МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ТЕХНОЛОГІЙ ТА ДИЗАЙНУ
Факультет механтроніки та комп'ютерних наук
Кафедра комп'ютерних наук

Дипломна магістерська робота

на тему: Розробка програмного забезпечення для створення довідкової системи комунікації студентів університету на базі Telegram Bot Api

Виконав: студент групи МгІТ-1-20
спеціальності 122 Комп'ютерні науки
освітньої програми Комп'ютерні науки

Андрій САВЧЕНКО

Керівник к.т.н., доц. Тетяна ДЕМКІВСЬКА

Рецензент д.т.н. проф. Віктор ЧУПРИНКА

Київ 2021

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ТЕХНОЛОГІЙ ТА ДИЗАЙНУ

Факультет механтроніки та комп'ютерних наук
Кафедра комп'ютерних наук
Спеціальність 122 Комп'ютерні науки
Освітня програма Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри комп'ютерних наук

_____ Володимир Щербань

"_____" _____2021 року

З А В Д А Н Н Я

НА ДИПЛОМНУ МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТУ

Савченко Андрію Сергійовичу

1. Тема роботи Розробка програмного забезпечення для створення довідкової системи комунікації студентів університету на базі Telegram Bot Api

Науковий керівник роботи Демківська Тетяна Іванівна, к.т.н., доцент

затверджені наказом вищого навчального закладу від «04» жовтня 2021 року № 286

2. Строк подання студентом роботи 21.12.2021 р.

3. Вихідні дані до роботи Розробка кафедри комп'ютерних наук.

4. Зміст дипломної роботи (перелік питань, які потрібно розробити) РОЗДІЛ 1. Постановка задачі та основні принципи роботи методологій розробки; РОЗДІЛ 2. Алгоритмічне забезпечення життєвого циклу програми; РОЗДІЛ 3. Програмне забезпечення; презентація дипломної магістерської роботи з основними результатами дослідження (в роздрукованому вигляді представлена у додатках).

5. Консультанти розділів дипломної магістерської роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Вступ	к.т.н., доц. Тетяна ДЕМКІВСЬКА		
Розділ 1	к.т.н., доц. Тетяна ДЕМКІВСЬКА		
Розділ 2	к.т.н., доц. Тетяна ДЕМКІВСЬКА		
Розділ 3	к.т.н., доц. Тетяна ДЕМКІВСЬКА		
Висновки	к.т.н., доц. Тетяна ДЕМКІВСЬКА		

6. Дата видачі завдання 10.2021 р.

КАЛЕНДАРНИЙ ПЛАН

№з/п	Назва етапів дипломної магістерської роботи	Терміни виконання етапів	Примітка про виконання
1	Вступ	02.10.2021	
2	Розділ 1 Постановка задачі та основні принципи роботи методологій розробки	18.10.2021	
3	Розділ 2 Алгоритмічне забезпечення життєвого циклу програми	30.10.2021	
4	Розділ 3 Програмне забезпечення	14.11.2021	
5	Висновки	24.11.2021	
6	Оформлення дипломної магістерської роботи (чистовий варіант)	28.11.2021	
7	Здача дипломної магістерської роботи на кафедру для рецензування (за 14 днів до захисту)	14.12.2021	
8	Перевірка дипломної магістерської роботи на наявність ознак плагіату (за 10 днів до захисту)	16.12.2021	
9	Подання дипломної магістерської роботи на затвердження завідувачу кафедри (з 7 днів до захисту)	21.12.2021	

Студент _____ Андрій САВЧЕНКО

Науковий керівник роботи _____ Тетяна ДЕМКІВСЬКА

Директор НМЦУПФ _____ Олена ГРИГОРЕВСЬКА

АНОТАЦІЯ

Савченко А.С. Розробка програмного забезпечення для створення довідкової системи комунікації студентів університету на базі Telegram Bot Api.

Дипломна магістерська робота за спеціальністю 122 - «Комп'ютерні науки» – Київський національний університет технологій та дизайну, Київ, 2021 рік.

Досліджено основні методології та принципи життєвого циклу програмного забезпечення які використовуються для побудови програм, зокрема методологію Scrum, DevOps та метод водоспаду. Розглянуто технології відкритого API та написання додатків на мові Python.

В результаті даної роботи був розроблений додаток, який на основі введених даних надає канали комунікації та доступ до важливої інформації. Додаток створено на платформі месенджера Telegram з використанням об'єкто-орієнтованої мови програмування Python.

Ключові слова: комунікація, месенджер, API, Telegram, Python, Bot, методологія, життєвий цикл, SDLC, Scrum.

ANNOTATION

Savchenko A.S. Development of software for creating a reference communication system for university students based on Telegram Bot Api.

Master's thesis in specialty 122 - "Computer Science" - Kyiv National University of Technology and Design, Kyiv, 2021.

The main methodologies and principles of the software lifecycle are investigated, including the Scrum methodology, DevOps and the waterfall method. The article covers technologies of open API and writing Python add-ons.

As a result of this work, the addendum was developed which provides communication channels and access to important information based on the entered data. The addendum was created on the Telegram messenger platform using the Python programming language.

Keywords: communication, messenger, API, Telegram, Python, Bot, methodology, life cycle, SDLC, Scrum.

ВСТУП

Актуальність теми **-** великі компанії по світу все більше роблять акцент на методах розробки програмного забезпечення – це стає основою розробки. Також набирає обертів розробка ботів дуже різних форматів, одним з видів є чат-боти які на даному етапі розвитку способу комунікації через месенджери є дуже актуальними. Тема розкриває чому тепер дуже важливі саме методи розробки програмного продукту і чому саме приклад чат-боту має показати усі ефективні сторони дотримання всіх процесів розробки? В двадцять першому столітті для бізнесу дуже важливо що б команди ефективно розробляли продукти програмного забезпечення і це відповідало вимогам замовника. Актуальність теми також розкриває питання що найважливішим елементом є комунікація людей між собою та доступ до публічної інформації . Дослідження також охоплює аналіз проблеми налагодження комунікації студентів та варіанти її вирішення. Дослідження рівня інформованості про діяльність вищого навчального закладу і способи донесення важливої інформації про події. Завдяки стрімкому зростанню популярності платформ спілкування під назвою месенджери стала можливою розробка програм на базі платформ що вже існують.

Мета дослідження. Розробка програмного засобу комунікації з використанням API готового продукту Telegram. Дослідження методики та варіантів розробка програмного забезпечення та вибір найкращої платформи для використання телеграм боту, що виступає програмним продуктом в даному дослідженні. Іншою задачею є розробка боту на платформі телеграм для підвищення ефективності навчального процесу шляхом спрощення організації взаємодії студентів між собою та забезпечення більш зручного доступу до інформаційних матеріалів

Завдання дослідження. Завданням дослідження є проєктування та реалізація програмного продукту, який допомагає в спілкуванні та надає простий доступ до інформації.

Об’єкт дослідження. Забезпечення можливості налагодження комунікації між студентами та доступу до інформації. **Предметом дослідження** є структура та реалізація системи комунікації

Методи дослідження. Використано метод інтерв’ювання майбутніх користувачів та роботи з документацією.

Практична цінність. Покращення процесу комунікації та знаходження нових соціальних контактів в новому середовищі (наприклад: новий студент в університеті).

Елементи наукової новизни. Розробка конкретного додатку з дотриманням загальновідомих процесів розробки на платформі з відкритим API.

Практична значущість роботи. Додаток являє собою сервіс, що надає користувачу можливість доступу в одному місці до системи комунікації, без залучення окремих програм для введення в дану систему.

Апробація результатів роботи. Система представлена у вигляді віконного застосування з легким та зрозумілим у використанні інтерфейсом та широкою функціональністю що надає змогу одразу знайти потрібну інформацію.

РОЗДІЛ 1. Постановка задачі та основні принципи роботи методології розробки

1.1 Аналіз методології розробки програмного забезпечення

Метою даного дослідження є розробка Python додатку у вигляді боту на платформі месенджера Telegram на базі бібліотеки Telebot використовуючи відкриту документацію яка знаходиться в мережі Інтернет. Наступною задачею є детальний аналіз і вибір методології розробки та слідування процесам які є в даній методології.

Методологія - сукупність методів, які застосовуються в будь-якій області людської діяльності. Надалі потрібно розуміти методологію як сукупність методів, що застосовуються в життєвому циклі та об’єднаних загальними філософським та технічним підходом. Методологія науки дає характеристику компонентів наукового дослідження - його об'єкта, предмета аналізу, задач дослідження, сукупності дослідницьких засобів, необхідних для вирішення завдань даного типу, а також формує представлення про послідовності руху в процесі вирішення завдань. Методологія пов'язана також з загальним алгоритмом розробки телеграм боту. Алгоритм в такому випадку – це детальний список послідовних кроків та інструкцій які потрібні для створення програми для розв'язання проблеми. З технічної точки зору комп'ютери слідують «крокам» алгоритму для слідування інструкціям за для викання поставленої задачі. З точки зору методології алгоритм являє собою заздалегідь відомі інструкції які повинні привести розробника або команду до створення програмного продукту який вирішує проблему. В такому випадку бачимо аналогічність даних термінів. Для вибору і побудови алгоритму розробки було проаналізовано наявні методології і зроблено висновок яка підходить в конкретному випадку. Для побудови чат-боту підходить методологія Agile. Також потрібно зрозуміти загальну алгоритмічну структуру телеграм ботів, його частин і можливостей взаємодії. Наступним етапом є визначення платформи на якій буде розроблятися та існувати бот. Далі треба провести детальне дослідження даної платформи за допомогою документації та вивчення API та способів комунікації в клієнт-серверній архітектурі.

Розглянуто декілька методології і детальних етапів розробки, визначенні відмінності та плюси та мінуси. При цьому описані засоби використання та зазначені конкретні особливості. На базі описаної вище інформації вибрати методологію та алгоритм розробки боту

Для професійного підходу в розробці важливою складовою є методологія розробки програмного продукту. В даному розділі будуть проаналізовані та вивчені основні методології розробки. Це важлива частина на всіх етапах життя програмного забезпечення.

Будь-яка теоретична або практична сфера діяльності використовує властиві тільки її способи розв'язання поставлених задач. Саме ці способи називаються методами. Метод - це спосіб досягнення якої-небудь цілі способом розв'язання конкретної задачі; сукупність прийомів або операцій практичного або теоретичного освоєння дійсності.

Методологія створення програмного забезпечення полягає в організації побудови інформаційної системи та забезпечення управління цим процесом для того, щоб гарантувати виконання вимог як до самої системи, так і до характеристик процесу розробки.

Основними задачами, рішення яких повинні забезпечити методологію створення програмних продуктів та інформаційних систем, є наступними:

1. Забезпечення створення інформаційних систем, які відповідають цілям і завданням замовника та відповідних вимог до них;
2. Гарантія створення системи із заданими параметрами в рамках заданого часу в рамках обговореного заздалегідь бюджету;

Основними показниками добре розробленою методології це простота супроводження, модифікації та розширення системи з метою забезпечення її відповідності змінюються умовами роботи замовника; Забезпечення та створення інформаційних систем, відповідних вимогам відкритості та масштабності; можливість використання в створюваній системі раніше розроблених засобів інформаційних технологій (програмне забезпечення, бази даних, засоби вичислювальної техніки, комунікацій телефонії і так далі).

На даному етапі розвитку цифрового домінування існує не так багато методології, особливо повних, тобто тих що враховують всі рівні життєвого циклу програмного забезпечення. Саме методологія визначає, які мови та системи будуть застосовуватися для розробки програмного забезпечення та дає бачення який технологічний підхід буде при цьому використано, а також дає основні алгоритми спілкування команди для підвищення ефективності розробки.

Аналізуючи досвід компаній зрозуміло що для ефективного керування проектом потрібна методологія розробки. Щоб ефективно керувати проектом, менеджер або команда розробників повинні вибрати методологію розробки програмного забезпечення, яка найкраще підійде для даного проєкту. Усі методики мають різні сильні та слабкі сторони та існують з різних причин. Нижче наведено найбільш часто використовувані методологій розробки програмного забезпечення та та розкрито тему чому існують різні методології. Після розкриття цієї теми буде обрано методологію для розробки телеграм-боту враховуючи усі деталі.

1.2 Методологія з огляду студента вищого навчального закладу

Прогресивна цифровізація бізнесу, економіки та суспільства ставить вищі навчальні заклади у центр дискусії про те, як ефективно реагувати на виклики та можливості, які таким чином запускаються. Кілька аспектів цього процесу і існують відповідні проблеми, включаючи складне питання, як узгодити навички та компетенції студентів із запитами та очікуваннями галузі і налагодити комунікацію між студентами. Трьома ключовими вимірами цього питання є:

- Чи будуть студенти мати комунікацію з іншими студентами?
- Чи збереже виш конкурентну перевагу цих студентів, таким чином забезпечивши інтегрування нових студентів до активностей,

- Чи буде сам виш вважатися достатньо конкурентоспроможним, щоб залучити кращих майбутніх студентів котрі можливо в майбутньому створять стартап чи наукове відкриття.

З іншого боку, враховуючи змінюючи характер роботи, процес, який різко прискорився під час пандемії Covid-19, вищі навчальні заклади відповідають за оснащення майбутніх співробітників навичками, необхідними для підтримання комунікації у віртуальних, розподілених, культурно різноманітних і часто глобальних груп. Тож формат повинен мати онлайн вигляд з формуванням на різні рівні комунікації і мати кінцеві комунікації вже в реальному житті.

Дана робота має на меті першочергово дати можливість швидкого інтегрування в навчальний процес та надання потрібної інформації за запитом.

Для цього в цій роботі розглядаються можливості та застереження розробки програмного забезпечення навчання, що пропонується студентам старших курсів, особливо з точки зору двох підходів до розробки програмного забезпечення, тобто водоспаду та scrum. Стверджується, що поки водоспад підхід має певні переваги, agile методи, включаючи scrum, створюють можливості, мною було вибрано технологію скрам.

З одного боку, створити робоче середовище, наближене до реальних викликів світу для залучених студентів у проєкт з розробки програмного забезпечення, а з іншого боку, для надання студентам навичок вимагає галузь, і ринок праці в цілому. Звичайно, в контексті вищих навчальних закладів питання підходу до розробки програмного забезпечення зазвичай є складним питанням, оскільки воно пов’язане з такими питаннями як організація та координація виконання проєкту, узгодження з академічним календарем, вимоги до навчального плану, зобов’язання щодо акредитації а також про переваги факультету та доступність часу. З цих причин питання не в тому, чи продовжувати застосовуючи метод водоспаду, а скоріше, як успішно перейти до гібрида чи спритності методи, у тому числі приклад реальної розробки.

1.3 Методологія Agile розробки

Ця методологія допомагає чітко реалізувати та представити готову частину продукту та може бути документованою але ставить перш за все спілкування команди і надання готового програмного забезпечення. Ітераційні процеси на даному етапі пов'язані і з аналізом даних і на надання зворотного зв’язку від замовника. Тож Agile методологія використовується коли є не чіткі вимоги і пріоритети розробки можуть швидко змінюватися. Адаптація до вимог замовника та потреб ринку це саме та частина яку часто виділяють в цій методології. При використанні даної методології вона представляє собою послідовність багатьох ітерацій, кожна з яких має час від тижня до чотирьох але може змінюватися в залежності віз ситуації. Команда заздалегідь визначає пріоритети і що саме буде робити у цей відрізок часу, данна виділена частина часу називається спринт. По результатам кожної ітерації вимоги до продукту уточняються та за необхідності змінюються. Ітерації можуть повторюватися за потреби. Але очевидно що втрати часу будуть вже менші аніж при інших методиках розробки. Основним аспектом даної методології є те що ведення процесів менш бюрократичне та швидше, бо документація відходить на другий план. Важливо що команда не відмовляється від документування різних частин продукту, просто виставляє пріоритети в сторону особистої комунікації й вимог. То ж основні акценти зміщенні на реалізацію що у нашому випадку підходить, бо надання готового боту для нас важливіше аніж документація.

В той же час зі сторони замовника є можливість зрозуміти який продукт потрібен користувачам на практиці, методом спроб та помилок. Тут важливо швидко проводити опитування або іншим способом дізнаватися враження користувачів і чи вирішу продукт їх проблеми. В такому випадку в замовника може бути рішення яке задовільне не тільки вимоги які були на початку розробки, а і які з'явились з часом що в кінцевому результаті дає актуальне програмне забезпечення. В розробника увесь час дуже щільна взаємодія з командою, він постійно перебуває в курсі етапу та стадії розробки, бере участь

в плануванні усіх етапів та може аналізувати результати.

Команди використовують методологію гнучкої розробки, щоб мінімізувати ризики (наприклад, помилки, перевищення витрат та зміни вимог) під час додавання нових функцій. У всіх методах Agile команди розробляють програмне забезпечення на ітераціях, які містять міні-збільшення нової функціональності. Існує багато різних форм методу гнучкої розробки, включаючи scrum, кристал, екстремальне програмування (XP) і розробку на основі функцій (FDD). В роботі я буду основою команди. Методологія повинна слідувати та підпорядковуватися наступним правилам:

1. Використання системи контролю версій
2. Оформлення коду повинно бути стандартизоване
3. Виправлення помилок першочергове до внесення змін
4. Обов’язкове виконання регулярного резервування копіювання всіх проектних даних
5. Використання засобів автоматизації документного вихідного коду та ведення знайдених помилок (багів)
6. Різні види тестування на різних етапах, створення тестових середовищ.

Також при великій команді в цій методології для покращення комунікації є щоденні зустрічі де кожен з команди має відповісти на три питання: Що він робив вчора? Що буде робити сьогодні? та Які брокери заважають далі працювати? Також по закінченню відрізку часу який було виділено для розробки частини програми проводиться зустріч під назвою Рефлексія де аналізується спринт та що було зроблено ефективно, а шо можна буде покращити в наступний раз.

Плюси: основна перевага гнучкої розробки програмного забезпечення полягає в тому, що вона дозволяє випускати програмне забезпечення в ітераціях. Ітераційні випуски підвищують ефективність, дозволяючи командам знаходити та виправляти дефекти та вирівнювати очікування на ранніх стадіях. Вони також дозволяють користувачам реалізувати переваги програмного забезпечення

раніше, з частими додатковими вдосконаленням.

Мінуси: agile методи розробки покладаються на спілкування в реальному часі, тому новим користувачам часто не вистачає документації, необхідної для швидкого ознайомлення. Вони вимагають великих витрат часу від користувачів і трудомісткі, оскільки розробники повинні повністю виконати кожну функцію протягом кожної ітерації для схвалення користувачем.

Гнучкі методи розробки подібні до швидкої розробки додатків і можуть бути неефективними у великих організаціях. Програмісти, менеджери та організації, які звикли до методу водоспаду, можуть мати труднощі з адаптацією до гнучкої SDLC. Тому гібридний підхід часто підходить для них.

1.4 Методологія розгортання DevOps

DevOps — це методологія розробки програмного забезпечення, яка поєднує розробку програмного забезпечення (Dev) з операціями з інформаційних технологій (Ops), які беруть участь разом у всьому життєвому циклі служби, від проектування через процес розробки до підтримки виробництва.

Цілі DevOps:

- Методології швидкого розвитку
- Методи швидкого забезпечення якості
- Методи швидкого розгортання
- Швидший час виходу на ринок

Ітерація та постійний зворотний зв’язок (потужне та безперервне спілкування між зацікавленими сторонами — кінцевими користувачами та клієнтами, власниками продуктів, розробниками, гарантіями якості та інженерами з виробництва)

Переваги DevOps:

- Стабілізація навколишнього середовища
- Підтримує послідовність, збільшує час роботи
- Коротший цикл розвитку

- Керування вимогами та репозиторієм коду
- Збільшена швидкість вивільнення
- Безперервна збірка, розгортання кнопками
- Зменшені дефекти
- Полкові процеси, автоматизоване тестування

Метрики процесу:

Відстежуйте як час на кожному етапі, також помилки та винятки

Етапи DevOps:

1. План: управління завданнями, розклади
2. Код: розробка коду та рецензування коду, інструменти керування вихідним кодом, злиття коду
3. Збірка: інструменти безперервної інтеграції, інструменти контролю версій, статус збірки
4. Тест: інструменти безперервного тестування, які надають зворотній зв’язок щодо бізнес-ризиків, визначають ефективність
5. Пакет: сховище артефактів, етап перед розгортанням програми
6. Випуск: керування змінами, затвердження випусків, автоматизація випуску
7. Керування: встановлення інфраструктури, зміни інфраструктури (масштабованість), конфігурація інфраструктури та управління, інфраструктура як інструменти коду, планування потужності, управління потужністю та ресурсами, перевірка безпеки, розгортання послуг, висока доступність, відновлення даних, керування журналами/резервним копіюванням, база даних управління
8. Моніторинг: моніторинг продуктивності послуг, моніторинг журналів, досвід кінцевих користувачів, управління інцидентами

DevOps — це не просто методологія розробки, а й набір практик, які підтримують організаційну культуру. Розгортання DevOps зосереджено на організаційних змінах, які покращують співпрацю між відділами,

відповідальними за різні сегменти життєвого циклу розробки, такі як розробка, забезпечення якості та операції.

Методологія розгортання DevOps:

Плюси: DevOps зосереджений на покращенні часу виходу на ринок, зниженні частоти збоїв у нових випусках, скороченні часу між виправленнями та мінімізації збоїв, максимізуючи надійність. Щоб досягти цього, організації DevOps прагнуть автоматизувати безперервне розгортання, щоб все відбувалося гладко та надійно. Компанії, які використовують методи DevOps, отримують вигоду, значно скорочуючи час виходу на ринок і покращуючи задоволеність клієнтів, якість продукції, а також продуктивність і ефективність співробітників.

Мінуси: навіть у світлі переваг DevOps має кілька недоліків:

Деякі клієнти не хочуть постійно оновлювати свої системи.У деяких галузях є правила, які вимагають ретельного тестування, перш ніж проект може перейти до етапу експлуатації.

Якщо різні відділи використовують різні середовища, невиявлені проблеми можуть проскочити у виробництво. Деякі атрибути якості вимагають взаємодії з людиною, що уповільнює конвеєр доставки. Та треба розуміти що повна автоматизація на разі не можлива тому комунікація все ж потрібна на будь з яких етапів.

1.5 Метод розробки водоспаду

Багато хто вважає метод водоспаду найбільш традиційним методом розробки програмного забезпечення. Метод водоспаду — це жорстка лінійна модель, яка складається з послідовних етапів (вимоги, проектування, впровадження, перевірка, технічне обслуговування), зосереджених на окремих цілях. Кожна фаза повинна бути завершена на 100%, перш ніж почати наступний етап. Зазвичай немає процесу повернення, щоб змінити проект або напрямок. До переваг часто відносять :

- Кожна стадія гарантує формування проектної документації

- Усі стадії робіт відомі заздалегідь і виконуються в логічній послідовності, що дозволяє планувати чіткі терміни робіт
- Розрахунок витрат чіткий так прозорий та відомий ще до початку робіт

Каскадна модель може використовуватися при створені програмного забезпечення в якому з самого початку відомі чіткі вимоги і що головне вони не будуть змінюватися в час розробки.

Розробка методом водоспаду є першою моделлю що мала великий успіх в широких колах і мала популярність серед багатьох компаній. Багато компаній виділяли що ця методологія структурує процеси розробки. Кожна стадія моделі закінчується з отриманням деяких результатів, які служать як початкові вхідні данні для наступної стадії. Окремо треба зазначити вимоги до розробки програмного забезпечення це заздалегідь визначені стадії, сформовані вимоги, суворо та чітко сформована документація яка має вигляд технічного завдання. Усі етапи чітко фіксуються увесь час проєкту.

Плюси: Лінійний характер методу розробки водоспаду полегшує розуміння та керування ним. Проекти з чіткими цілями та стабільними вимогами найкраще можуть використовувати метод водоспаду. Менш досвідчені менеджери проектів і проектні групи, а також команди, склад яких часто змінюється, можуть отримати найбільшу користь від використання методології розробки каскаду.

Мінуси: метод розробки водоспаду часто є повільним і дорогим через його жорстку структуру та жорсткий контроль. Ці недоліки можуть спонукати користувачів водоспаду до вивчення інших методологій розробки програмного забезпечення.

1.6 Швидка розробка додатків

Швидка розробка додатків (RAD) — це стислий процес розробки, який створює високоякісну систему з низькими інвестиційними витратами. Цей процес RAD дозволяє розробникам швидко пристосуватися до вимог, що змінюються, на ринку, що також швидко змінюється. Можливість швидкого

регулювання – це те, що дозволяє знизити вартість інвестицій.

Метод швидкої розробки додатків містить чотири фази: планування вимог, проєктування користувача, конструювання та перемикання. Фази проектування та конструювання користувача повторюються, поки користувач не підтвердить, що продукт відповідає всім вимогам.

Провівши детальний аналіз методології швидкої розробки можна зробити висновок що вид методології підходить в стартапах. Основні три частини з яких складається дана методологія :

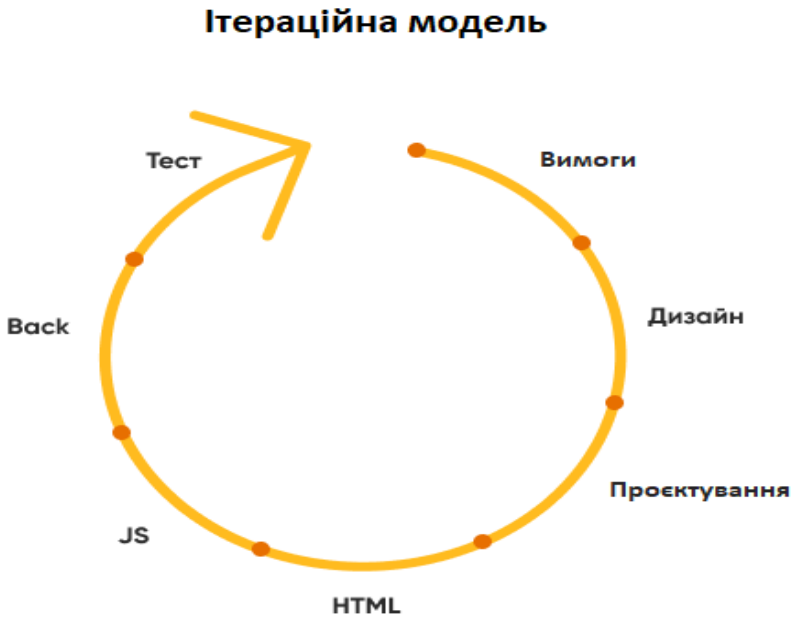


Рис. 1. Основні етапи моделі

- Низька вартість
- Висока якість
- Велика швидкість

Життєвий цикл розробки програми поділяється на підходи і складається з чотирьох основних етапів:

- Реалізація
- Впровадження
- Проектування
- Аналіз вимог

При використанні швидких методів для інтерфейсу користувача системи зазвичай створюються інтерактивними засобами. Замість довгого виконання розробки коду та прототипу вона дозволяє працювати з графічними піктограмами, що представляють функції, або компоненти інтерфейсу користувача, і відповідні сценарії управління цими піктограмами. Програма, готова до виконання, генерується автоматично з візуального представлення системи. Це спрощує розробку графічного інтерфейсу та зменшує фінансові витрати на системи прототипування.

Слід окремо зазначити, що методи швидкої розробки, такі як RAD, особливо ефективні при високих проектних ризиків, наприклад, неясних цілях проекту, недокументованих програмних процедурах, нестабільних вимогах від замовника. Незважаючи на те, що ефективність таких розробок досягається за рахунок підвищення трудомісткості і матеріальних витрат, довготривалі вкладення в ці методи можуть окупитися набагато швидше інших.

Технологія швидкої розробки не може претендувати на універсальність, вона хороша в першу чергу для відносно невеликих проектів, що розробляються в короткі сроки для конкретного замовника. Вона непридатна для розробки операційних комп’ютерних систем; складних розрахункових програм з великим об'ємом програмного коду і складними унікальними алгоритмами управління; додатків, в яких відсутня яскраво виражена інтерфейсна частина, наочно визначає логіку роботи системи (додатки реального часу), так як ітераційний підхід передбачає, що кілька перших версій не будуть повністю відповідати вимогам.

Висновки розділу 1

Були розглянуті основні методології розробки програмного забезпечення, розглянуто основні етапи при використанні. Охарактеризовані основні чинники що впливають на збільшення продуктивності при використанні методологій. Було проведено ознайомлення з останніми публікаціями що стосуються розвитку даних методологій у сучасному світі розробки.

Проведено дослідження та наведені плюси і мінуси використання кожної методології. Проаналізовано сучасні інструменти та види документації для розробки програмного забезпечення. Описані засоби використання та зазначені конкретні особливості. Розкрито вплив методології розробки на сам процес побудови програмного забезпечення та розглянуто ролі в команді та зони відповідальності.

РОЗДІЛ 2. Алгоритмічне забезпечення життєвого циклу програми

2.1 Життєвий цикл розробки

Життєвий цикл розробки програмного забезпечення (SDLC) є одним із ключових питань у програмній інженерії та покращення процесів розробки. Це тому, що структура діяльності, необхідна для розробки програмної системи так само важлива як і якість та зручність використання розробленого програмного забезпечення. Тож при розробці бота який виступає програмним забезпеченням в даній роботі треба розуміти усі етапи розробки. Загальний SDLC складається з п'яти етапів, включаючи: аналіз вимог, проєктування програмного забезпечення, впровадження програмного забезпечення та тестування програмного забезпечення. Поверхневий аналіз показує що фаза аналізу вимог — це процес фіксації вимог користувача та створення специфікацій програмного забезпечення, проте існують різні варіації. Під час розробки програмного забезпечення, абстрактні програмні моделі розробляються через ілюстрацію різних точок зору використання мов моделювання, таких як Unified Modeling Language (UML). Фаза впровадження програмного забезпечення – це процес побудови виконуваного програмного забезпечення на основі моделі, створені на попередній стадії проєктування з заздалегідь відомою ідеєю. Фаза тестування програмного забезпечення має на меті переконатися в цьому розроблене програмне забезпечення відповідає попереднім специфікаціям і не містить критичних помилок.

Життєвий цикл розробки програмного забезпечення — концепція, яка надзвичайно важлива, бо мати глибоке розуміння як інженер-програміст це потрібно не зважаючи на те що сам програміст не приймає участь у всіх етапах. З цим проєктом моєю метою було концептуально вивчати складності кожного кроку та застосовувати свої навички до розробки реального додатку який виступає у давній роботі об'єктом роботи. SDLC включає наступні етапи: планування та аналіз потреб, проєктування та розробка, впровадження,

тестування, інтеграція та обслуговування. Щоб застосувати ці концепції, я створив бот-додаток, щоб користувачі могли планувати надсилання повідомлень надіслано в майбутній час і дату.

Сам по собі життєвий цикл розробки програмного забезпечення являє собою структурний вид представлення розробки програмного продукту в якому описані всі кроки його створення і етапи існування.

Життєвий цикл розробки програмного забезпечення був широко адаптований через більшість, якщо не всі, технологічні компанії. З розвитком технологій і зростає кількість компаній, які покладаються на власні користувацькі програми, навчання та розуміння складнощів SDLC стає все більш важливим тож в цій роботі ця тема розкривається детальною. Звичайно знання, як кодувати, є основою для будь-якої кар'єри розробника програмного забезпечення, але розуміння SDLC приносить багато нових баз навичок, усі вони корисні для інженерних процесів. Ці навички включають планування, проєктування та тестування. SDLC традиційно має десять фаз, які включають ініціювання, розробку концепції системи, планування, аналіз вимог, проєктування, розробка, інтеграція та тестування, впровадження, експлуатація та обслуговування, а також розпорядження. Важливо розуми що в залежності від масштабу проєкт та контексту кількість фаз може змінюватися. Дуже складні та великі проєкти повинні розділяти кожен етап на менші та більше детальні, бо потребують деталізації та розгорнутої документації.

В сучасному світі не всі етапи необхідні в кожній ситуації і часто або пропускаються, або поєднуються разом.

Усі ці етапи будуть більш детально розглянуті далі в роботі. Крім того, варіації SDLC, включаючи все більш важливі аспекти, присвячені безпеці в розробці програмного забезпечення, будуть розглянуті далі.

Тут створено та проілюстровано бот-додаток, створений з метою навчання та розуміння на практиці фази, продемонстровані в SDLC. Додаток що написаний дозволяє користувачам планувати надсилання повідомлень будь-якій особі та доступ до бази корисної інформації. Він написаний на мові

програмування Python і замовником цього проєкту виступив я, який вивчає подібні концепції. Деталі програми та важливі фрагменти коду будуть розглянуті пізніше в розділі номер 3.

Життєвий цикл розробки програмного забезпечення (SDLC) - це серія кроків, яким слідує розробник або команда для розробки та впровадження свого програмного забезпечення. Немає єдиного, уніфікованого життєвого циклу розробки програмного забезпечення. Швидше, існує кілька рамок і моделей, яким слідують команди розробників для створення, тестування, розгортання та підтримки програмного забезпечення.

Основними повними етапами SDLC є:

1. Ідея
2. Визначення вимог
3. Дизайн (архітектура) системи
4. Розробка
5. Тестування
6. Розвиток
7. Підтримка

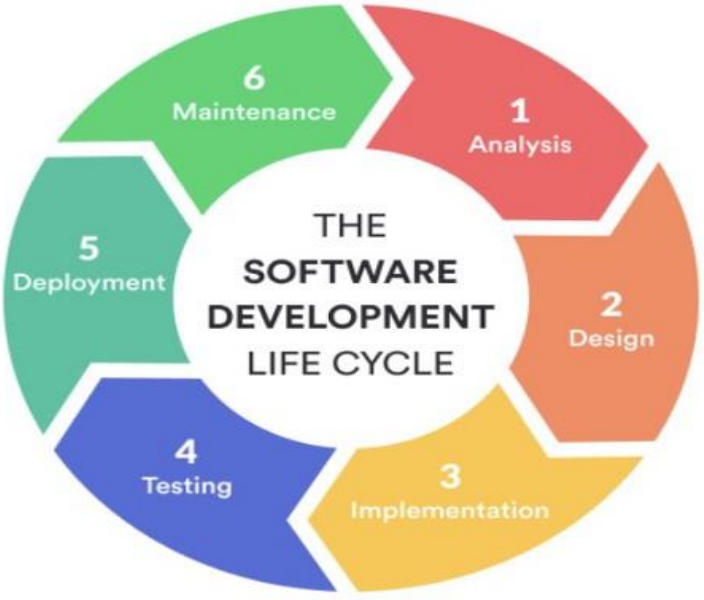


Рис 2. Схема циклу розробки програмного забезпечення

2.2 Ідея та об'єкт розробки

Іноді цю фазу називають «Horizon» і вона визначає потребу або можливості в бізнес-одиниці або просто охарактеризовану суть програмного продукту. Фаза Horizon в бізнес-процесі розпочинає старша зацікавлена сторона бізнесу або відповідальна людина з технологічного керівництва. Заходи включають в себе:

- Попередній аналіз доцільності
- Охоплення вимог високого рівня
- Узгодженість бізнесу
- Визначення та призначення ресурсів планування
- Визначення бізнес-спонсора та технологічного менеджера

IT-операції має вести:

1. Технічний керівник
2. Розробник Тренер
3. Бізнес-аналітик
4. Кінцевий користувач
5. QA Analyst Архітектор
6. Рецензент коду Менеджер випуску
7. Пропозиція концепції
8. Менеджер проєкт

Наступний крок процесу: концепційна пропозиція

Під час цього кроку бізнес-аналітик фіксує вимоги спонсора на високому рівні, бачення рішення, масштаби і обмеження проекту, а також бізнес-контекст, в якому буде працювати готовий проект.

Діяльність на даному етапі включає в себе так етапи:

- Менеджер технологій вводить проект у SharePoint. Заповнюються такі поля: Ставиться визначення статусу проекту вводиться в SharePoint як Horizon етап розробки
- Технологічне лідерство яке має повноваження визначає керівника проекту, а бізнес-аналітик входить до керівника проекту
- Бізнес-аналітик пише документ пропозицій концепції :
- Працює із зацікавленою стороною та менеджером проекту за потреби
- За потреби можна консультуватися з розробниками.
- Концепційна пропозиція (бізнес-аналітик)
- Прототип (необов'язково) (розробник)

Етап — Погодження:

- Business Analyst додає дію «Затвердження: ініціація проекту» у примітки SDLC для SharePoint пункт.
- Спонсор додає дію «Схвалення: пропозиція концепції» у примітки SDLC.

Ідея є дуже важливою б кожен проект починається саме з цього. Саме на цьому етапі можна зрозуміти унікальність проекту та вибудувати стратегію просування. В нашому випадку ідея має аналогічну структуру як мета дослідження та ідея дослідження. Ідея сформована в співпраці з наочним керівником що моделює собою процес описаний вище. Було зважено ризики та оцінено актуальність даного продукту. Узгоджено структуру та планування ресурсів, попередньо проаналізовано доцільність та охоплення вимоги. У якості визначення та призначення ресурсів я виступаю головним технологічним менеджером даної розробки телеграм боту. Прототип роботи буде розглянуто у третій частині.

2.3 Визначення вимог та початок документування

Метою цього етапу є забезпечення певної міри обґрунтованого очікування того, що надійне, ефективне програмне забезпечення може бути створене вчасно та в межах витрат, щоб задовольнити запит клієнта. Ця частина

добре підходить для ілюстрації розробки програмного продукту в межах дипломної роботи бо ми маємо обмежений час та ресурси і клієнтом можу виступати я.

На цьому етапі SDLC необхідно отримати зворотний зв'язок та підтримку від відповідних внутрішніх та зовнішніх зацікавлених сторін. Це процес який передбачає отримання вимог від клієнтів. Бізнес-аналітики зустрічаються з клієнтами та збирають їхні вимоги (щодо програмного забезпечення, що розробляється). Ці аналітики використовують зібрану інформацію для підготовки специфікації бізнес-вимог (BRS). Потім готується документ зі специфікацією вимог, щоб надати вказівки для інших етапів SDLC.

Крок процесу: бачення та обсяг / функціональна специфікація в розробці

Використання документів «Пропозиція концепції» як ті що використовується керівництвом, бізнес-аналітик створює документ «Бачення та обсяг» та робить презентацію для замовника або інвестора. В моєму проекті також підготовлено презентацію яка буде представлена при захисті даної роботи.

Документ функціональної специфікації. (Примітка: для проектів меншого розміру або тривалості технологічний менеджер не буде вимагати документ «Бачення та обсяг».) Доволі часто все ж на проектах створення документа «Бачення та обсяг» є надзвичайно важливим і потребують спільних зусиль бізнес-аналітика та спонсора проекту. Хоча явного схвалення не потрібно, бізнес-аналітик повинен мати достатню участь під час створення даного документу, перш ніж приступити до Функціональної специфікація. Схвалення бачення та обсягу мається на увазі в концепції функціонального огляду самої специфікації, де спонсор затверджує функціональну специфікацію. В даному випадку зважаючи що проект не великий ми можемо відмовитися вид даного виду документу.

Документ функціональної специфікації який згадувавсявище зосереджений на бізнес-потребах і націлений на технічного спеціаліста, кінцевого користувача, контроль якості та розробника яких має виконуватися аудиторією що буде використовувати данні документи. Схвалена спонсором або

замовником Функціональна специфікації формує контракт на обсяг проекту, і дозволяє команді розпочати роботу над іншими плановими документами.

Створення плану проекту починається з визначення структури роботи проекту (WBS), який в свою чергу розділяє проект де може працювати і розробляти завдання надані керівником та логічно впорядковувати їх, щоб забезпечити плавну еволюцію між завданнями. Конкретні віхи потім визначаються в в структурі роботи, кількість яких залежить від розміру проекту. Віхи служать щоб позначити ключові події та привернути до них увагу керівництва. Після визначення WBS ресурси призначаються до кожного елемента роботи. Потім надається оцінка часу, необхідного для виконання кожного пункту. Якщо звернутися до першої частини роботи саме за таким принципом розділять задачі в методології скрам в розділяють роботу на строрі-поінти. Цей план потім стає розкладом, який коригується протягом життя проекту відповідно до вимог і використовується для відстеження прогресу. Відстеження та підтримка плану проекту – це діяльність, яка триває протягом інших фаз проект. Таким чином, основна відповідальність менеджера проекту включає постійну співпрацю з командою і є дуже важливою на даному етапі.

Команда керівництва, щоб точно відображати дати реалізації проекту в календарі випуску також може розробляти документ розрахунку часу проекту. Коли отримані відповіді задовольнять усіх учасників, ви зможете перейти до наступної фази.

2.4 Архітектура системи

Поширені артефакти даного етапу розробки є проектні документи системи які відображають архітектуру системи, структуру компонентів програми або системи, їх взаємозв’язки, а також принципи та вказівки, що керують їх проектуванням і еволюцією з часом. Зв'язки бувають як під час виконання, так і без виконання, і, отже, архітектура також виражається в термінах компонентів і з'єднувачів. Архітектура системи є не лише продуктовою

вимогою, а й результатом організаційних цілей, досвіду архітектора та її технічного середовища. Проєкт системи містить загальні вказівки щодо функцій системи, вимог до продуктивності, вимог безпеки, характеристик платформи та має включати:

- Бізнес-правила/логіка застосування
- Дизайн інтерфейсу (додаток до програми, додаток до БД)
- Інтерфейси користувача (GUI)
- Дизайн бази даних (зберігання даних і доступ до БД)

Деякі додаткові артефакти/документи, створені в результаті діяльності на цьому етапі, включають:

- Словник даних
- Схеми процесу
- Схеми компонування екрана
- Таблиці правил ведення бізнесу
- Прототип/доказ концепції

Документи плану тестування дизайну та архітектури включають заплановані тестові дії, які відповідають вимогам користувачів, включаючи опис рівнів тестів, які мають місце під час розробки: інтеграція, безпека системи та приймальні тести користувача, а також необхідне планування. Тестове середовище описується з точки зору етапів, графіків та ресурсів, необхідних для підтримки тестування.

Плани навчання призначені для визначення користувачів і того, як вони будуть навчатися новому рішенню. Зазвичай необхідний для великих проектів. Аналізуючи це можемо замінити даний вид документації на архітектурний стиль з прикладом очікуємо результату від програми яка в розробці.

Посібники з технічного обслуговування включають системні процедури, необхідні для встановлення, налаштування та підтримки системи. Вони створюються на етапі проектування та переглядаються на етапах будівництва та тестування та завершуються на етапі впровадження. Документи технічного

обслуговування можуть містити процедури реагування на надзвичайні ситуації; резервні механізми, процедури та відповідальність; а також процедури та відповідальність після процесу відновлення.

Посібники користувача містять інформацію, необхідну для використання системи або компонента для отримання бажаних результатів. Зазвичай описуються можливості системи або компонентів, обмеження, опції, дозволені введення, очікувані виходи, можливі повідомлення про помилки та спеціальні інструкції. Посібник користувача відрізняється від керівництва оператора, коли розрізняють тих, хто керує комп'ютерною системою (монтажні стрічки тощо), і тих, хто використовує систему за призначенням. Враховуючи що бот якого ми розробляємо буде розміщено на платформі телеграм що передбачає що користувачі вже мають досвід роботи з ботами. То ж даний вид документації потрібен для внутрішнього використання. Приклад рис 3.

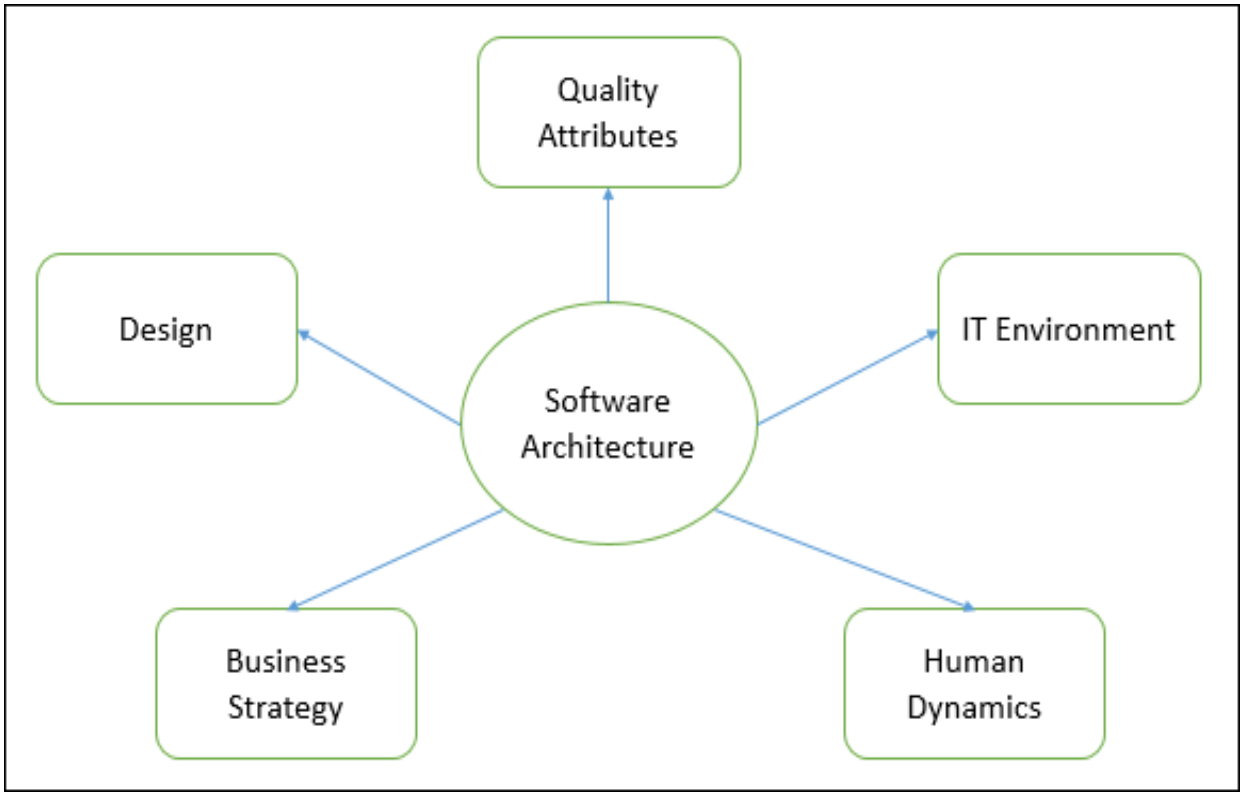


Рис 3. Складові частини архітектури

2.5 Розробка

Метою цього етапу є перетворення затвердженої архітектури та дизайну в робочу систему, яка відповідає функціональним і технічним вимогам, визначеним на попередніх етапах життєвого циклу програмного забезпечення. Після завершення етапу проектування системи наступним етапом є кодування. На цьому етапі розробники починають будувати всю систему, пишучи код за допомогою обраної мови програмування. В даній роботі було обрано мову програмування Python що являє собою інтерпретовану об’єктно-орієнтовану мову. На етапі кодування завдання поділяються на блоки або модулі і призначаються різним розробникам. Це найдовша фаза процесу життєвого циклу розробки програмного забезпечення що відображено у розрахунку часу дипломної роботи.

На цьому етапі розробнику необхідно дотримуватися певних попередньо визначених правил кодування. Їм також потрібно використовувати інструменти програмування, такі як компілятор, інтерпретатори, налагоджувач для створення та реалізації коду. Приклади вибору інструментів даного типу будуть розглянуті далі в роботі.

Розробка характеризується залученням розробників для кодування рішення на основі наданих деталей продукту та на основі документації яка була розроблена в попередніх кроках. Саме на цьому етапі можна зрозуміти наскільки важлива документація та розуміння масштабів і ресурсів проекту. Без попередніх етапів неможливо розробити коректний та відповідний вимогам програмний продукт.

Також починається розробка у функціональних і технічних характеристиках. QA Analyst збирає результати модульного тестування розробників у порядку та може коригувати функціональні та структурні плани тестування під час кодування проєкту. Керівник проекту керує змінами до термінів і завдань у плані проекту на основі щоденного прогресу, досягнутого Розробниками. Для великих проектів, Бізнес-аналітик і Кінцевий користувач

проводитимуть додаткові перевірки в міру розвитку, щоб переконатися, що вимоги реалізуються належним чином і підтверджують бачення кінцевого користувача.

2.6 Тестування

Важливою частиною даного етапу е аналіз вимог як загальної документації так й загалом вимог до будь яких тестових артефактів. Аналіз вимог входить на пряму в процес тестування.

Після того, як програмне забезпечення буде завершено, і воно буде розгорнуто в середовищі тестування. Команда тестувальників починає тестування функціональності всієї системи. Це робиться для того, щоб переконатися, що вся програма працює відповідно до вимог замовника.

Рецензент коду, менеджер з випуску, аналітик QA, тренер і кінцевий користувач беруть участь у забезпеченні того, щоб рішення надане відповідає очікуванням з функціональної точки зору, а також із загальної стабільності системи перспектива. Тобто тести можна поділити на два види: функціональні та не функціональні. Головна задача функціонального тестування забезпечити впевненість в коректній роботі головного функціоналу який потрібен замовнику. Дану інформацію ми можемо взяти з документу про специфікацію продукту який був розроблений на етапі документування та архітектури. Функціональне тестування забезпечується за допомогою наступних видів тестування : модульне, інтеграційне, системне, регресійне, приймальне. Відповідно до не функціонального виду тестування відносяться усі інші види. Тести які не повинні відповідати вимогам технічної специфікація але також е не менше важливими. До не функціонального тестування відносяться такі види: надійність в стресових ситуаціях, продуктивність, зручність, можливість масштабування, безпека, можливість перенести програмний продукт на іншу платформу. Тобто головна ідея не функціонального тестування перевірити що програмне забезпечення залишає гране враження користувачу.

На цьому етапі команда контролю якості та тестування може виявити деякі помилки/дефекти, про які вони повідомляють розробникам. Команда розробників виправляє помилку та надсилає до QA для повторного тестування що називають ретестінгом. Цей процес триває до тих пір, поки програмне забезпечення не стане безпомилковим, стабільним і працюватиме відповідно до бізнес-потреб цієї системи.

При тестуванні програмного забезпечення треба розуміти 7 принципів тестування, вони дають можливість адаптувати усю концепцію тестування загалом.

До семи принципів тестування відносять :

1. Тестування показує що дефекти є, а не їх відсутність.
2. Кластеризація дефектів.
3. Парадокс пестициду.
4. Вичерпне тестування не можливе.
5. Принцип раннього тестування.
6. Спосіб тестування залежить від контексту.
7. Відсутність помилок це ілюзія.

Також один із видів тестування це передати програмний продукт справжнім користувачам і збирати зворотній зв’язок, в цього способу є недоліки бо якщо програма дужу поганої якості користувачі можуть перестати її використовувати і в них залишаться погані враження що в свою чергу може надалі може погано вплинути на розвиток програмного продукту. Саме тому даним способом не користуються великі компанії. Але враховуючи обмежені ресурси ми можемо провести димне тестування (Smoke Testing), він перевіряє основний функціонал і що програмний продукт можна використовувати і основна функціональна ідея працює.

2.7. Розвиток та підтримка

Під час фази розгортання команда розгортає основні технології та компоненти продукту, стабілізує розгортання, передає проект на експлуатацію та підтримку та отримує остаточне схвалення проекту замовником. Стабілізуючий діяльність може продовжуватися протягом цього періоду, оскільки компоненти проекту будуть перенесені з тестового середовища в виробниче середовище але у нашому випадку нема розділення на тестове и середовище для користувачів. Це зумовлено тим що платформа телеграм е вільно доступною і фактор який впливе на користування нашою платформою залежить тільки від того коли ми надамо доступ до боту.

Після завершення етапу тестування програмного забезпечення та жодних помилок чи неточностей у системі починається остаточний процес розгортання. На основі відгуків, наданих менеджером проекту, випускається остаточне програмне забезпечення та перевіряється наявність проблем із розгортанням, якщо такі є.

Після того, як система розгорнута, і клієнти починають використовувати розроблену систему, відбуваються наступні 3 дії які можуть існувати на всьому подальшому життєвому циклі програмного забезпечення:

- Виправлення помилок – повідомляють про помилки через деякі сценарії, які взагалі не тестуються
- Оновлення – оновлення програми до новіших версій Програмного забезпечення
- Покращення – додавання деяких нових функцій (фічі) до існуючого програмного забезпечення

Основна увага на цьому етапі SDLC полягає в тому, щоб забезпечити, щоб потреби й надалі задовольнялися, і щоб система продовжувала працювати відповідно до специфікації, згаданої на першому етапі.

Засвоєно що життєвий цикл розробки програмного забезпечення є основою для структурування, планування, і виконання завдань, пов'язаних з

розробкою нашої інформаційної системи . Внаслідок відмінність у вимогах систем поряд з численними іншими відмінностями, варіаціями на SDLC були розроблені. Перед початком проєкт було важливо визначити, який із них є найбільш відповідний, особливо з огляду на суттєве просування веб-технології та месенджерів. SDLC в нашому випадку включає окремі фази, призначені для надання інженерам-програмістам чітко визначеної мети, як працювати . Він спрямований на допомогу розробникам та іншим співробітникам проєкту для створення системи, яка відповідає всім технічним вимогам і вимогам користувачів, як а також перевищує очікування клієнтів. Усі ці фази обертаються навколо центру теми забезпечення якості в умовах невеликих ресурсів.

Попри проблеми з витратами, впровадження безпеки в життєвий цикл програмного забезпечення є надзвичайно важливим. З розвитком технологій і надійністю користувачів в системи, будь-яка вразливість ставить компанію під загрозу. Завдяки початковій підготовці розробники зможуть мати кращу основу для розв'язання проблем, які вони вже знають, що існують у їхній базі коду. Хоча це навчання потребує великих витрат часу та фінансових витрат, воно перевищить ризики, пов’язані з тим, що безпека не буде першочерговим завданням протягом життєвого циклу розробки. Для цього проєктне впровадження цих методів у майбутніх ітераціях є першочерговим завданням, особливо оскільки відкрита розробка все більше популярна.

Головним пріоритетом проєкту є забезпечення якісного програмного забезпечення для користувачів і клієнта. Візуальне та практичне зображення цього циклу показано в наступному розділі

Висновки розділу 2

У цьому розділі було проаналізовано та розглянуто всі етапи створення Telegram боту, в результаті наведено приклади етапів розробки та побудови процесів в умовах обмежених ресурсів. Було розглянуто та повторено весь життєвий цикл за технологією SDLC (System/Software Development Life Cycle).

Досліджено основні етапи побудови додатку та розроблено алгоритмічну частину програми. Наведено блок-схему основних модулів алгоритмічної частини. Проаналізовано сучасні методології та підходи до вирішення задач розробки програмного забезпечення. Досліджено наявні переваги та недоліки методологій. Зроблено покриття техніками тест дизайну та розроблено тестову документацію.

РОЗДІЛ 3. Програмне забезпечення

3.1 Опис та основні функції платформи телеграм

Telegram Bot Api - унікальна платформа з підтримкою майже всіх мов програмування і надає можливість використовувати будь-яку методологію і слідувати життєвому циклу розробки. Представляє для розробки багатьом програмістам без оплати з ним працювати, а головне - дає можливість безплатно створити свій програмний продукт для великої кількості людей, де кожний може взаємодіяти з вашим програмним продуктом завдяки внутрішнім алгоритмам та надавати доступ усім охочим. Також дає можливість створити різного формату продукти та клиенто-сервісні забезпечення, без залучення великої команди, для розробки достатньо знати мову програмування та вміти працювати з форматом JSON або XML.

Розглянемо алгоритм створення боту та отримання унікального індивідуального токену для подальшої роботи з ним. На платформі Telegram є окремий «головний» бот через якого створюються інші боти, за аналогією його названо BotFather що в перекладі на українську означає Бот-батько. Цей бот має багато функцій їх можна поділити на 3 основні частини :

1. Створення і змінення боту. Основні команди :

- /newbot – створення нового боту
- /mybots – додавання боту
- /setname – зміна ім’я боту
- /setdescription – змінити опис боту
- /setabouttext – змінити інформацію про бота
- /setusrpic – встановити/змінити фото профілю
- /setcommands - змінити лист команд
- /deletebot – видалити бота

2. Налаштування боту

- /token – Згенерувати токен авторизації
- /revoke – Відізвати згенерований токен
- /setinline – Переключити вбудований режим
- /setinlinegeo – Переключити вбудований запит місцеперебування
- / setinlinefeedbeck – Змінити налаштування вбудованого зв’язку
- / setinlingroups – Можливість додавати бота в групи
- /setprivacy – Встановити режим приватності

3. Ігри та подібне

- /mygames – Мої ігри
- /newgame – Нові ігри
- /listgame – Перелік ігор
- /editgame – Додати ігри
- /deletegame – Видалити

Використовуючи данні команди ми можемо почати створювати бота. В пошуку телеграм додатку вводимо таку назву «BotFather» і вибираємо перший чат. Далі використовуючи вище зазначені команди починаємо створення боту. Вводимо значення через слеш, перша команда «/newbot» вона дає команду на створення нового боту. Далі вводимо назву боту і наступним кроком юзернейм(важливо що юзернейм повинен бути унікальним в системі).

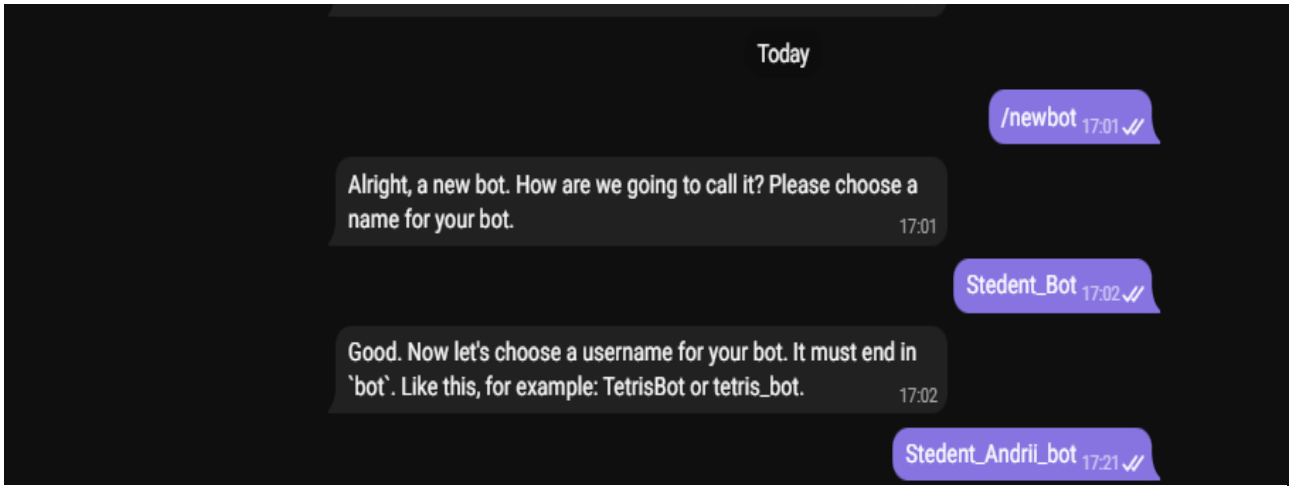


Рис 4. Оголошення назви боту

Наступним етапом ми отримуємо API за протоколом HTTP і посилання на самого бота.

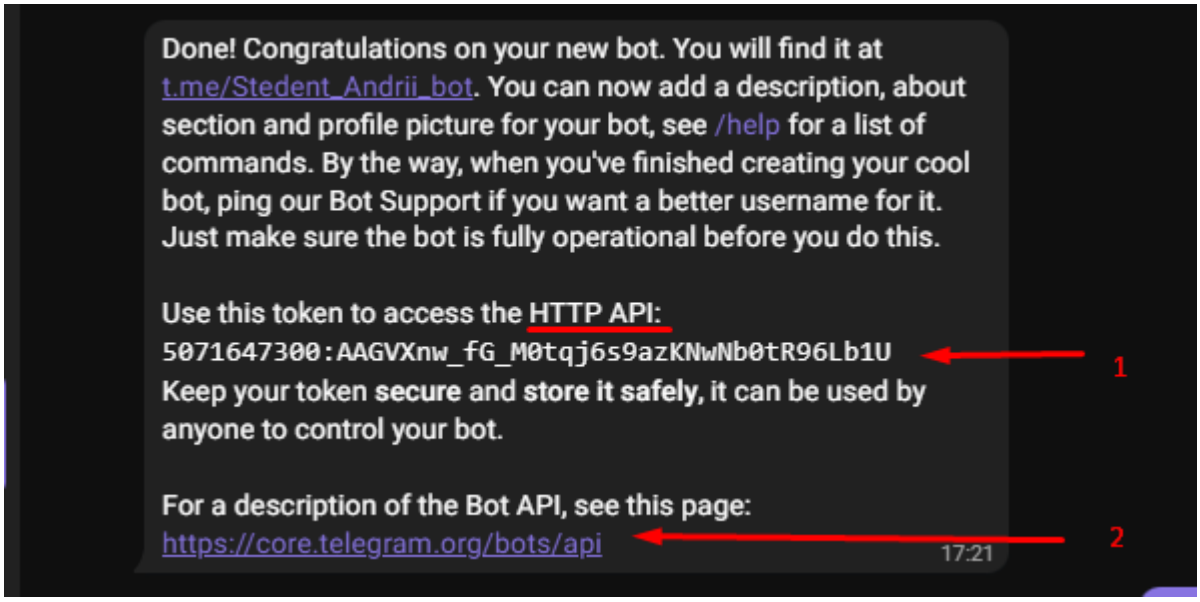


Рис 5. Отримання токену

На зображенні під номером 5 видно токен (TOKEN) що буде далі використання для керування, а під номером 2 видно посилання на бота для доступу в мережі інтернет.

3.2 PyCharm як середовище розробки

PyCharm це додатком що можна використовувати на одній з трьох платформ: платформами Linux, macOS та Windows. Це один з найкращих інтегрованих середовищ розробки на мові Python, PyCharm забезпечує підтримку версій Python 2 (2.7) і Python 3 (3.5 і вище).

PyCharm поставляється з безліччю модулів, пакетів та інструментів для прискорення розробки Python. Крім того, PyCharm може бути налаштований відповідно до вимог розробників та особистих уподобань має широкий функціонал. Вперше його було представлено публіці у лютому 2010 року. Крім аналізу коду, PyCharm пропонує такі можливості:

- графічний відладчик

- інтегрований модульний тестер
- Інтеграційна підтримка систем контролю версій (VCS)
- Підтримка науки про дані за допомогою Anaconda

Провівши аналіз інтегрованих середовищ розробки вибір було зроблено саме на користь PyCharm бо він надає можливість слідувати швидким етапам розробки. Інтелектуальний редактор коду PyCharm поставляється з розумним редактором коду, який полегшує написання високоякісного коду Python. Він пропонує покращений рівень розуміння та читабельності коду за допомогою різних колірних схем для ключових слів, класів і функцій, тобто синтаксису та підсвічування помилок.

На додаток до можливості інтелектуального завершення коду, редактор коду генерує інструкції для завершення поточного коду. Визначення помилок і проблем набагато зручніше, а також інтеграція лінтера та швидкі виправлення.

Наявність інструментів інтеграції - PyCharm надає підтримку для інтеграції низки інструментів. Ці інструменти відрізняються від допомоги у підвищенні продуктивності коду до полегшення роботи з проектами науки про дані. Деякі з найважливіших інструментів інтеграції, доступні для PyCharm, включають:

Дані та машинне навчання - PyCharm поставляється з підтримкою наукових бібліотек, таких як Matplotlib і SciPy, щоб допомогти розробникам Python виконувати проекти з науки про дані та машинного навчання. Інтегроване налагодження та тестування - IDE поставляється з підтримкою програм налагодження та тестування. Щоб досягти того ж, PyCharm має вбудований налагоджувач Python та інтегроване модульне тестування з порядковим покриттям коду. Можлива розробка кількох технологій та редагування в реальному часі також доступне в PyCharm, тобто розробники можуть створювати/змінювати веб-сторінку, одночасно запускаючи її. Таким чином, зміни можна відстежувати безпосередньо у веб-браузері.

3.3 Алгоритмічне відтворення Telegram боту

Імпортування бібліотек для подальшої роботи. Основною бібліотекою виступає telebot, вона надає доступ до всіх функцій та надає змогу працювати з токенами та звертатися до денних

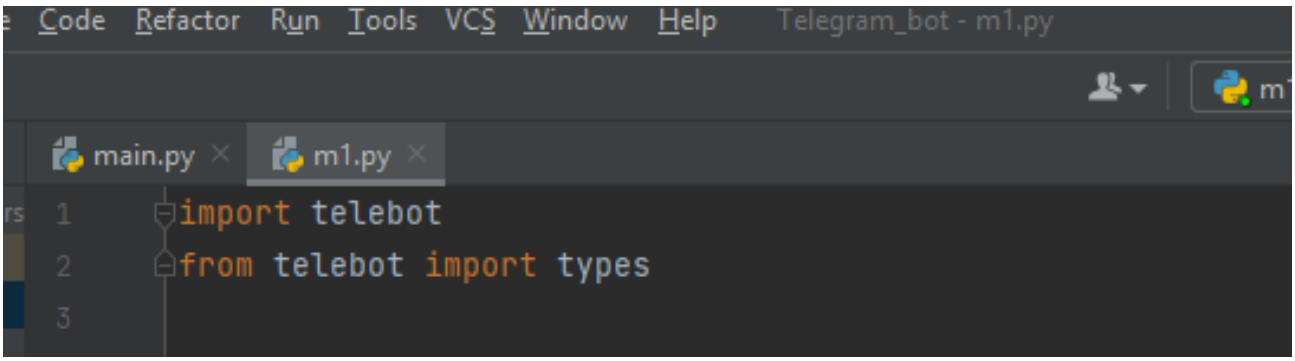


Рис. 6 Імпорт бібліотек

Наступним кроком треба використати токен (TOKEN) для зручності роботи в коді і подальшого використання записуються данні токену в змінну за аналогічною назвою TOKEN. Другий рядок створює об'єкт за допомогою якого можливо керувати ботом і саме в цей об'єкт буде використовувати раніше створений токен.

```
TOKEN = '1074511338:AAFvLcVeva0zAia8PqnIVZ_WYHnomJV9HxE'

bot = telebot.TeleBot(TOKEN)  #t.me/SavAvaS_test_bot
```

Рис 7 Доступ до токену

Якщо в розвивати бот далі то ця функція, а саме токен може буде змінена, що в совою чергу робить універсальним написаний код даного програмного продукту, таким чином ми робимо наш продукт універсальним.

3.4 Організація методу першого контакту

Далі треба оголосити обробники наших повідомлень. Обробники повідомлень оголошують фільтри через які повинні пройти повідомлення і якщо повідомлення проходить даний фільтр тоді функція декорації визвався і вхідне повідомлення передається як аргумент.

Нижче ми бачимо функцію Lamba і якщо вона повертає значення true, то повідомлення буде оброблятися функцією декорації.

Тепер в нас додані два методи і якщо користувачі введуть «/help» бот направить його до менеджера з посиланням на чат з ним. Далі починаємо на рівні наступної функції обробку повідомлень від користувача за допомогою розгалужень if та elif.

```
13
14  @bot.message_handler(commands=['start', 'help'])
15  def send_welcome(message):
16      bot.reply_to(message, "Howdy, how are you doing?")
17
18  @bot.message_handler(func=lambda m: True)
19  def echo_all(message):
20      if message.text == 'Привет':
21          bot.reply_to(message, ': Hi student!')
22      elif message.text == 'hi':
23          bot.reply_to(message, 'Hi again! The bot Help you')
24      elif message.text == '/reg':
25          bot.send_message(message.from_user.id, "Tell me abot you what your name?")
26          bot.register_next_step_handler(message, reg_name)
```

Рис8 Обробка перших повідомлень

Повідомлення спочатку потрапляють в системну чергову повідомлень операційної системи. З неї повідомлення передаються додатками, якими вони призначені, і записуються в чергу доданих. Кожне додаток має власну чергу повідомлення. Додаток у циклі, який називається циклом обробки повідомлень, отримує повідомлення з чергових додатків і керує їх відповідними функціями

вікна, яка і виконує роботу повідомлень. Цикл обробки повідомляється зазвичай з оператора while в якому циклічно викликалися функції GetMessage

Повідомлення, які можуть оброблятися додатком, побудованим на основі бібліотеки класів MFC, розділяються в залежності від процедур їх обробки на трьох великих групах. Групи повідомлень:

- віконні повідомлення
- повідомлення від органів управління
- команди

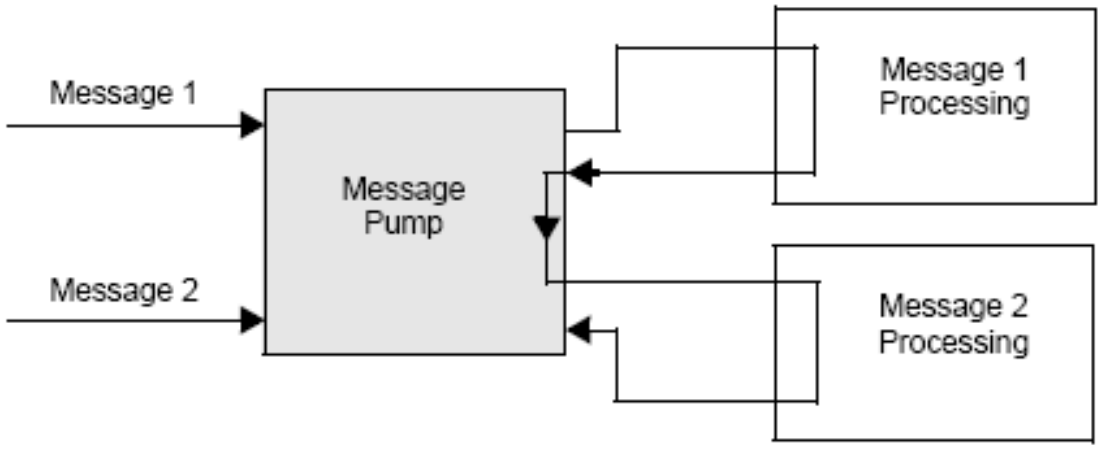


Рис 9 Схема обробки повідомлень

3.5 Функції збору даних

В цьому розділі починається створення функцій, у коді вони відображені як def, що будуть відповідати за збирання та зберігання інформації про користувача для подальшої обробки. Дані функції будуть визнані якщо користувач написав в чат /reg. Нижчее ви бачите дві функції що зберігають ім'я та прізвище користувача.

Важливо що в середині кожної функції ми оголошуємо що вони будуть глобальні, це потрібно для того що б ми могли їх використовувати в будь якому місці нашого програмного продукту. Далі за допомогою крапки звітуємо яку інформацію вніс користувач і заносимо її в наші глобальні змінні.

```python
def reg_name(message):
    global name
    name = message.text
    bot.send_message(message.from_user.id, "Last Name?")
    bot.register_next_step_handler(message, reg_surname)

def reg_surname(message):
    global surname
    surname = message.text
    bot.send_message(message.from_user.id, "How Old Yout?")
    bot.register_next_step_handler(message, reg_age)
```

Рис10 Функція обробки Name

Наступним кроком ми зберігаємо вік. Це потрібно для можлівого подальшого розвитку та використання бо вік входить в список особистих даних тому в кінці ми запитуємо про згоду на збір даних.

Функція def в даному випадку обробляє message що в подальшому дозволить обробляти будь які повідомлення які передбачені нашим додатком і в той же час дає можливість подальшого розвитку продукту.

```python
def reg_age(message):
    global age
    #age = message.text
    while age == 0:
        try:
            age = int(message.text)
        except Exception:
            bot.send_message(message.from_user.id, "Please fill number!")
```

Рис 11 Обробка віку користувача

Дана функція зберігає глобальне значення, але також має перевірку що саме вніс користувач, якщо введено не цифру, то функція попросить ввести ще раз. Це потрібно для запобігання ситуацій коли користувач ввів вік буквами або хоче передати не достовірні данні. Також розроблено валідацію яка перевіряє що цифра не виходить за діапазон можливого.

3.5 Додавання кнопок та доступу до інформації університету

Для спрощення користуванням боту розроблено кнопки де користувач може впевнитися що вірно вніс данні, а також одразу перейти на сайт університету. За допомогою кнопки KNUTD.

```
keyboard = types.InlineKeyboardMarkup()
key_yes = types.InlineKeyboardButton(text='Yes', callback_data='yes')
keyboard.add(key_yes)
key_no = types.InlineKeyboardButton(text='No', callback_data='no')
keyboard.add(key_no)
key_KNUTD = types.InlineKeyboardButton(text='KNUTD', callback_data='KNUTD')
keyboard.add(key_KNUTD)
question = 'You' + str(age) + ' years old? And your name is: ' + name + ' ' + surna
bot.send_message(message.from_user.id, text = question, reply_markup=keyboard)
```

Рис12 Створення кнопок доступу

Користувач може бачити данні які він вносив та повернутися до введення знов якщо потрібно.

Кнопки у додатку діляться на чотири типи Розклад, Чати, Канали та офіційний сайт Київського національного університету технології та дизайну що можна побачити на рис 13.

```
@bot.callback_query_handler(func=lambda call: True)
def callback_worker(call):
    if call.data == "ROZKLAD":
        bot.send_message(call.message.chat.id, "Розклад університетуhttps://www.knutd.edu.ua/ekts/grafik/")
    elif call.data == "KNUTD":
        bot.send_message(call.message.chat.id, "Достууп к информации о университее https://www.knutd.edu.ua/")
    elif call.data == "Chenels":
        bot.send_message(call.message.chat.id, "Список каналів https://t.me/knutd_stud")

    elif call.data == "Chatsss":
        bot.send_message(call.message.chat.id, "ALL Chats here : ")
        bot.send_message(call.message.chat.id, "LIST OF CHATS : https://t.me/adghkkddggg")
       #bot.register_next_step_handler(call.message, reg_name)
```

Рис 13 Оголошення можливих натискань

Після оголошення кнопок ми починаємо обробку і оформлення кнопок за допомогою знову функції lamda.Графічне представлення кнопок для користувача видно на Рис 14:

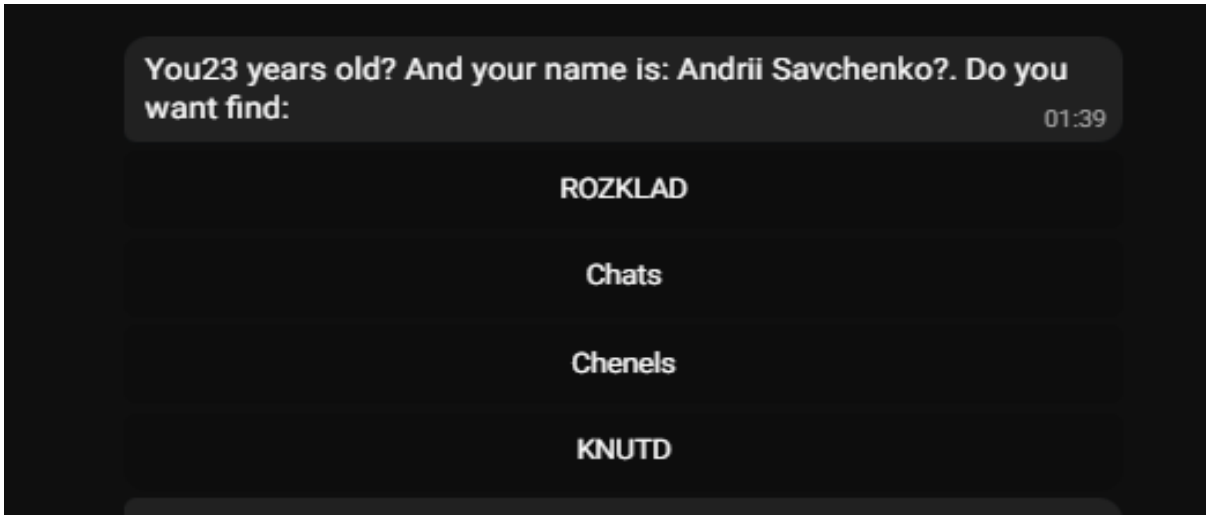


Рис 14 Кнопки доступу

Відповідь боту залежить від вибраної кнопки та надасть конкретну інформацію користувачу. Додаток розроблено так що запити можна відправити одразу по чотирьох направленнях. Це дозволяє багато поточність оброблення процесів що входить в основні інструменти телеграм api. То наш додаток починає обробляти кілька запитів одночасно.

Таким способом найдеться увесь доступ в одному місці до користувацької інформації що впливає на процес адаптації в університеті

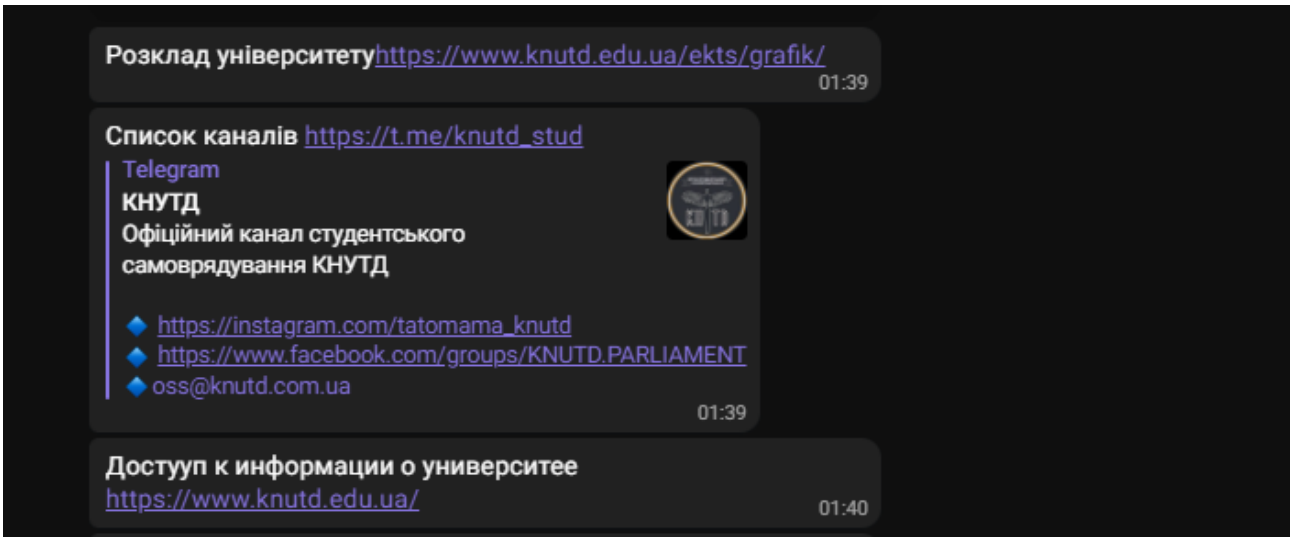


Рис 15 Отримання даних

Висновки розділу 3

В результаті проведеного дослідження було розроблено додаток телеграм бот для довідкової системи комунікації студентів на базі Python бібліотеки Telebot з використанням відкритого API Telegram. Розглянуто основний функціонал додатку: доступ до всіх чатів комунікації студентів, відображення важливої інформації, надання важливих посилань на сайт університету. Показано способи доступу та реалізації програмного коду в середовищі розробки.

Розглянуто інтегровані середовищ розробки а саме PyCharm та наведено приклад функціоналу бібліотеки Telebot за допомогою якої створено бот. Досліджено можливості платформи телеграм та головні команди для керування ботом

Висновки

Досліджено основні види методологій розробки програмних продуктів та розглянуто чинники які впливають на вибір цих методологій. Проведено аналіз відкритих джерел даних стосовно цих тем.

Наведено основні етапи життєвого циклу програмних продуктів та розроблено алгоритм розробки за технологією SDLC. Проведено огляд розробки візуальної алгоритмічної структури.

Розглянуто основний функціонал створення телеграм боту та детально розписані команди для керування даним ботом.

Було розроблено програмний продукт у вигляді Telegram боту для довідкової системи комунікації між студентами в зручному форматі та з доступним інтерфейсом. Надано скріншоти роботи боту та коротко описаний головний функціонал. Наведені приклади використання та прораховані різні сценарії. Розглянуто такий функціонал: збір інформації про користувача; надання інформації про користувача; отримання публічних даних про університет; доступ до вузлів комунікації студентів у вигляді чатів; доступ до соціально важливих активностей університету.

Для розробки програмного продукту було використано об'єктно орієнтовану мову програмування Python; в якості платформи розробки (IDE – інтеграційна платформа розробки) було використано PyCharm. В ході розробки були задіяні також відкриті джерела документації, а також використані принципи побудови додатків за технологією Об'єктно орієнтованого програмування: глобальні змінні, наслідування та інкапсуляція.
В ході виконання роботи було виконано такі завдання:

- Зроблено висновки щодо необхідності розробки програмного продукту, для вирішення проблеми комунікації що поєднує виявлені переваги та позбавлено зазначених недоліків;
- Розроблено систему на основі Телеграму за допомогою мови програмування Python;

Літературні джерела

1. Telegram [Електронний ресурс]. — Режим доступу: https://ru.wikipedia.org/wiki/Telegram.
2. . Bot Code Examples — Telegram APIs [Електронний ресурс]. — Режим доступу: https://core.Telegram.org/bots/samples (дата звернення: 01.10.2021).
3. Документация Telegram: Примеры ботов [Електронний ресурс]. — Режим доступа: https://tlgrm.ru/docs/bots/samples#c-sharp (дата обращения: 11.03.2018).
4. Microsoft [Інтернет ресурс]: https://www.microsoft.com/
5. Алгоритм [Електронний ресурс] – URL: http://www.machinelearning.ru/wiki/index.php?title=Алгоритм (дата обращения: 13.05.2020)
6. Самоучитель PYTHON [Електронний ресурс]. – URL: http://pythoshka.ru/p1138.html (Дата обращения: 20.05.2020).
7. Документація C#. [Інтернет ресурс]: https://docs.microsoft.com/
8. Исключения в python. Конструкция try - except для обработки исключений [Електронний ресурс]. – URL: https://pythonworld.ru/tipydannyx-v-python/isklyucheniya-v-python-konstrukciya-try-except-dlyaobrabotki-isklyuchenij.html.
9. Чим відрізняються чат-боти в Telegram, WhatsApp, Viber, Facebook, VK [Електронний ресурс]. – URL: https://www.rpv-bot.ru/chem-otlichaetsyachat-bot-v-telegram-whatsapp-vk-viber-facebook (Дата обращения: 19.05.2020).
10. Хабр, навчальний процес у IT [Електронній ресурс]. – URL: https://habr.com/ru/company/skillbox/blog/443412/ (дата звернення: 01.10.2021).
11. Что такое программное обеспечение [Електронний ресурс]. – URL: http://procomputer.su/comp-gramotnost/79-chto-takoe-programmnoeobespechennie (Дата обращения: 17.05.2020).

12. Мова програмуваня Python [Електронний ресурс]. – URL: https://web-creator.ru/articles/python (Дата обращения: 17.05.2020).
13. Чому саме месенджери [Електронний ресурс]. – URL: https://vc.ru/marketing/51778-kak-ispolzovat-bot-whatsapp-effektivnyysposob-generacii-kachestvennyh-lidov-cherez-messendzher (дата звернення: 20.11.2021).
14. A Brief History of .NET Core; [Електронний ресурс]: https://dotnetcore.show/episode-1-a-brief-history-of-net-core/
15. Telegram Bot Api; [Електронний ресурс]: https://core.telegram.org/bots/api
16. Telegram Bots; [Електронний ресурс]: https://core.telegram.org/bots
17. Telegram APIs; ; [Електронний ресурс]: https://core.telegram.org/
18. Telegram Bot Code Examples; [Інтернет ресурс]: https://core.telegram.org/bots/samples
19. Agile [Електронний ресурс] – Режим доступу до ресурсу: https://agile.yakubovsky.com/ua/2015/10/9-osnovnykh-metodolohiy-rozrobky-prohramnoho-zabezpechennya-agile/.
20. TelegramBotSchema;[Електронний ресурс]: https://core.telegram.org/schema
21. Telegram Bot Apps; [Електронний ресурс]: https://telegram.org/apps
22. Telegram Bot Instant View; [Електронний ресурс]: https://instantview.telegram.org/
23. Тестування телеграм-бота[Електронний ресурс]. – Режим доступу: https://habr.com/ru/post/322816/
24. REST API[Електронний ресурс]. – Режим доступу: https://habr.com/ru/post/351890/
25. Посібник з роботи з HTTP у Python. Бібліотека requests [Електронний ресурс]. – Режим доступу: https://khashtamov.com/ru/pythonrequests/ HTML Text Formatting; [Електронний ресурс https://www.w3schools.com/html/html_formatting.asp
26. JSON Format; [Електронний ресурс]: https://www.json.org/json-ru.html

27. Телеграм бот за допомогою TeleBot [Електронний ресурс]. – Режим доступу: https://habr.com/ru/post/448310/
28. Офіційний сайт Telegtam [Електронний ресурс]: API – Режим доступу: https://core.telegram.org/api.
29. Козлов А. А., Телеграм-бот як простий та зручний спосіб отримання інформації [Електронний ресурс] / А. А. Козлов, А. В. Батищев // Территория науки. – 2017. – №5. – с. 50-60. – Режим доступу: https://cyberleninka.ru/article/v/telegram-bot-kak-prostoy-i-udobnyy-sposobpolucheniya-informatsii
30. Sikore M. Dart Essentials // Martin Sikora, 2015. – 234 с.
31. Strom C. Dart for Hipsters // Chris Strom, 2014. – 144 с.
32. Buckett C. Dart in Action // Chris Buckett, 2013. – 424 с.
33. Mitchell D. Dart: Scalable Application Development // D. Mitchell, S. Akopkokhyants, I. Balbaert, 2017. – 1298 с.

Додаток А

Код программы

```
import telebot
from telebot import types

#926615086:AAHUl8OG-xpenaOLNwj70O3BCXlqPULCAls

name = ''
surname = ''
age = 0

bot = telebot.TeleBot("926615086:AAHUl8OG-xpenaOLNwj70O3BCXlqPULCAls")

@bot.message_handler(commands=['start', 'help'])
def send_welcome(message):
        bot.reply_to(message, "Howdy, how are you doing?")



@bot.message_handler(func=lambda m: True)
def echo_all(message):
    if message.text == 'Привет':
        bot.reply_to(message, 'Привет создатель бота!')
    elif message.text == 'hi':
        bot.reply_to(message, 'Hi again! The bot creator!')
    elif message.text == '/reg':
        bot.send_message(message.from_user.id, "Привет! Давай познакомимся! Как тебя зовут?")
        bot.register_next_step_handler(message, reg_name)
```

```
        #bot.reply_to(message, message.text)


def reg_name(message):
    global name
    name = message.text
    bot.send_message(message.from_user.id, "Какая у вас фамилия?")
    bot.register_next_step_handler(message, reg_surname)


def reg_surname(message):
    global surname
    surname = message.text
    bot.send_message(message.from_user.id, "Сколько вам лет?")
    bot.register_next_step_handler(message, reg_age)


def reg_age(message):
    global age
    #age = message.text
    while age == 0:
        try:
            age = int(message.text)
        except Exception:
            bot.send_message(message.from_user.id, "Вводите цифрами!")
```

```
    keyboard = types.InlineKeyboardMarkup()
    key_yes = types.InlineKeyboardButton(text='Да', callback_data='yes')
    keyboard.add(key_yes)
    key_no = types.InlineKeyboardButton(text='Нет', callback_data='no')
    keyboard.add(key_no)
    question = 'Тебе ' + str(age) + ' лет? И тебя зовут: ' + name + ' ' +
surname + '?'
    bot.send_message(message.from_user.id, text = question,
reply_markup=keyboard)



@bot.callback_query_handler(func=lambda call: True)
def callback_worker(call):
    if call.data == "yes":
        bot.send_message(call.message.chat.id, "Приятно познакомиться!
Теперь запишу в БД!")
    elif call.data == "no":
        bot.send_message(call.message.chat.id, "Попробуем еще раз!")
        bot.send_message(call.message.chat.id, "Привет! Давай
познакомимся! Как тебя зовут?")
        bot.register_next_step_handler(call.message, reg_name)
def start(update, context):
    first_name = update.message.chat.first_name
    update.message.reply_text(f"Hi {first_name}, nice to meet you!")
    start_getting_birthday_info(update, context)
def start_getting_birthday_info(update, context):
    global STATE
    STATE = BIRTH
    update.message.reply_text(
        f"I would need to know your birthday, so tell me what year did you
born in...")
```

```
def on_start(update, context):
    chat = update.effective_chat
    context.bot.send_message(chat_id=chat.id, text="Привет, я Валютный бот")



def on_message(update, context):
    chat = update.effective_chat
    text = update.message.text
    try:
        number = float(text)
        rate = 80.34
        soms = number * rate
        message = "$%.2f = %.2f сом" % (number, soms)
        context.bot.send_message(chat_id=chat.id, text=message)
    except:
        context.bot.send_message(chat_id=chat.id, text="Напишите число для перевода")

token = "802414251:AAGGRWasfg8UmXVJPk79zNdjBQG3w33mvGE"

updater = Updater(token, use_context=True)

dispatcher = updater.dispatcher
dispatcher.add_handler(CommandHandler("start", on_start))
dispatcher.add_handler(MessageHandler(Filters.all, on_message))

updater.start_polling()
updater.idle()
def received_birth_year(update, context):
```

```
    global STATE
    try:
        today = datetime.date.today()
        year = int(update.message.text)
        if year > today.year:
            raise ValueError("invalid value")
        context.user_data['birth_year'] = year
        update.message.reply_text(
            f"ok, now I need to know the month (in numerical form)...")
        STATE = BIRTH_MONTH
    except:
        update.message.reply_text(
            "it's funny but it doesn't seem to be correct...")


@bot.message_handler(commands=['start'])
def start_message(message):
    bot.send_message(message.chat.id, 'Пришли мне Emoji, который необходимо сделать размытым.)


@bot.message_handler(content_types=['text'])
def send_text(message):
    if message.text.lower() == 'нам первый смайлик':
        img = open('Смайлики и люди 1.png', 'rb')


        bot.send_document(message.chat.id, img)
    elif message.text.lower() == 'наш второй смайлик':
        img = open('Смайлики и люди 2.png', 'rb')
        bot.send_document(message.chat.id, img)


    else:
```

```
        bot.send_message(message.chat.id, 'Прости, но пока у меня нет этих
Emoji')
def get_requirements(raw=False):
    """Build the requirements list for this project"""
    requirements_list = []

    with open('requirements.txt') as reqs:
        for install in reqs:
            if install.startswith('# only telegram.ext:'):
                if raw:
                    break
                continue
            requirements_list.append(install.strip())

    return requirements_list



def get_packages_requirements(raw=False):
    """Build the package & requirements list for this project"""
    reqs = get_requirements(raw=raw)

    exclude = ['tests*']
    if raw:
        exclude.append('telegram.ext*')

    packs = find_packages(exclude=exclude)
    # Allow for a package install to not use the vendored urllib3
    if UPSTREAM_URLLIB3_FLAG in sys.argv:
        sys.argv.remove(UPSTREAM_URLLIB3_FLAG)
        reqs.append('urllib3 >= 1.19.1')
```

```
        packs = [x for x in packs if not
x.startswith('telegram.vendor.ptb_urllib3')]

    return packs, reqs



def get_setup_kwargs(raw=False):
    """Builds a dictionary of kwargs for the setup function"""
    packages, requirements = get_packages_requirements(raw=raw)

    raw_ext = "-raw" if raw else ""
    readme = f'README{"_RAW" if raw else ""}.rst'

    fn = os.path.join('telegram', 'version.py')
    with open(fn) as fh:
        for line in fh.readlines():
            if line.startswith('__version__'):
                exec(line)
def get_requirements(raw=False):
    """Build the requirements list for this project"""
    requirements_list = []

    with open('requirements.txt') as reqs:
        for install in reqs:
            if install.startswith('# only telegram.ext:'):
                if raw:
                    break
                continue
            requirements_list.append(install.strip())
```

```
    return requirements_list


def get_packages_requirements(raw=False):
    """Build the package & requirements list for this project"""
    reqs = get_requirements(raw=raw)

    exclude = ['tests*']
    if raw:
        exclude.append('telegram.ext*')

    packs = find_packages(exclude=exclude)
    # Allow for a package install to not use the vendored urllib3
    if UPSTREAM_URLLIB3_FLAG in sys.argv:
        sys.argv.remove(UPSTREAM_URLLIB3_FLAG)
        reqs.append('urllib3 >= 1.19.1')
        packs = [x for x in packs if not
x.startswith('telegram.vendor.ptb_urllib3')]

    return packs, reqs



def get_setup_kwargs(raw=False):
    """Builds a dictionary of kwargs for the setup function"""
    packages, requirements = get_packages_requirements(raw=raw)

    raw_ext = "-raw" if raw else ""
    readme = f'README{"_RAW" if raw else ""}.rst'

    fn = os.path.join('telegram', 'version.py')
    with open(fn) as fh:
```

```
        for line in fh.readlines():
            if line.startswith('__version__'):
                exec(line)
bot.polling()
```

ДОДАТОК Б

Копії публікацій за темою магістерської роботи

Міністерство освіти і науки України

Київський національний університет технологій та дизайну

МЕХАТРОННІ СИСТЕМИ: ІННОВАЦІЇ ТА ІНЖИНІРИНГ

ТЕЗИ ДОПОВІДЕЙ

V МІЖНАРОДНОЇ НАУКОВО-ПРАКТИЧНОЇ КОНФЕРЕНЦІЇ

4 листопада 2021

MSIE 2021

Київ 2021

УДК 519.246.8(075.8)

РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ СТВОРЕННЯ СИСТЕМИ КОМУНІКАЦІЇ СТУДЕНТІВ УНІВЕРСИТЕТУ НА БАЗІ TELEGRAM

А.С. Савченко, магістрант
Київський національний університет технологій та дизайну
Т.І. Демківська, кандидат технічних наук, доцент
Київський національний університет технологій та дизайну

Ключові слова: кросплатформенність, програмне забезпечення, месенджер, комунікатор.

Метою даного дослідження є розробка програмного забезпечення та вибір найкращої платформи для використання даного програмного продукту, підвищення ефективності навчального процесу шляхом спрощення організації взаємодії студентів між собою та забезпечення більш зручного доступу до інформаційних матеріалів

Дослідження також охоплює аналіз проблеми налагодження комунікації студентів варіанти її вирішення. За допомогою опитування серед студентів необхідно визначити їхні потреби і проблеми: дослідження рівня інформованості про діяльність вищого навчального закладу і способи донесення важливої інформації про події.

Завдяки стрімкому зростанню популярності платформ спілкування під назвою месенджери стала можливою розробка програм на базі платформ що вже існують.

Програмування на початку свого існування не могло виконувати складні задачі. На сьогодні програмування проникло майже в усі сфери нашого життя і допомагає нам зробити його легшим.

Рисунок 1 - Схема комунікації студентів

Студенти є головними користувачами нашого програмного продукту. Для розуміння вибору платформи було проведено опитування

САВЧЕНКО А.С., ДЕМКІВСЬКА Т.І.

РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ СТВОРЕННЯ СИСТЕМИ КОМУНІКАЦІЇ СТУДЕНТІВ УНІВЕРСИТЕТУ НА БАЗІ TELEGRAM

SAVCHENKOA.S, DEMKIVSKAT.I.

DEVELOPMENT OF SOFTWARE FOR THE CREATION OF A COMMUNICATION SYSTEM FOR UNIVERSITY STUDENTS ON THE BASIS OF TELEGRAM

Relations and interaction of students among themselves as a form of pragmatic kerned _social communication contributes to the development and increase the efficiency of functioning of higher education institutions. In today's world and the society processes that are taking place confirm the need to work on the establishment of communication at different levels and in organizations of different professions, including in the institutions of higher vocational education. Creation of a favorable communicative environment increases the assessment of activities and enhances the reputation of higher education institution in the eyes of the public. Relations with the community as a form of pragmatic keyed social communication contributes to the development and improvement of the efficiency of functioning of higher education institutions

Вступ

Зв'язки і спілкування студентів між собою як форма прагматичної керованої соціальної комунікації сприяє розвитку і підвищенню ефективності функціонування інститутів вищої освіти. В сучасному світі і суспільстві процеси, що відбуваються підтверджують необхідність роботи над налагодженням комунікації з громадськістю на різних рівнях і в організаціях різного профілю, в тому числі і в установах вищої профільної освіти. Створення сприятливого комунікаційного середовища підвищує оцінку діяльності і зміцнює репутацію вищого навчального закладу в очах громадськості. Зв'язки з громадськістю, як форма прагматичної керованої соціальної комунікації, сприяє розвитку і підвищенню ефективності функціонування інститутів вищої освіти.

Постановка завдання

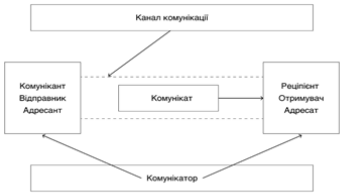
Метою даного дослідження є розробка програмного забезпечення та вибір найкращої платформи для використання даного програмного продукту. Підвищення ефективності навчального процесу, шляхом спрощення організації взаємодії студентів між собою та забезпечення більш зручного доступу до інформаційних матеріалів

Дослідження також охоплює аналіз проблеми налагодження комунікації студентів варіанти її вирішення. За допомогою опитування серед студентів необхідно визначити їхні потреби і проблеми: дослідження рівня інформованості про діяльність вищого навчального закладу і способи донесення важливої інформації про події.

Завдяки стрімкому зростанню популярності платформ спілкування під назвою месенджери стала можливою розробка програм на базі платформ що вже існують.

Основна частина

Програмування на початку свого існування не могло виконувати складні задачі. На сьогодні програмування проникло майже в усі сфери нашого життя і допомагає нам зробити його легшим.Компанії та підприємства впроваджують нові заходи для оптимізації робочого процесу.За допомогою IT-технологій також розвинулась соціальна складова в суспільстві. У сучасному світі для продуктивного навчання та розвитку найголовнішим фактором є комунікація. Так би мови програмування, комп'ютери і загалом все навкруги створено за допомогою комунікації людей. Створене мною програмне забезпечення розв'язує проблеми комунікації студентів університету і має на меті створити



екосистему для продуктивного спілкування.

Рис. Схема комунікації студентів

Спілкування та люди нас завжди оточують, розвиток мережі інтернет дав можливість максимально прискорити процес налагодження зв'язку з іншими людьми. І спрощувати комунікацію між суб'єктами різних направлень та професій.

Студенти є головними користувачами нашого програмного продукту. Для розуміння вибору платформи було проведено опитування серед студентів. За допомогою дослідження було визначено, що студенти найбільше використовують два месенджери для комунікації: Viber та

Міністерство освіти і науки України

Київський національний університет технологій та дизайну

МЕХАТРОННІ СИСТЕМИ: ІННОВАЦІЇ ТА ІНЖИНІРИНГ

ТЕЗИ ДОПОВІДЕЙ V МІЖНАРОДНОЇ НАУКОВО-ПРАКТИЧНОЇ КОНФЕРЕНЦІЇ

4 листопада 2021

Рекомендовано Вченою радою
факультету мехатроніки та комп'ютерних технологій
Київського національного університету технологій та дизайну

КИЇВ 2021

серед студентів. За допомогою дослідження було визначено, що студенти найбільше використовують два месенджери для комунікації: Viber та Telegram.

Проблематика питання в тому, як саме пов'язати всіх студентів так, щоб їм було це корисно і як зробити впровадження цієї системи комфортним та не складним. Провівши дослідження, я можу зробити висновок, що у більшості студентів є мобільні пристрої з доступом в мережу інтернет. Дві найпопулярніші операційні системи на телефонах це Android та iOS. Враховуючи ці дані, платформа повинна бути доступна на двох програмних забезпеченнях.

Також важливою перевагою створюваної платформи є те, що буде можливість зменшити витрати часу та ресурсів на розробку окремої платформи. Важливим фактором є кросплатформеність, а для найлегшого впровадження цього способу комунікацій бажано, щоб програмне забезпечення вже було встановлено на телефоні.

З результатів проведеного дослідження зроблено висновок про те, що можливості цієї платформи повинні включати доступність не тільки на мобільних пристроях, а і на персональних комп'ютерах.Тож за допомогою бота реалізується автоматизація взаємодії між користувачем та мессенджером.Важливим фактором є доступ до інформації в боті цілодобово та незалежно від місця знаходження користувача, що є особливо важливим під час навчання дистанційно.

Проведено дослідження та порівняно платформи, на яких можлива розробка програмного забезпечення, а також проаналізовано вимоги до розробки програмного забезпечення. В результаті вибір було зроблено на користь платформи Telegram.

Проведено аналіз переваг та недоліків Telegram, а також здійснено порівняльну характеристику відносно відомих аналогів.

Розроблено програмне забезпечення, на прикладі якого було наочно досліджено особливості роботи платформи, механізми розробки боту для платформ Android та iOS на основі єдиної системи телеграм.

Визначено що патерн користування телеграм боту надзвичайно простий та знайомий кожному студенту.Зроблено висновок що телеграм найкраща платформа для нашого програмного продукту. Також використано вбудований функціонал TelegramBotApi .

Список використаних джерел

1. V.V. Rizun.— "MASS COMMUNICATION THEORY" (2018). — 260 pages

2. Staticprogramanalysis [Електронний ресурс] // wikipedia.org. – 2004. – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Static_program_analysis.

3. Dynamicprogramanalysis [Електронний ресурс] // wikipedia.org. – 2009. – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Dynamic_program_analysis.

Telegram. Але важливою частиною є з ким і в якому месенджері вони спілкувались, задля того, щоб наш продукт потрапив саме під вирішення потреби користувачів. В більшої частини студентів комунікація з друзями та тими, з ким вони навчаються в університеті проходить саме в Telegram. Viber більше сприймається як месенджер для спілкування з родичами. Тож переді мною стояла задача вибрати одну платформу, враховуючи вищезазначені ввідні дані та вимоги до платформи.

Проблематика питання в тому, як саме пов'язати всіх студентів так, щоб їм було це корисно і як зробити впровадження цієї системи комфортним та не складним. Провівши дослідження, я можу зробити висновок, що у більшості студентів є мобільні пристрої з доступом в мережу інтернет. Дві найпопулярніші операційні системи на телефонах це Android та iOS. Враховуючи ці дані, платформа повинна бути доступна на двох програмних забезпеченнях.

Також важливою перевагою платформи є те, що буде можливість зменшити витрати часу та ресурсів на розробку окремої платформи. Важливим фактором є кросплатформеність, а для найлегшого впровадження цього способу комунікацій бажано, щоб програмне забезпечення вже було встановлено на телефоні.

З результатів проведеного дослідження зроблено висновок про те, що можливості цієї платформи повинні включати доступність не тільки на мобільних пристроях, а і на персональних комп'ютерах.Тож за допомогою бота реалізується автоматизація взаємодії між користувачем та мессенджером.Важливим фактором є доступ до інформації в боті цілодобово та незалежно від місця знаходження користувача, що є особливо важливим під час навчання дистанційно.

Висновки

Проведено дослідження та порівняно платформи, на яких можлива розробка програмного забезпечення, а також проаналізовано вимоги до розробки програмного забезпечення. В результаті вибір було зроблено на користь платформи Telegram.

Проведено аналіз перевагта недоліківTelegram, а також здійснено порівняльну характеристикувідносно відомих аналогів.

Розроблено програмне забезпечення, на прикладі якого було наочно досліджено особливості роботи платформи, механізми розробки боту для платформ Android та iOS на основі єдиної системи телеграм.

Визначено що патерн користування телеграм боту надзвичайно простий та знайомий кожному студенту.Зроблено висновок що телеграм найкраща платформа для нашого програмного продукту. Також використано вбудований функціонал TelegramBotApi .

Література

Інформаційні технології в науці, виробництві та підприємництві
Київський національний університет технологій та дизайну

1. V.V. Rizun.— "MASS COMMUNICATION THEORY" (2018). — 260 pages
2. Staticprogramanalysis [Електронний ресурс] // wikipedia.org. – 2004. – Режим доступу до ресурсу:
https://en.wikipedia.org/wiki/Static_program_analysis.
3. Dynamicprogramanalysis [Електронний ресурс] // wikipedia.org. – 2009. – Режим доступу до ресурсу:
https://en.wikipedia.org/wiki/Dynamic_program_analysis.

Додаток В

проєкт

Розробка програмного забезпечення для створення довідкової системи комунікації студентів університету на базі Telegram Bot Api»

Савченко Андрій

Мета

Розробка програмного засобу комунікації з використанням API готового продукту Telegram.
А також аналіз метадологій та розрбка за технолоіею SDLC

Враховуючи змінюючи характер роботи, процес, який різко прискорився під час пандемії Covid-19, ВНЗ відповідають за оснащення майбутніх співробітників навичками, необхідними для підтримання комунікації у віртуальних,

РОЗДІЛ 1, Постановка задачі та основні принципи роботи методологій розробки;
РОЗДІЛ 2, Алгоритмічне забезпечення життєвого циклу програми;
РОЗДІЛ 3, Програмне забезпечення;

Актуальність

Проєкт є актуальним . Тема розкриває чому тепер дуже важливі саме методи розробки програмного продукту.

Будуть вирішені такі проблеми :

- Аналіз методологій;
- Заглиблене вивчення SDLC;
- Налагоджена комунікація, та спрощення доступу до інформації;
- Залучення на старті.

Які проблеми вирішує?

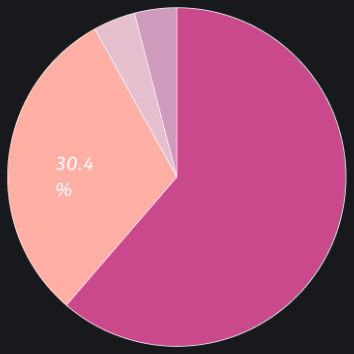
Дослідження

Потенційною аудиторією боту, що взяла участь в опитуванні є люди, що підпадають під такі критерії:

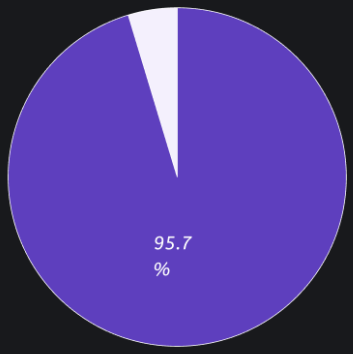
- користуються телеграмом кожен день;
- Молоді студенти преших курсів;

Привіт, підкжи будласка а де чат нашої группи і де знайти розклад онлайн занять ?

Дослідження



60,9% сучасних команій використовуе SCRUM.
30,4% Змішані види методологій
9 % Dev Ops

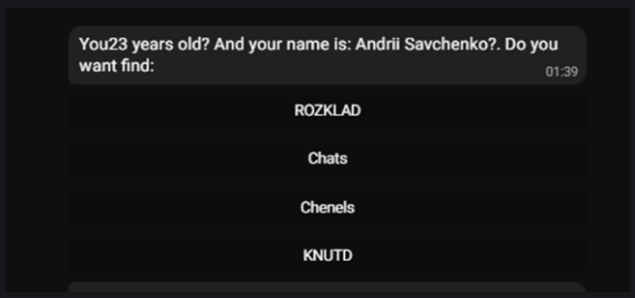


95.7% Згодні що для керування компанією потрібно розуміння SDLC

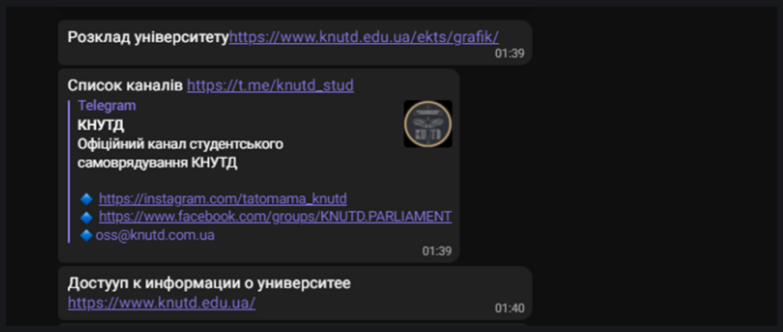
User flow

Прототип

MVP продукту :



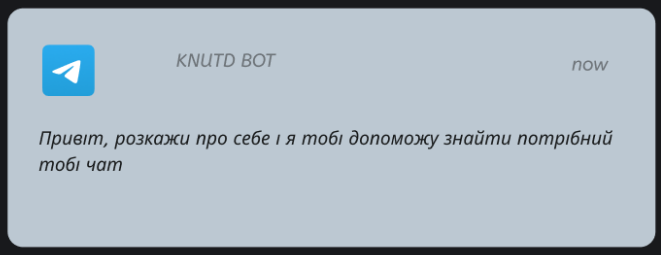
В залежності що вибирає користувач

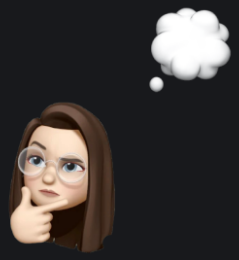


Бот надасть інформацію

Відтворення

SDLC - і зберігання часу для відтворення боту





95.7% Згодні що для керування компанією потрібно розуміння SDLC