і технологій за різними характеристиками.

В результаті дослідження було встановлено, що на сьогоднішній день існує широкий вибір засобів та технологій програмування, які дозволяють розробникам створювати якісне та інноваційне програмне забезпечення. Кожен засіб має свої особливості, відмінності в синтаксисі та підходах до програмування, що дозволяє вибрати найбільш підходящий для конкретного проекту.

Згідно до поставлених задач було виконано ряд досліджень:

- Наявно показані основні традиційні сучасні засоби та технології програмування.
- Досліджено основні принципи та особливості кожної з аналізованих мов програмування, а також сфери їх використання.
- Проаналізовано розвиток та популярність цих засобів та технологій протягом останніх років.

У роботі також було проаналізовано різні мови програмування і виявлено, що вони мають свої особливості та додаткові можливості. Вибір мови програмування залежить від конкретного проекту, вимог до швидкості розробки, потреб у підтримці та інших факторів.

Дослідження показало, що кожен засіб та технологія програмування має свої переваги та обмеження, що залежать від конкретних вимог проекту, типу програмного продукту та особистих уподобань розробників. Наприклад, деякі мови програмування, такі як Java і C++, надають високий рівень контролю над системою та велику швидкодію, що робить їх популярними для

розробки системного програмного забезпечення. З іншого боку, мови програмування, такі як Python і JavaScript, відомі своєю простотою вивчення та широкими можливостями у розробці веб-додатків.

Засоби та технології програмування також постійно розвиваються, надаючи нові функціональні можливості та поліпшення. Наприклад, з'явилися нові фреймворки та бібліотеки, що спрощують розробку веб-додатків та розширюють їх функціонал. Також, розвиток мобільних платформ призвів до появи спеціалізованих мов програмування та інструментів для розробки мобільних додатків.

Порівняння засобів та технологій програмування було проведено за такими критеріями, як функціональність, масштабованість, спільнота та підтримка, вартість, сумісність та безпека. Ці критерії визначають ефективність та придатність засобів та технологій для реалізації різних проектів.

Висновок з дипломної роботи демонструє, що вибір правильних засобів та технологій програмування є ключовим фактором для успішного виконання проектів. Кожен розробник повинен ретельно вивчити особливості і можливості різних засобів та технологій та обрати найбільш підходящі для своїх потреб. Це дослідження дозволяє розуміти важливість вибору правильних засобів та технологій програмування для досягнення успіху у процесі розробки програмного забезпечення і відповіді на виклики сучасного світу.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. History of programming languages [Електронний ресурс] – Режим доступу до ресурсу: https://devskiller.com/blog/history-of-programming-languages/.
2. The Evolution of Programming Languages [Електронний ресурс] – Режим доступу до ресурсу: https://www.geeksforgeeks.org/the-evolution-of-programming-languages/.
3. A must know for all Programmers [Електронний ресурс] – Режим доступу до ресурсу: https://hackr.io/blog/programming-paradigms.
4. Асемблер [Електронний ресурс] – Режим доступу до ресурсу: https://studfile.net/preview/5200239/page:3/#5.
5. ТЕХНОЛОГІЇ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ [Електронний ресурс] – Режим доступу до ресурсу: http://reposit.nupp.edu.ua/bitstream/PoltNTU/4431/1/Учебник_ТРПЗ_проверено-converted.pdf.
6. Best Programming Languages To Learn [Електронний ресурс] – Режим доступу до ресурсу: https://www.simplilearn.com/best-programming-languages-start-learning-today-article.
7. Що таке технологія блокчейн [Електронний ресурс] – Режим доступу до ресурсу: https://aws.amazon.com/ru/what-is/blockchain/?aws-products-all.sort-by=item.additionalFields.productNameLowercase&aws-products-all.sort-order=asc.
8. blockchain [Електронний ресурс] – Режим доступу до ресурсу: https://www.blockchain.com.
9. ar/vr technologies [Електронний ресурс] – Режим доступу до ресурсу: https://softengi.com/blog/ar-vr-for-ehs/.

10. Google AR & VR [Електронний ресурс] – Режим доступу до ресурсу: https://arvr.google.com.
11. Фреймворки у мовах програмування [Електронний ресурс] – Режим доступу до ресурсу: https://cloud.itstep.org/blog_3/frameworks-in-programming-languages-what-are-they-for-and-how-to-choose-them.
12. Використання сучасних Web-фреймворків [Електронний ресурс] – Режим доступу до ресурсу: https://moodle.znu.edu.ua/pluginfile.php/594196/mod_resource/content/1/Використ.сучасних.фреймворків_1_2020.pdf.
13. Cloud computing [Електронний ресурс] – Режим доступу до ресурсу: https://www.investopedia.com/terms/c/cloud-computing.asp.
14. Cloud Computing In the Finance Iindustry/ [Електронний ресурс] – Режим доступу до ресурсу: https://vexxhost.com/blog/cloud-computing-in-the-finance-industry/
15. Чому ШІ та хмарні технології очолюють списки трендів 2022 року? [Електронний ресурс] – Режим доступу до ресурсу: https://worldvision.com.ua/chomu-shi-ta-khmarni-tekhnologii-ocholuut-spiski-trendiv-2022-roku/.
16. [Електронний ресурс] Mobile Vs. Desktop Usage - https://www.broadbandsearch.net/blog/mobile-desktop-internet-usage-statistics
17. [Електронний ресурс] The benefits of using web-based applications - https://www.geeks.ltd.uk/about-us/blog/details/eQU5Ip/the-benefits-of-usingweb-based-applications
18. [Електронний ресурс] The pros and cons of building a Mobile app vs a Web app - https://designli.co/blog/the-pros-and-cons-of-building-a-mobile-app-vs-aweb-app/

19. [Електронний ресурс] SwiftUI https://developer.apple.com/xcode/swiftui/
20. [Електронний ресурс] iOS MVVM Tutorial https://www.raywenderlich.com/6733535-ios-mvvm-tutorial-refactoring-frommvc
21. [Електронний ресурс] Introducing XCode 12 https://developer.apple.com/xcode/
22. Backend Developer [Електронний ресурс] – Режим доступу до ресурсу: https://roadmap.sh/backend.
23. A Guide to Front-End vs. Back-End vs. Full-Stack Development [Електронний ресурс] – Режим доступу до ресурсу: https://www.indeed.com/career-advice/finding-a-job/back-end-vs-front-end-vs-full-stack-development.
24. Криптографія [Електронний ресурс] — Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Криптографія#Історія_криптографії
25. W.Stallings. Cryptography and Network Security, 2nd Edition, 1999.
26. Python [Електронний ресурс] — Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/Python
27. C++ (язык программирования) [Електронний ресурс] — Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/C%2B%2B
28. Електронний цифровий підпис [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Електронний_цифровий_підпис.
29. Як працює шифрування з приватним та публічним ключем [Електронний ресурс] – Режим доступу до ресурсу: https://tsecrypto.com/article/shho-take-kryptografiya/.
30. Яку мову програмування краще вивчати першою [Електронний ресурс] – Режим доступу до ресурсу: https://issoft.ua/news/yaku-movu-programuvannya-krashhe-vivchati/.

31. Introduction to BouncyCastle [Електронний ресурс] – Режим доступу до ресурсу: https://www.baeldung.com/java-bouncy-castle.
32. Java Base64 Encoding and Decoding [Електронний ресурс] – Режим доступу до ресурсу: https://www.baeldung.com/java-base64-encode-and-decode.
33. What Is Cloud Computing and the Top Cloud Technologies to Look Out [Електронний ресурс] – Режим доступу до ресурсу: https://www.simplilearn.com/cloud-technologies-article.
34. Machine Learning, ML [Електронний ресурс] – Режим доступу до ресурсу: https://www.it.ua/knowledge-base/technology-innovation/machine-learning.
35. Історія розвитку мов програмування [Електронний ресурс] – Режим доступу до ресурсу: https://crashbox.ru/tools-to-help/a-short-history-of-the-development-of-programming-languages-is-a-brief-history-of-programming-languages/.
36. ТОП 10 фреймворков и библиотек для Front-end Dev: [Електронний ресурс] – Режим доступу до ресурсу: https://web-academy.ua/blog/junior/10-front-end-dev-2018.
37. Текстовий редактор [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Текстовий_редактор.
38. Інтерпретатори і компілятори [Електронний ресурс] – Режим доступу до ресурсу: https://disted.edu.vn.ua/courses/learn/584.
39. Про систему контролю версій [Електронний ресурс] – Режим доступу до ресурсу: https://git-scm.com/book/uk/v2/Вступ-Про-систему-контролю-версій.
40. Version Control Systems [Електронний ресурс] – Режим доступу до ресурсу: https://www.geeksforgeeks.org/version-control-systems/.