МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Харківський національний університет імені В.Н.Каразіна
Факультет математики і інформатики
Кафедра теоретичної та прикладної інформатики

Кваліфікаційна робота

бакалавр

на тему “Розробка сервісу сповіщень систем контролю версій на основі Telegram бота”

Виконав: студент 4 курсу, групи МФ-42
спеціальність 122 «Комп’ютерні науки»
освітньо-професійна програма
«Інформатика»

Гридін Д. Є
_____
(прізвище та ініціали)

Керівник ст.викладач Бережна Н.І.
(прізвище та ініціали)

Рецензент
(прізвище та ініціали)

Харків – 2023 року

1. ВСТУП

1.1 Мета та задачі роботи

У сучасному світі розробка програмного забезпечення є надзвичайно актуальною і важливою галуззю. Контроль версій коду є необхідною складовою процесу розробки програмного забезпечення, який дозволяє команді розробників ефективно співпрацювати та відстежувати зміни у кодовій базі.

Однак, існуючі системи контролю версій, такі як Git або Subversion, часто не надають зручних та швидких способів сповіщення розробникам про події, що відбуваються в репозиторії. Це може призводити до затримок у виявленні проблем, конфліктів та некоректних змін у коді.

Розробка сервісу сповіщень на основі Telegram бота має великий потенціал для поліпшення процесу комунікації та сповіщення розробників у системах контролю версій. У сучасному світі, де швидкість розробки програмного забезпечення є ключовим фактором успіху, ефективна передача інформації про зміни в коді стає надзвичайно важливою. Звичайні електронні листи та повідомлення в системах контролю версій не завжди забезпечують достатню швидкість та зручність сповіщень, що призводить до можливих пропусків важливої інформації та затримок у виявленні проблем.

Одним із варіантів розв'язання цих проблем є використання месенджерів, які широко застосовуються у нашому повсякденному житті. Зокрема, Telegram є одним з найпопулярніших месенджерів, який надає розширений функціонал та можливості для розробки ботів. Завдяки цим можливостям, розробка сервісу сповіщень на основі Telegram бота стає інноваційним підходом, який поєднує сучасні технології з системами контролю версій.

Дипломна робота має на меті розробку такого сервісу сповіщень, який буде інтегруватися з існуючими системами контролю версій і забезпечувати оперативне та зручне сповіщення розробників про зміни в коді. Використання Telegram бота як каналу сповіщень дозволить надіслати повідомлення розробникам безпосередньо на їхні робочі пристрої, що сприятиме швидкій реакції та покращенню комунікації у команді.

1.2 Актуальність роботи

Швидкість розробки програмного забезпечення є ключовим фактором успіху. Контроль версій коду в системах розробки програмного забезпечення є необхідним для ефективної співпраці та відстеження змін у кодовій базі. Однак, існуючі системи контролю версій, можуть бути неефективними у передачі інформації про зміни розробникам.

Це призводить до затримок у виявленні проблем, конфліктів та некоректних змін у коді. Звичайні електронні листи та повідомлення в системах контролю версій не завжди забезпечують достатню швидкість та зручність сповіщень. Це може призводити до пропуску важливої інформації та затримок у виявленні проблем.

В такому контексті розробка сервісу сповіщень на основі Telegram бота є інноваційним підходом, який поєднує сучасні технології з системами контролю версій. Telegram на даний час є одним з найпопулярніших месенджерів з розширеним функціоналом та можливостями для розробки ботів.

Використання Telegram бота як каналу сповіщень надає переваги у швидкості та зручності передачі інформації. Розробники отримують повідомлення безпосередньо на свої робочі пристрої, що сприяє швидкій реакції та поліпшенню комунікації у команді.

Актуальність даної роботи полягає в необхідності розв'язання проблеми ефективного сповіщення розробників про зміни в коді. Розробка сервісу сповіщень на основі Telegram бота може покращити ефективність та якість розробки програмного забезпечення, забезпечуючи оперативне та зручне сповіщення.

Подальше дослідження та реалізація цього проєкту відкривають можливості для покращення комунікації у команді розробників та підвищення продуктивності роботи над проєктами з використанням систем контролю версій. Це створює потенціал для розширення і вдосконалення процесу розробки програмного забезпечення з використанням інноваційних технологій та комунікаційних каналів.

1.3 Методи дослідження

Для досягнення поставленої мети та вирішення поставлених завдань були використані наступні методи дослідження:

- Вивчення та аналіз документації. Був проведений огляд існуючих документів та матеріалів, пов'язаних з системами контролю версій та сповіщеннями розробників. Цей аналіз дозволив оцінити поточний стан досліджуваної проблеми, виявити практично вирішені завдання та ідентифікувати актуальні проблеми, які можуть бути ефективно вирішені за допомогою розробки Telegram бота.

- Проєктування системи сповіщень. Були розроблені вимоги до функціональності та інтерфейсу сервісу сповіщень, визначено архітектуру та компоненти системи. Планувалися імплементація сповіщень з використанням Telegram API, інтеграція з існуючими системами контролю версій, а також розробка зручного інтерфейсу користувача.

- Реалізація та тестування прототипу. Було проведено програмування Telegram бота, розроблено серверну частину, яка взаємодіє з системами контролю версій, та виконано випробування сервісу на тестових репозиторіях.

- Аналіз результатів та висновки. На основі отриманих даних з експерименту було проведено аналіз ефективності та користь використання сервісу сповіщень на основі Telegram бота. Були зроблені висновки щодо досягнення мети дослідження та вирішення поставлених завдань, а також наведені рекомендації для подальшого вдосконалення та розширення сервісу.

Отже, в даній дипломній роботі використано такі методи дослідження: аналіз документації, проєктування системи сповіщень, реалізація та тестування прототипу, та аналіз результатів. Ці методи дозволили досягти мети дослідження та вирішити поставлені завдання.

2. ОСНОВНА ЧАСТИНА

2.1 Постановка задачі

У даній дипломній роботі ставиться завдання розробки сервісу сповіщень для систем контролю версій коду на основі Telegram бота. У цій частині детально будуть описані постановка задачі, її мета та основні вимоги до розроблюваного сервісу.

Мета даної дипломної роботи полягає у розробці ефективного і зручного сервісу сповіщень, який забезпечує користувачів систем контролю версій коду, зручним і ефективним способом отримувати повідомлення про зміни та події, які відбуваються в системі.

Отже, розробка сервісу сповіщень для систем контролю версій коду на основі Telegram бота має на меті створити інтегрований, зручний та ефективний засіб сповіщення користувачів про зміни в коді, котрий базується на платформі Telegram.

Основні задачі, які потрібно вирішити у рамках розробки сервісу сповіщень, включають:

1. Розробка інтерфейсу зв'язку між системою контролю версій і Telegram ботом Тобто налаштування зв'язку з використанням API Telegram та налаштування інтеграції з вибраною системою контролю версій.
2. Розробка механізму відстеження подій в системі контролю версій.
3. Розробка механізму автентифікації та авторизації. Сервіс повинен забезпечувати безпеку і конфіденційність користувачів, переконуючись, що тільки автентифіковані та авторизовані користувачі можуть мати доступ до сервісу сповіщень.

4. Тестування та налагодження сервісу. Після розробки всіх основних компонентів сервісу, необхідно провести тестування його функціональності, а також виконати налагодження для виправлення можливих помилок та недоліків.

В наступних частинах дипломної роботи будуть детально описана реалізація кожної з поставлених задач, а також результати тестування та аналіз отриманих результатів.

2.2 Аналіз предметної галузі

У сучасному програмуванні системи контролю версій коду відіграють ключову роль в управлінні розробкою програмного забезпечення. Системи контролю версій забезпечують збереження, відстеження та управління змінами в програмному коді, дозволяючи розробникам працювати паралельно над одним проєкт, вносити зміни та відновлювати попередні версії коду.

Однак, в сучасному програмуванні дедалі більше розробників використовують месенджери для комунікації та спілкування з командою. Telegram, який є одним з найпопулярніших месенджерів, зарекомендував себе як потужний та зручний інструмент для спілкування, який надає можливості для розробки ботів.

Останні роки також характеризуються активним розвитком інтеграцій між програмними системами та месенджерами. Це дозволяє створювати нові інструменти та сервіси, які полегшують та автоматизують робочий процес розробників. Розробка інтеграції між системою контролю версій коду та Telegram ботом є одним з таких перспективних напрямків.

Існують деякі інструменти, які надають певну підтримку інтеграції між системами контролю версій та месенджерами. Наприклад, деякі CI/CD (Continuous Integration/Continuous Deployment) системи дозволяють надсилати сповіщення про зміни в коді через месенджери. Однак, на сьогодні відсутні повноцінні сервіси сповіщень, розроблені спеціально для систем контролю версій коду на основі Telegram бота.

Огляд сучасного стану справ у даній області свідчить про потребу в розробці такого сервісу. Інтеграція між системою контролю версій та Telegram ботом може значно полегшити спілкування та сповіщення між розробниками,

сприяючи швидкому та ефективному обміну інформацією про зміни в коді. Такий сервіс дозволить розробникам отримувати миттєві сповіщення про зміни в репозиторії, нові пулл-реквести та інші події, що стосуються розробки програмного забезпечення.

Сьогодні існують окремі рішення та бібліотеки, які можна використовувати для розробки сервісу сповіщень для систем контролю версій на основі Telegram бота. Деякі з них надають основну функціональність для взаємодії з системою контролю версій та месенджером, але є обмеження у розширюваності та конфігурації. Отже, розвиток нового сервісу сповіщень для систем контролю версій на основі Telegram бота має великий потенціал і може стати цінним інструментом для розробників програмного забезпечення.

У цьому розділі було проведено детальний огляд сучасного стану справ у області розробки сервісів сповіщень для систем контролю версій коду на основі Telegram бота. Наступні розділи дипломної роботи розкриють більш детально вимоги до розробленого сервісу, аналіз існуючих інструментів та рішень, опис архітектури та реалізацію. Всі ці відомості сприятимуть розумінню проблематики та розробці ефективного та потужного сервісу сповіщень для систем контролю версій на основі Telegram бота.

2.3 Модель вимог до інформаційної системи

У цьому розділі буде надано детальний опис моделі вимог до розроблюваної інформаційної системи — сервісу сповіщень для систем контролю версій коду на основі Telegram бота. Опис моделі вимог є критичним етапом у процесі розробки, оскільки він дозволяє чітко визначити функціональні та нефункціональні вимоги до системи, що допоможе забезпечити її успішну реалізацію та використання.

2.3.1 Функціональні вимоги

У цьому підрозділі будуть детально перераховані та описані функціональні вимоги до розроблюваної системи. Ці вимоги визначають функціональні можливості системи та операції, які вона повинна забезпечувати. Опис функціональних вимог відображає широкий спектр функціональних можливостей системи та деталізує їх характеристики.

1. Створення та налаштування Telegram бота. Бот повинен бути налаштований на підключення до відповідних систем контролю версій та взаємодію з ними.

2. Сповіщення про зміни в системах контролю версій: Система повинна здійснювати моніторинг систем контролю версій, та надсилати сповіщення користувачам про зміни, які відбуваються. Сповіщення повинні містити необхідну інформацію, таку як автора, дату, опис змін тощо.

3. Інтеграція з Telegram: Система повинна взаємодіяти з месенджером Telegram для передачі сповіщень користувачам. Це може включати

створення бота, підключення до API Telegram, обробку та відправку повідомлень.

4. Зберігання access токенів користувачів: Система повинна надійно зберігати access токени користувачів, які використовуються для авторизації та взаємодії з системами контролю версій. Забезпечення безпеки та конфіденційності цих токенів є важливим аспектом системи.

5. Моніторинг та журналювання: Система може мати можливості моніторингу та журналювання, щоб забезпечити відслідковування та аналіз стану системи, виявлення помилок та проблем, а також вирішення їх у найкоротший термін.

2.3.2 Нефункціональні вимоги

У цьому підрозділі будуть надані більш детальні описи нефункціональних вимог до розроблюваної системи сповіщень на основі Telegram бота для систем контролю версій коду. Нефункціональні вимоги стосуються якостей та властивостей системи, а також параметрів, що визначають її ефективність та надійність. Вимоги цього типу визначаються з урахуванням необхідних стандартів та очікувань користувачів.

1. Продуктивність. Необхідно забезпечити продуктивність системи, здатну ефективно обробляти великі обсяги сповіщень та запитів.

2. Безпека та конфіденційність. Система повинна гарантувати безпеку передачі даних між системою контролю версій та Telegram ботом. Необхідно забезпечити конфіденційність переданих даних, щоб

інформація про зміни коду не була доступна неавтентифікованим користувачам.

3. Масштабованість. Система повинна мати можливість масштабуватися для обробки великої кількості сповіщень та користувачів. Також необхідно врахувати можливість збільшення обсягу роботи системи та забезпечувати її стабільну роботу під високим навантаженням.

4. Надійність. Система повинна бути надійною та стійкою до відмов. Вона повинна продовжувати працювати навіть у разі виникнення помилок або проблем з підключенням. Необхідно передбачити механізми для виявлення та відновлення випадків відмов, щоб забезпечити безперебійну роботу системи.

5. Сумісність. Система повинна бути сумісною з різними системами контролю версій коду, що використовуються широкою спільнотою розробників. Необхідно забезпечити сумісність з різними версіями Telegram та його API, щоб забезпечити безперебійну взаємодію з користувачами.

2.3.3 Забезпечення безпеки

У цьому підрозділі будуть надані більш детальні вимоги щодо безпеки інформації та забезпечення конфіденційності в розроблюваній системі сповіщень на основі Telegram бота. Додатково до вимог, що були згадані раніше, також необхідно враховувати:

1. Аутентифікація та авторизація користувачів: Система повинна надавати механізми аутентифікації, щоб переконатися, що тільки вповноважені

користувачі мають доступ до системи сповіщень. Крім того, повинні бути встановлені механізми авторизації, які контролюють рівень доступу користувачів до різних проєктів та репозиторіїв.

2. Надійне зберігання access токенів: Access токени, які використовуються для авторизації Telegram бота та взаємодії з системою контролю версій коду, повинні бути збережені надійно. Це містить в собі застосування відповідних методів шифрування та зберігання в захищеному середовищі, щоб уникнути несанкціонованого доступу до конфіденційних даних.

2.3.4 Підтримка та документація

У цьому підрозділі будуть детальніше описані вимоги щодо підтримки та документації розроблюваної системи сповіщень на основі Telegram бота. Додатково до зазначених вимог можна врахувати:

1. Система підтримки користувачів: Розроблена система повинна мати належний механізм підтримки користувачів. Це може включати наявність контактної інформації, використовуючи яку користувачі можуть звернутися зі своїми запитаннями та проблемами.

2. Оновлення та підтримка: Розроблена система повинна мати механізми для оновлення та підтримки. Це означає, що коли з'являються нові версії Telegram або систем контролю версій коду, система сповіщень повинна бути сумісною з ними. Крім того, слід забезпечити регулярні оновлення системи сповіщень, задля розв'язання можливих проблем та вдосконалення функціональності.

3. Документація: Супровідна документація грає важливу роль у використанні та розробці системи. Розробникам та користувачам необхідно мати доступ до чіткої та зрозумілої документації, яка надає повне пояснення процесу налаштування, використання та розробки системи. Документація повинна включати опис необхідних кроків для налаштування Telegram бота та інші релевантні аспекти.

4. Технічна документація для розробників: Окрім загальної документації, система повинна мати докладну технічну документацію, яка призначена для розробників. Ця документація має надавати детальний опис архітектури системи, використаних технологій та компонентів.

2.4. Опис архітектури інформаційної системи

При розробці сервісу сповіщень для систем контролю версій коду, важливим аспектом є вибір підходящої архітектури для побудови системи. Два основних варіанти архітектури, які було розглянуто, це монолітна та мікросервісна архітектура.

Монолітна архітектура – це традиційний підхід до розробки програмного забезпечення, де весь додаток розгортається як єдиний компонент. У цьому випадку, вся логіка була б об'єднана в одному монолітному додатку. Хоча такий підхід може бути простим у початкових етапах розробки, він може стати проблематичним з часом.

Мікросервісна архітектура, з іншого боку, передбачає розбиття системи на невеликі незалежні компоненти – мікросервіси, які виконують окремі функції та комунікують між собою через API. У такому випадку, сервіс сповіщень може бути розбитий на окремі мікросервіси котрі призначені для виконання спеціальних атомарних задач.

Мікросервісна архітектура є кращим рішенням для сервісу сповіщень з наступних причин:

1. Модульність і розширюваність. Завдяки розбиттю системи на мікросервіси, вдалося отримати модульну структуру, де кожен сервіс відповідає за конкретну функцію. Це дозволяє легко розширювати систему, додавати нові функції або використовувати інші системи контролю версій коду та месенджери, не впливаючи на інші компоненти. Наприклад, можна додати підтримку для інших популярних систем контролю версій, або розширити сервіс сповіщень на інші платформи месенджерів.

2. Резистентність до помилок. У мікросервісній архітектурі, якщо один з мікросервісів несправний або відмовив, це не призведе до повного зупину системи. Тільки певні функції, які залежать від цього мікросервісу, можуть бути недоступні. Це дозволяє системі продовжувати працювати в цілому, навіть якщо виникають проблеми з окремими компонентами.

3. Швидкодія та масштабованість: У мікросервісній архітектурі існує можливість масштабування окремих мікросервісів незалежно один від одного, в залежності від їх потреб. Це дозволяє використовувати ресурси ефективно та забезпечувати швидкодію системи.

У контексті розробки сервісу сповіщень для систем контролю версій коду, мікросервісна архітектура виявляється кращим рішенням порівняно з монолітною архітектурою. Модульність, розширюваність, резистентність до помилок, швидкодія та масштабованість є ключовими перевагами мікросервісної архітектури, які відповідають вимогам проєкту. Використання мікросервісної архітектури дозволить ефективно і гнучко розробити сервіс сповіщень, а також легко розширити його функціональність на інші системи контролю версій та месенджери у майбутньому.

2.5 Опис функціональності системи

У цьому розділі будуть детально описані функціональні можливості розроблюваної системи. Опис функціональності системи допоможе краще зрозуміти, як система функціонує та які завдання вона виконує.

1. Аутентифікація через GitHub. Система надає можливість користувачам аутентифікуватися за допомогою своїх облікових записів GitHub. Аутентифікація через GitHub дозволяє зручно та безпечно використовувати систему, використовуючи існуючі облікові записи та спільний механізм авторизації.

2. Налаштування та редагування команди проєкту. Система надає можливість користувачам налаштовувати та редагувати команду проєкту, що включає учасників, відповідальних осіб та їх ролі.

3. Додавання репозиторію для відстеження пул-реквестів. Користувачі можуть додавати нові репозиторії до системи для відстеження пул-реквестів та отримання сповіщень про їх стан. Ця функція дозволяє користувачам включати до моніторингу конкретні репозиторії, щоб бути в курсі подальшого розвитку проєкту.

4. Зміна репозиторію для відстеження. Користувачі мають можливість змінювати репозиторій, який вони відстежують, щоб пристосовувати свій список спостереження до змін в проєкті або переорієнтуватися на інші проєкти. Зміна репозиторію дозволяє користувачам гнучко налаштовувати свій список сповіщень відповідно до їх поточних потреб.

5. Відправка сповіщень. Система автоматично відслідковує обраний репозиторій та надсилає користувачам сповіщення про нові не переглянуті пул-реквести.

6. Відправка повідомлень лише в робочий час. Система має можливість автоматично контролювати часові обмеження для відправки сповіщень. Система гарантує, що повідомлення будуть відправлятися лише протягом визначеного періоду, навіть якщо певний пул-реквест відкривається поза робочим часом.

7. Підтримка користувачів. Користувачі мають доступ до контактів підтримки, де вони можуть отримати відповіді на запитання та розв'язати проблеми, що виникають у процесі використання системи.

2.5.1 Telegram – GitHub аутентифікація

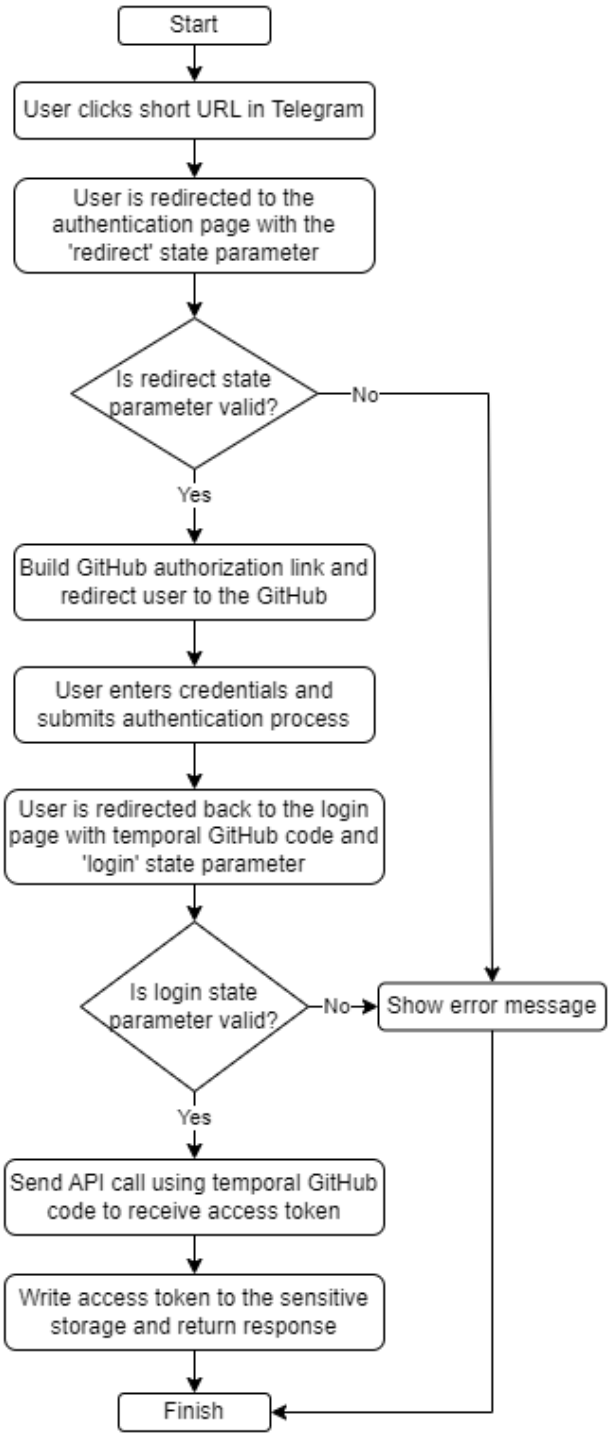


Рисунок 1. Процес аутентифікації між сервісами Telegram та GitHub

1) Користувач в телеграмі має постійне коротке посилання, яке прив'язане до його телеграм акаунту. Це посилання є унікальним URL, який асоціюється з конкретним користувачем. Це посилання використовується для ідентифікації користувача в системі.

2) Коли користувач натискає на коротке посилання, на серверній частині додатка відбувається процес побудови повного посилання, що веде на сторінку аутентифікації, і користувач перенаправляється на цю сторінку. Бекенд отримує запит з коротким посиланням і перетворює його на довге посилання, додаючи необхідні параметри, а саме “Redirect state” параметр.

3) На сторінці аутентифікації відправляється запит на сервер для валідації Redirect state параметру. Redirect state параметр – це унікальний ідентифікатор, прив’язаний до конкретного користувача. Важливо підкреслити, що цей параметр є одноразовим.

4) Сервер перевіряє валідність параметра та переконується, що він є дійсним і не був використаний раніше. Якщо redirect state проходить перевірку, він може бути оновлений, щоб уникнути його повторного використання в майбутньому. Це зроблено для того, щоб даний параметр неможливо було перевикористати та задля забезпечення додаткової безпеки.

5) Після успішної валідації redirect state параметру відбувається процес побудови посилання на сторінку аутентифікації GitHub. Це посилання включає необхідні параметри, такі як:
    - Client Id – унікальний ідентифікатор додатка в системі GitHub.

- Scopes – обсяг можливостей та інформації котра буде доступна додатку після завершення процесу аутентифікації.
- Login state - унікальний ідентифікатор користувача під час процесу аутентифікації.

Користувач перенаправляється на сторінку аутентифікації GitHub, де він може увійти до свого облікового запису GitHub та надати дозвіл на доступ до необхідних даних та ресурсів.

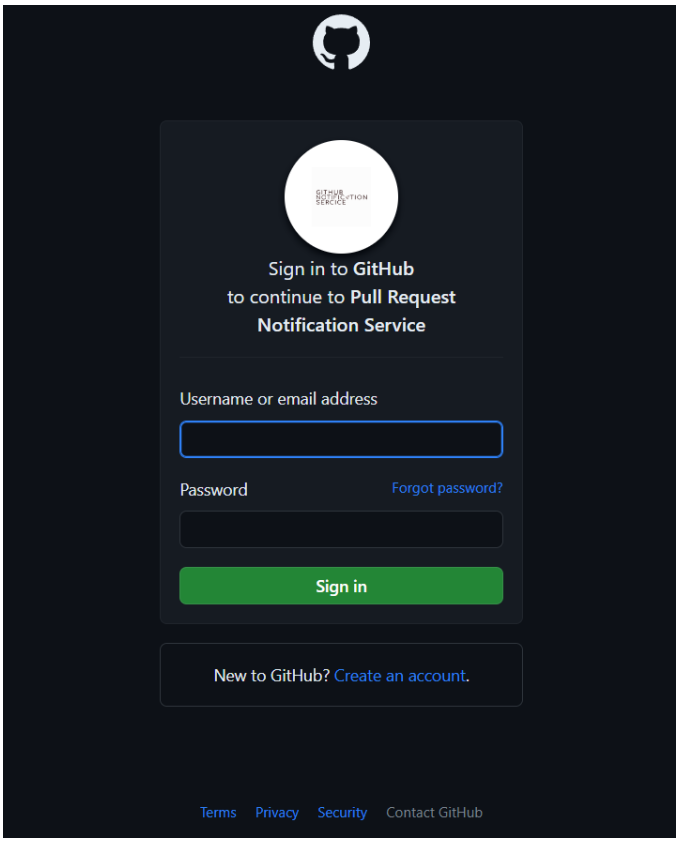


Рисунок 2. Користувач перенаправлений на сторінку аутентифікації

6) Користувач вводить дані необхідні для входу в акаунт та приймає пропозицію авторизації у додаток через вбудовані механізми GitHub.

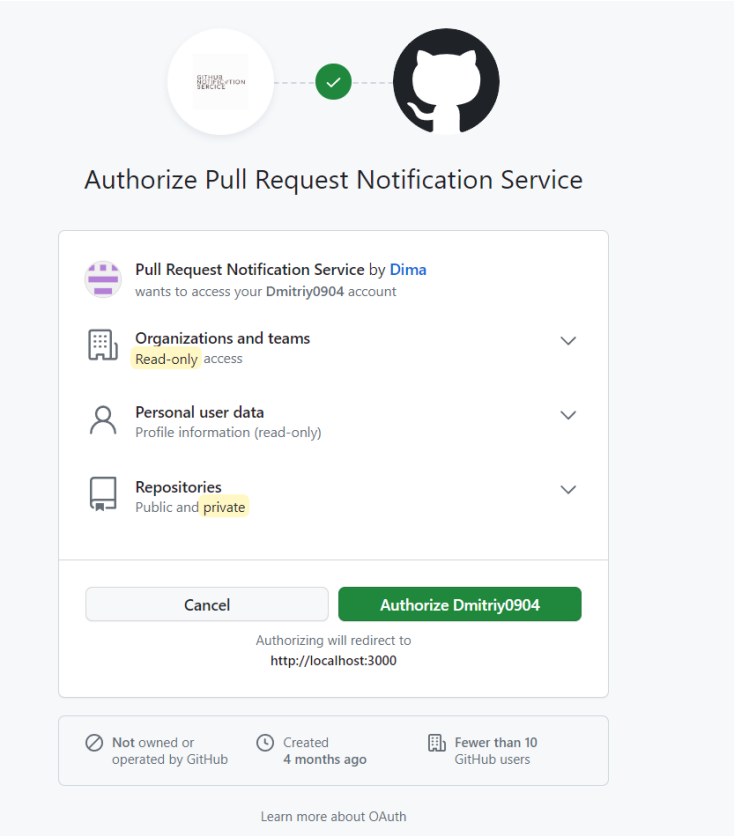


Рисунок 3. Користувач може побачити інформацію, котра буде доступна додатку

7) Користувач перенаправляється назад на сторінку логіну з тимчасовим GitHub кодом котрий був автоматично згенерований сервісом, та унікальним “Login State” параметром. Це забезпечує безпеку інформації, що передається між додатком та GitHub.

8) На сторінці логіну відбувається перевірка login state параметру використовуючи API запит на сервер. Серверна частина додатка отримує запит і перевіряє валідність параметра, котрий був переданий на попередньому кроці. Якщо параметр є валідним, він оновлюється для уникнення повторного використання, і бекенд продовжує виконання процесу аутентифікації.

9) Після успішної перевірки параметра, сервер відправляє запит на GitHub API для отримання постійного access токену. Для цього, сервер передає

тимчасовий GitHub код, який був отриманий на попередньому кроці. GitHub API перевіряє тимчасовий код та ідентифікує додаток, після чого повертає постійний access токен для подальшого використання при роботі програми.

10) Якщо відповідь від GitHub API прийшла успішно і містить необхідні дані включаючи access токен, то цей токен зберігається в HashiCorp Vault використовуючи системний мікросервіс для роботи з чутливими даними.

11) Після збереження access токену, користувач бачить повідомлення про успішну аутентифікацію в додатку і перенаправляється назад в Telegram, де може продовжити використання додатка вже з аутентифікованим обліковим записом.

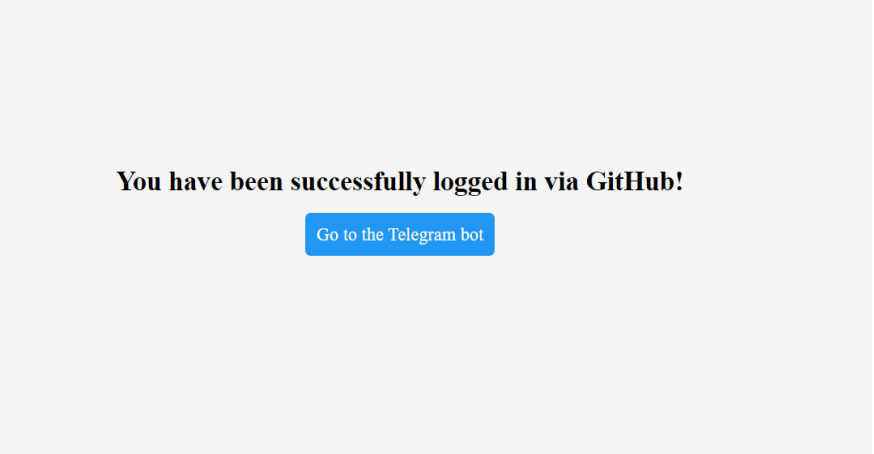


Рисунок 4. Користувач бачить напис про успішну аутентифікацію

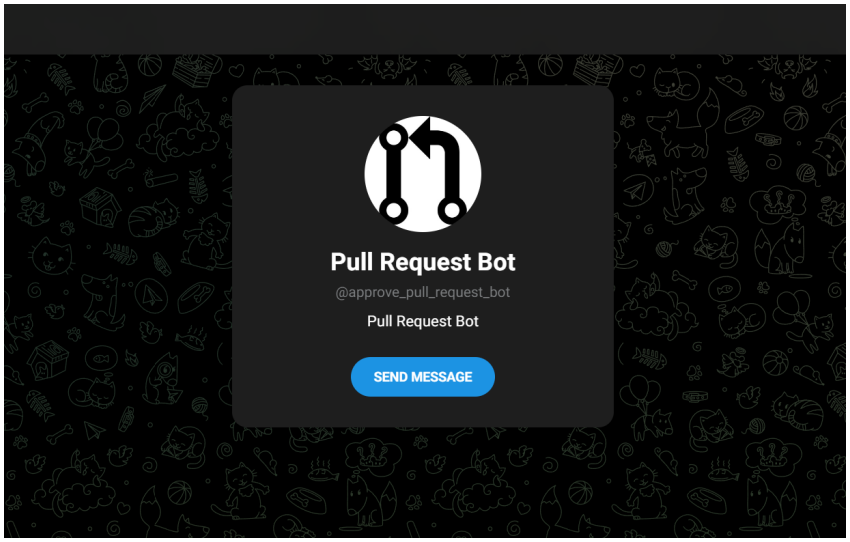


Рисунок 5. Користувач перенаправляється в месенджер Telegram

У цьому розділі був детально розглянутий процес аутентифікації між Telegram та GitHub в додатку. Починаючи з короткого посилання в Telegram, яке перенаправляє користувача на сторінку аутентифікації, проходженням через перевірки додаткових параметрів та отриманням GitHub access токену, де кожен крок процесу має свою роль у забезпеченні безпеки та ідентифікації користувача.

Описаний процес аутентифікації надає користувачам зручний шлях для входу в додаток, використовуючи свій обліковий запис Telegram та забезпечуючи безпеку обміну даними використовуючи вбудовані механізми аутентифікації та авторизації GitHub.

2.5.2 Відправка сповіщень

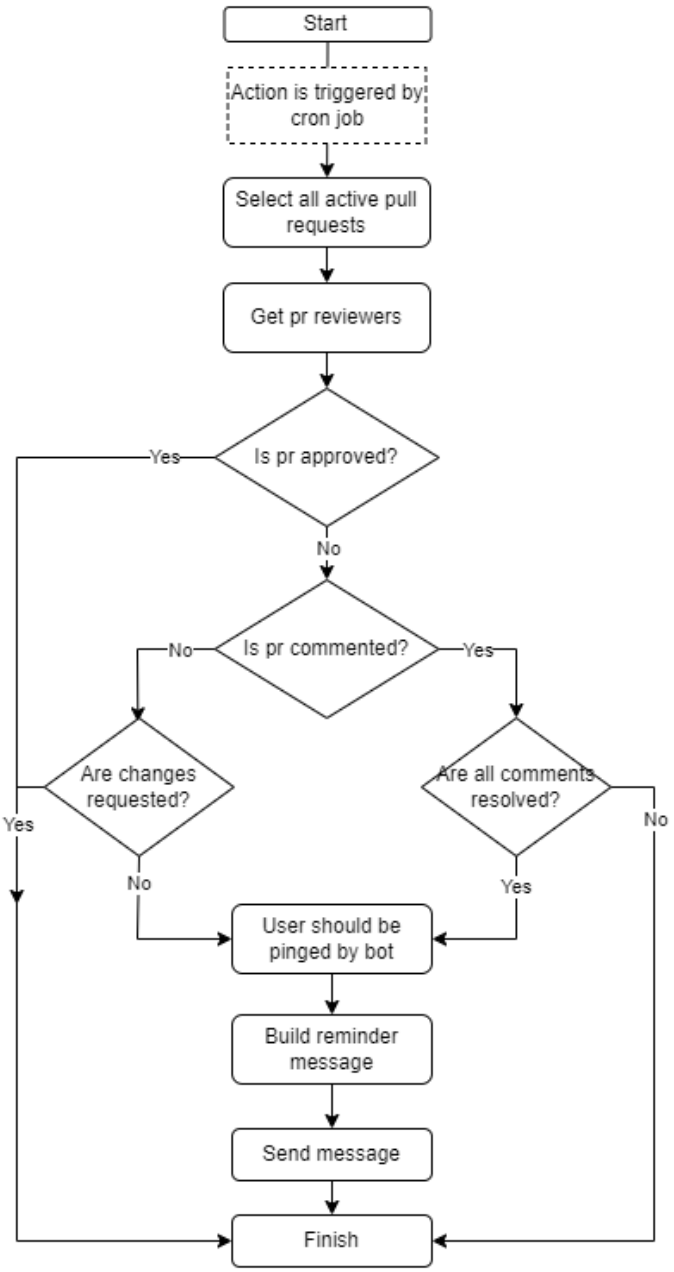


Рисунок 6. Процес обробки інформації про пул-реквести та відправки сповіщень

1) Процес відправки сповіщень автоматично відпрацьовує тричі на робочий день в задані часи використовуючи cron механізм. Cron механізм – це інструмент в операційних системах, що дозволяє запускати програми або сценарії автоматично в задані часи або з заданими інтервалами. В даній роботі цей підхід імплементований за

допомогою окремого мікросервісу з використанням спеціальних Spring Boot анотацій.

2) Здійснюється відбір пул-реквестів, які є активними, тобто ті, що потребують перегляду та рев'ю. Активними пул-реквестами є ті, котрі не були закриті або злиті у репозиторії та продовжують перебувати у відкритому стані.

3) Для кожного активного пул-реквесту отримується список рев'юверів. Рев'ювери – це члени команди, яким необхідно переглянути пул-реквест та надати свої коментарі та відгуки. Список рев'юверів може бути отриманий з метаданих пул-реквесту в GitHub за допомогою API запиту.

4) Для кожного рев'ювера проводяться перевірки з метою визначення, яким чином нагадування мають бути відправлені.
    a) Якщо пул-реквест вже схвалений. Якщо користувач вже схвалив пул-реквест, це означає, що він переглянув його, зрозумів зміст та вважає його прийнятним для злиття. В такому випадку, нагадування не потрібні, оскільки рев'ювер вже завершив свою роботу.
    b) Якщо пул-реквест отримав коментарі, але вони не були вирішені. У такому випадку, хоча рев'ювер переглянув пул-реквест і залишив коментарі, він чекає, щоб автор вніс відповідні зміни або відповів на коментарі. Таким чином, нагадування не надсилаються, оскільки вважається, що автор пул-реквесту повинен спочатку вирішити, або відповісти на коментарі рев'ювера.

c) Якщо рев'ювер не переглянув пул-реквест або не виконав жодних дій, це означає, що рев'ювер ще не пройшов через процес рев'ю або може бути забув про нього. В такому випадку, рев'ювер додається до списку для відправки нагадувань.

На основі цих перевірок визначається, чи потрібно відправляти нагадування користувачам щодо їхньої участі у процесу код рев’ю. Таким чином, процес перевірки статусів рев'юверів допомагає визначити, хто з користувачів потребує додаткового нагадування.

5) Після перевірки кожного рев’ювера відбувається процес побудови нагадувань. На цьому етапі генерується відповідний текстовий контент для нагадувань, який буде включати інформацію про пул-реквест та необхідні дії користувача.

6) Згенеровані нагадування відправляються користувачам у додатку Telegram за допомогою додаткової проєктної залежності Telegrambots Spring Boot Starter.

Таким чином, процес відправки нагадувань автоматично перевіряє статуси пул-реквестів та надсилає нагадування розробникам, котрі не переглянули пул-реквест або не виконали жодних дій, щоб забезпечити активний процес код рев'ю.

2.6 Використані технології та обґрунтування вибраного інструментарію

2.6.1 Серверна частина

Для реалізації серверної частини системи була обрана Java 17 - потужна та широко використовувана мова програмування, яка забезпечує високу продуктивність та надійність. Java має велику екосистему інструментів та бібліотек, що полегшує розробку складних проєктів.

Для керування залежностями та збірки додатка було використано Maven - потужний та популярний інструмент з автоматизації збірки, який спрощує управління залежностями та будівництво проєкту.

Для контролю версій коду використовується Git, а збереження репозиторію з кодом здійснюється на платформі GitHub, що є широко використовуваним та надійним сервісом для спільної роботи над проєктами.

Основні технології та залежності:

- Spring Boot 3.0.3: Spring Boot є потужним фреймворком для розробки вебдодатків. Він надає зручність та простоту у налаштуванні та створенні мікросервісів, що робить його ідеальним вибором для проєкту.
- Spring Boot Web: Ця залежність надає можливості для створення вебдодатків з використанням Spring Boot. Вона дозволяє нам створювати контролери, обробляти HTTP-запити та взаємодіяти з клієнтами.
- Spring Boot Data JPA: Використовувати Spring Boot Data JPA дозволяє зручно працювати з базою даних та реалізовувати репозиторії з можливістю доступу до даних.

- Spring Boot Validation: Ця бібліотека дозволяє проводити валідацію вхідних даних та забезпечує їхню коректність та безпеку.
- Telegrambots Spring Boot Starter: Цей стартер спрощує інтеграцію з Telegram API та дозволяє створювати Telegram ботів. Він забезпечує зручну роботу з повідомленнями, клавіатурами та іншими функціями Telegram бота.
- Spring Cloud Vault (Core / Config): Використання Spring Cloud Vault дозволяє безпечно зберігати конфіденційні дані, такі як паролі, ключі API тощо. Ця залежність забезпечує інтеграцію з HashiCorp Vault - інструментом для керування секретами.

2.6.2 Клієнтська частина

Основним компонентом клієнтської частини став безпосередньо месенджер Telegram. Одним з найбільш визнаних аспектів Telegram є його інтуїтивний, привабливий та зрозумілий користувачу інтерфейс.

Для розробки сторінки аутентифікації між Telegram та GitHub, була використана технологія React. React - це відкрита JavaScript бібліотека, що використовується для створення високоефективних інтерфейсів користувача. Вона забезпечує швидку та ефективну роботу з користувацьким інтерфейсом, надає компонентний підхід та спрощує управління станом додатка.

Інтерфейс користувача розробленого додатка поєднав зручність та функціональність завдяки використанню Telegram як основного інтерфейсу користувача, та технології React для сторінки аутентифікації як допоміжну ланку у досягненні широкого спектра функціональних можливостей продукту.

2.6.3 База даних

В рамках дослідження було здійснено порівняння двох популярних систем управління базами даних (СУБД) - MySQL та PostgreSQL. Метою дослідження було визначити переваги та недоліки кожної з СУБД з огляду на ключові параметри, такі як безпека, масштабованість, швидкість та функціональність. Хоча обидві СУБД мають свої переваги та недоліки, PostgreSQL може бути кращим вибором з численних причин:

1. Безпека. PostgreSQL відома своїм високим рівнем безпеки. Вона надає широкий спектр механізми для захисту даних, таких як аутентифікація, авторизація, шифрування та механізми контролю доступу. PostgreSQL також має активну спільноту, яка швидко реагує на виявлення потенційних вразливостей і надає оновлення безпеки. Це дозволяє захистити дані від несанкціонованого доступу та зберігати їх у безпечному середовищі.

2. Масштабованість. PostgreSQL підтримує горизонтальне та вертикальне масштабування. Вона має вбудовану підтримку реплікації, яка дозволяє розподілити навантаження між кількома серверами. Крім того, PostgreSQL може працювати на великих обсягах даних і масштабуватись до більшого числа одночасних підключень.

3. Швидкість. PostgreSQL володіє оптимізованим механізмом обробки запитів, який забезпечує високу продуктивність при виконанні складних запитів. Вона також має розширені можливості для налаштування параметрів процесу оптимізації запитів. PostgreSQL підтримує індексацію, що дозволяє прискорити пошук та фільтрацію даних.

4. Функціональність. PostgreSQL володіє багатим набором функцій, включаючи вирази, процедури, тригери, географічні типи даних та багато іншого. Вона підтримує ряд сучасних функцій, таких як JSON/XML-обробка, розширення для роботи з графовими даними та розширені можливості для роботи з реляційними даними.

З урахуванням вищезгаданих факторів, PostgreSQL може бути кращим вибором у порівнянні з MySQL. Вона надає високий рівень безпеки, гнучкість масштабування, високу швидкість та широкий набір функціональних можливостей що дозволяє забезпечити надійне та ефективне зберігання даних, зручну маніпуляцію даними та розширення можливостей бази даних у майбутньому.

2.6.4 Зберігання чутливих даних

У сучасному світі безпека конфіденційних даних є пріоритетною задачею для багатьох організацій. Зберігання та керування секретами, такими як паролі, ключі API та сертифікати, є необхідною складовою для забезпечення безпеки додатків. У роботі було детально розглянуто два популярних рішення - HashiCorp Vault та CyberArk.

Функціональність:

- HashiCorp Vault: Vault - розширювана платформа, яка надає широкі можливості для зберігання, керування та автоматизації доступу до секретів. Він підтримує різноманітні типи секретів, включаючи паролі, ключі API та сертифікати. Vault також надає потужні механізми автентифікації, авторизації та аудиту, що дозволяє точно керувати доступом до секретів.

- CyberArk: CyberArk - комплексна платформа для управління привілеями, яка забезпечує контроль і моніторинг доступу до секретів та інших конфіденційних даних. Вона надає функції безпеки паролів, управління доступом та аналітики. Однак, в порівнянні з Vault, функціональність CyberArk може бути менш розширеною і не такою гнучкою.

Інтеграція:

- HashiCorp Vault: Vault має широку підтримку інтеграції з різними інструментами та технологіями. Він може бути легко інтегрований із Spring Boot додатками за допомогою плагінів та бібліотек, що спрощує роботу з секретами.
- CyberArk: Хоча CyberArk також надає інтеграційні можливості, його рівень інтеграції з платформою Spring Boot може бути менш поглибленим порівняно з Vault. Це може призвести до складнішого процесу інтеграції та налаштування.

Вартість:

- HashiCorp Vault: Vault пропонує безплатну версію (Open Source). Комерційна версія Vault (Enterprise) надає розширений функціонал та підтримку, але має вартість ліцензії залежно від обсягу використання.
- CyberArk: CyberArk є комерційним продуктом, що вимагає ліцензії, оренди або підписки. Використання CyberArk може бути пов'язано з додатковими витратами, що може вплинути на загальний бюджет проєкту.

З урахуванням функціональності, інтеграційних можливостей та вартості, HashiCorp Vault видається виграшним рішенням для зберігання сенситивних даних у проєкті. Використання Vault в системі дозволяє забезпечити

безпечне зберігання та керування токенами, що використовуються для автентифікації та доступу до інших систем або API. Використання такої технології допомагає уникнути витоку конфіденційної інформації та підвищує загальну безпеку системи.

2.6.5 Додаткові залежності та бібліотеки

У розробці програмного забезпечення використання додаткових бібліотек та залежностей є поширеною практикою. Бібліотеки надають готові реалізації різних функцій і операцій, які можна використовувати в програмному коді, щоб прискорити розробку, покращити продуктивність та забезпечити більшу функціональність програми. При розробці додатка були використані наступні бібліотеки:

- Apache Commons - це набір корисних бібліотек, котрі надають широкий спектр функцій і утиліт, які можуть бути використані для розв'язання різноманітних завдань. Apache Commons був використаний для роботи з рядками, колекціями, та математичними операціями.

- Slf4j & Logback. Застосування Slf4j та Logback у програмному коді дозволило ефективно і зручно керувати логуванням подій та помилок, спростило налагодження та аналіз програми та допомогло забезпечити якість та надійність програмного забезпечення.
- com-github-snksoft: Ця залежність надала додаткові функціональні можливості, які були використані в системі для генерації коротких посилань.

Вибір цих технологій та інструментів обґрунтовується їхньою потужністю, популярністю, великою спільнотою розробників та зручністю використання для розробки сервісу сповіщень для систем контролю версій коду на основі

Telegram бота. Вони дозволять забезпечити надійну та ефективну роботу системи, зручну взаємодію з користувачами та забезпечать широкі можливості для подальшого розширення та покращення системи.

2.7 Результати тестування

Після завершення розробки сервісу сповіщень для систем контролю версій коду на основі Telegram бота було проведено тестування його функціональності для перевірки коректності та надійності роботи. Нижче наведені основні результати тестування:

1. Взаємодія з системою контролю версій. Під час тестування було перевірено роботу сервісу з системою контролю версій Git та сховищем репозиторіїв GitHub. Встановлено, що сервіс успішно встановлює зв'язок із цими системами та отримує необхідну інформацію про зміни в репозиторіях. Забезпечена коректна синхронізація між сервісами, що дозволяє надсилати актуальні сповіщення користувачам.

2. Аутентифікація та авторизація. Тестування включало перевірку процесу автентифікації та авторизації користувачів. Встановлено, що сервіс успішно перевіряє облікові дані користувачів і дозволяє доступ тільки аутентифікованим та авторизованим користувачам. Це забезпечує конфіденційність та безпеку отримання інформації.

3. Надсилання сповіщень. Під час тестування була перевірена точність, швидкість та надійність надсилання сповіщень користувачам. Встановлено, що сервіс надсилає сповіщення вчасно та доставляє їх до користувачів без затримок. Текст повідомлень форматується зрозуміло та чітко, що дозволяє користувачам швидко отримати інформацію про зміни в коді.

4. Надійність та стабільність: Під час тестування проводилось перевірка надійності та стабільності роботи сервісу. Було виконано тестування в різних умовах, включаючи високе навантаження та непередбачені

ситуації. Сервіс продемонстрував стабільну роботу та виявив мінімальний рівень відмов. Були застосовані заходи для забезпечення відновлюваності після можливих випадків втрати зв'язку або інших помилок.

5. Інтерфейс та користувальницький досвід: Під час тестування оцінювався зовнішній вигляд та зручність використання сервісу. Було встановлено, що інтерфейс є зрозумілим, що сприяє легкій навігації та використанню функцій сервісу. Користувачі отримують зручний і позитивний досвід взаємодії із кінцевим продуктом.

6. Журналювання та аналітика: Перевірка наявності системи журналювання подій та можливості аналізу зібраних даних. Оцінка зручності аналізу журналів для виявлення помилок, відстеження активності користувачів та поліпшення продуктивності сервісу.

В цілому, результати тестування підтвердили успішну розробку та ефективну роботу сервісу сповіщень для систем контролю версій коду на основі Telegram бота. Виконані пункти завдань та вимоги до сервісу були успішно виконані, що дає змогу зробити висновок про його придатність та цінність для розробників програмного забезпечення.

3. ВИСНОВКИ

У рамках даної дипломної роботи було поставлено завдання розробки сервісу сповіщень для систем контролю версій коду на основі Telegram бота. Метою роботи було створення ефективного і зручного сервісу, який забезпечує користувачів систем контролю версій зручним способом отримувати повідомлення про зміни та події в системі.

Основні задачі, які було вирішено у рамках розробки сервісу сповіщень, включають:

1. Розробка інтерфейсу зв'язку між системою контролю версій і Telegram ботом: Було налаштовано зв'язок між сервісом сповіщень та Telegram ботом, використовуючи API Telegram. Це дозволило користувачам отримувати сповіщення про зміни в коді безпосередньо у своєму Telegram-акаунті.

2. Розробка механізму відстеження подій: Було реалізовано механізм, який відстежує події, що відбуваються в системі контролю версій. При відкритті пул-реквесту, або інших подіях в системі, сервіс сповіщень автоматично надсилає повідомлення користувачам.

3. Розробка механізму аутентифікації та авторизації: Для забезпечення безпеки та конфіденційності користувачів було реалізовано механізм автентифікації та авторизації. Це означає, що лише аутентифіковані та авторизовані користувачі мають доступ до сервісу сповіщень. Це забезпечує, що інформація про зміни в коді залишається конфіденційною та доступною тільки відповідним користувачам.

4. Тестування та налагодження сервісу: Після розробки основних компонентів сервісу, було проведено тестування його функціональності.

Тестування допомогло переконатись в коректній роботі сервісу та виявити та виправити можливі помилки та недоліки.

У результаті роботи було розроблено інтегрований, зручний та ефективний сервіс сповіщень для систем контролю версій коду на основі Telegram бота. Цей сервіс дозволяє користувачам отримувати актуальну інформацію про зміни в системах контролю версій швидко і зручно, спрощуючи їх роботу та поліпшуючи продуктивність.

Розроблений сервіс має потенціал для подальшого розширення та покращення. Наприклад, можливі напрямки розвитку включають підтримку інших популярних систем контролю версій, розширення функціональності сповіщень, таких як налаштування фільтрів або налаштування рівня деталізації сповіщень, а також підтримку додаткових можливостей Telegram бота, таких як взаємодія з іншими сервісами чи розширені можливості сповіщень через використання кнопок та клавіатури.

У цілому, розробка даного сервісу сповіщень для систем контролю версій коду на основі Telegram бота є важливим кроком у поліпшенні процесу роботи з системами контролю версій коду та забезпеченні користувачів актуальною інформацією про зміни в їх проєктах.

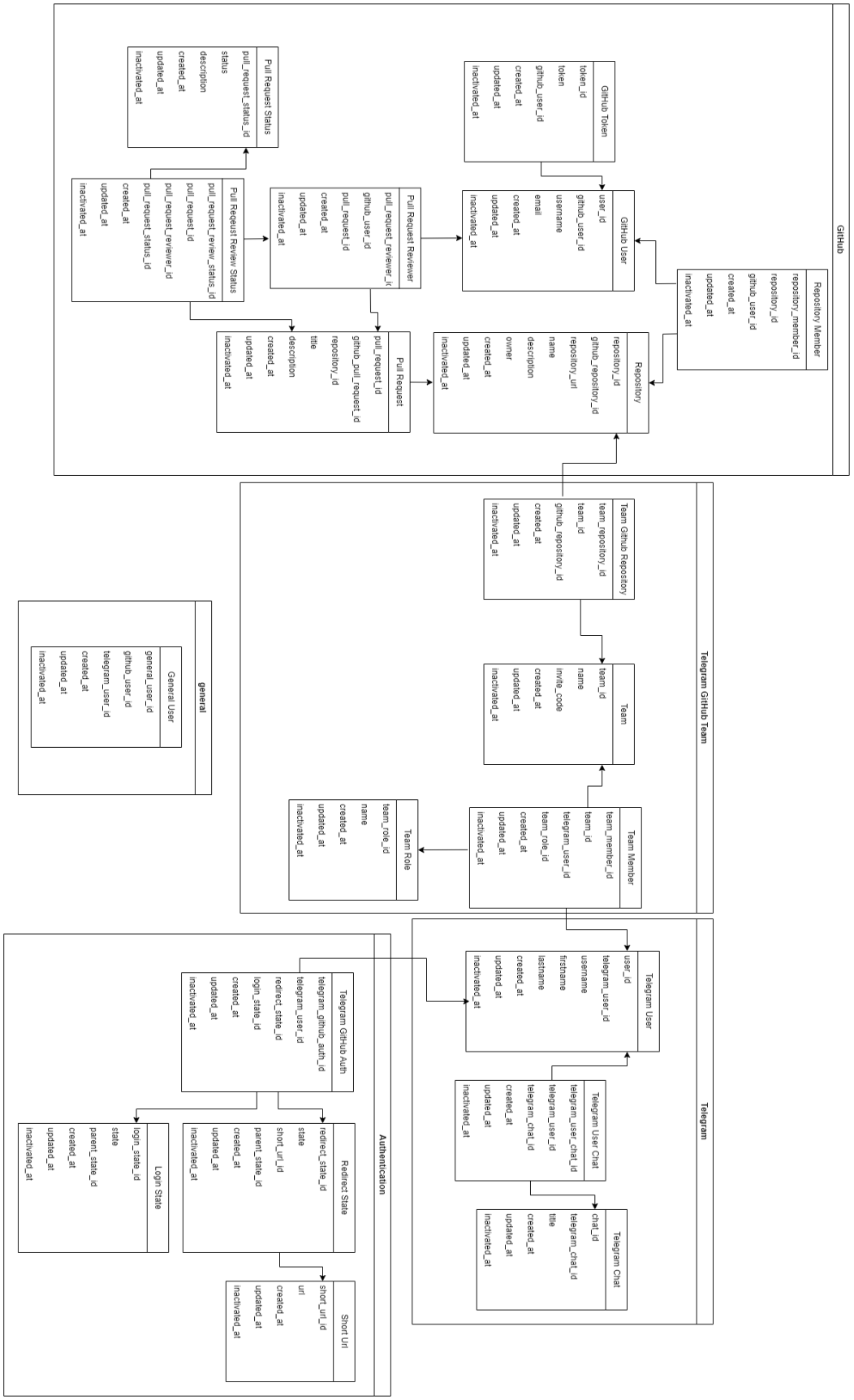
4. СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Git – Documentation. URL: https://git-scm.com/doc
2. GitHub – Help Documentation, *GitHub Docs*. URL: https://docs.github.com/
3. JDK 17 Documentation. *Oracle Help Center*, Jan. 25, 2022. URL: https://docs.oracle.com/en/java/javase/17/
4. Maven Documentation. *J. Van Zyl*, Aug. 01, 2009. URL: https://maven.apache.org/guides/
5. Spring Boot Documentation. URL: https://spring.io/projects/spring-boot
6. Spring Framework Documentation. URL: https://spring.io/projects/spring-framework
7. Spring Web Documentation. URL: https://docs.spring.io/spring-boot/docs/current/reference/html/web.html
8. Spring Data JPA Documentation. URL: https://spring.io/projects/spring-data-jpa
9. Spring Validation, Data Binding, Type Conversion – Documentation. URL: https://docs.spring.io/spring-framework/docs/3.0.x/spring-framework-reference/html/validation.html
10. TelegramBots: Java library to create bots using Telegram Bots API. *Rubenlagus*. URL: https://github.com/rubenlagus/TelegramBots
11. Spring Vault Documentation. URL: https://spring.io/projects/spring-vault
12. Spring Cloud Vault Documentation. URL: https://spring.io/projects/spring-cloud-vault
13. Apache Commons Documentation. *A. C. D. Team*. URL: https://commons.apache.org/
14. SLF4J – Documentation. URL: http://www.slf4j.org/
15. HashiCorp Vault - Developer Documentation. URL: https://developer.hashicorp.com/vault/docs
16. GitHub - snksoft/java-crc: Generic CRC implementation for java language - documentation. *Snksoft*. URL: https://github.com/snksoft/java-crc
17. GitHub REST API Documentation. *GitHub Docs*, Nov. 28, 2022. URL: https://docs.github.com/en/rest?apiVersion=2022-11-28
18. Telegram APIs Documentation. URL: https://core.telegram.org/api
19. Telegram Bots: An introduction for developers. URL: https://core.telegram.org/bots

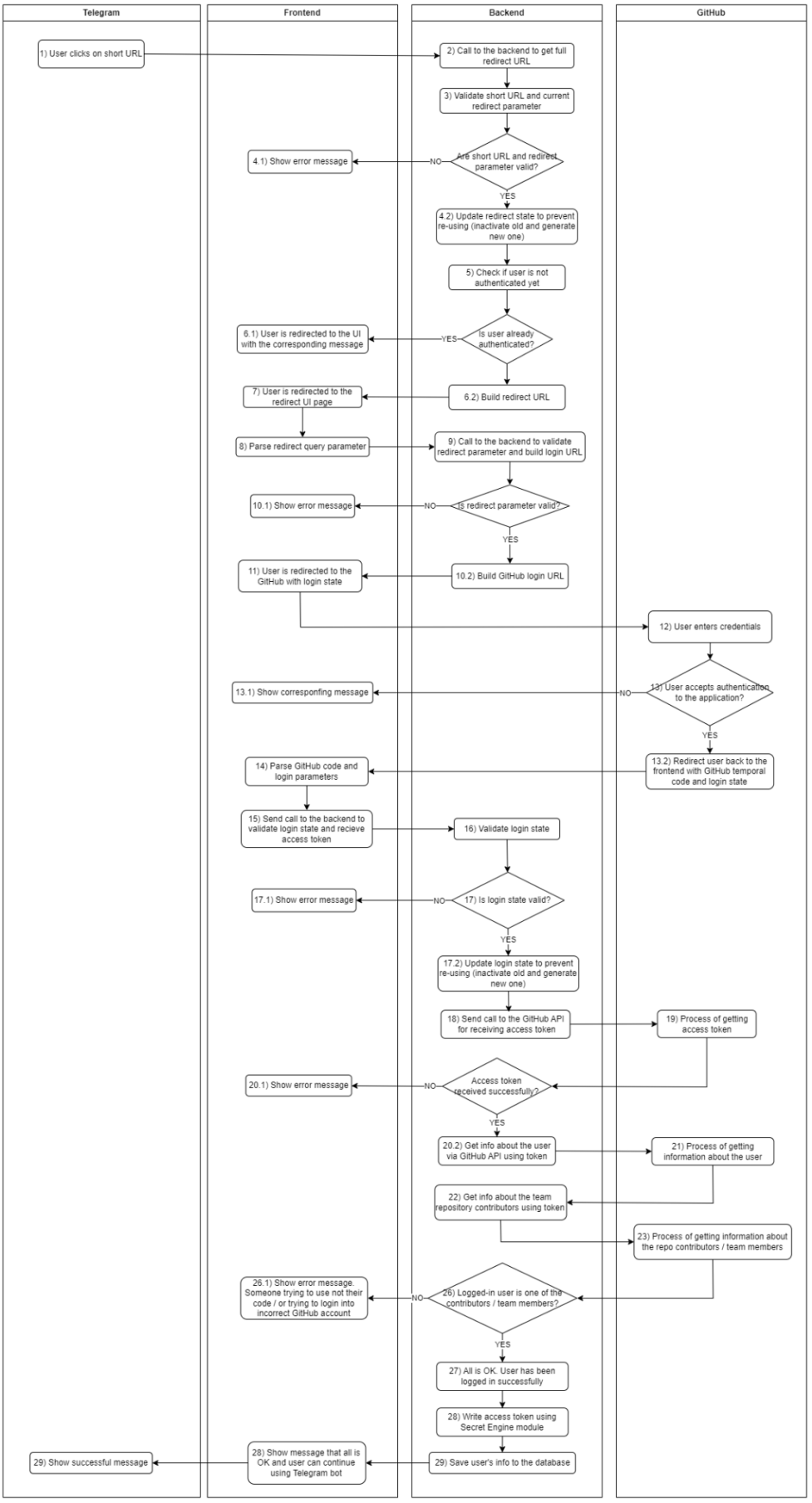
20. Quick Start – React Documentation. URL: https://react.dev/learn
21. PostgreSQL 10.23 Documentation. *PostgreSQL Documentation*, Nov. 10, 2022. URL: https://www.postgresql.org/docs/10/index.html
22. CyberArk Secrets Management. *CyberArk Software*, Jan. 05, 2023. URL: https://www.cyberark.com/products/secrets-management/
23. JUnit 5 User Guide. *S. B. S. B. De Rancourt Christian Stein Johannes Link, Matthias Merdes, Marc Philipp, Juliette*. URL: https://junit.org/junit5/docs/current/user-guide/
24. MySQL Documentation. URL: https://dev.mysql.com/doc/
25. Authorizing OAuth Apps. GitHub Docs. URL: https://docs.github.com/en/apps/oauth-apps/building-oauth-apps/authorizing-oauth-apps

5. ДОДАТКИ

Додаток 1. Діаграма бази даних



Додаток 2. Детальний процес Telegram – GitHub аутентифікації



Додаток 3. Детальний процес відправки сповіщень