1. Огляд предметної області
   1. Опис проблеми. Актуальність

Сучасні месенджери досить придатні для того, щоб інтегрувати багато корисних функцій у них, зазвичай це робиться за допомогою чат ботів. Чат бот – це програма, яка імітує реальний діалог з користувачем. Чат боти присутні на багатьох рекламних сайтах в якості асистентів. Також сучасні месенджери дають платформу для створення чат ботів у рамках їхнього додатку для спілкування. Ці боти дають змогу виконувати деякі складні дії лише даючи текстові, аудіо або графічні команди.

Кожен месенджер має свій інтерфейс для розробки і керування чат ботами. Для розробки програмного продукту був обраний месенджер «Telegram».

Основні переваги Telegram:

* Популярність  
  На теперішній час Telegram займає першу позицію у рейтингу Google Play серед інших месенджерів, таким чином розробка боту під цю платформу охопить більше авдиторії
* Зручність  
  Telegram має досить багато вбудованих функцій, які дозволяють досить зручно інтегрувати майже будь-який інтерфейс.

Досить часто люди ведуть бізнес через месенджери. Іноді для цього залучають спеціальних чат ботів, які допомагають робити купівлі, відповідають на часті запитання. Але для декого створення повністю свого боту є не зовсім доречним або занадто складним за багатьма причинами. Ці люди часто вимушені надсилати однакові або майже однакові повідомлення багатьом контактам. Знаходження схожого повідомлення у попередніх чатах, його копіювання, вставка цього тексту у потрібний чат, можливо дрібне редагування може займати досить багато часу. Мій бот пропонує рішення цієї проблеми.

«Keyword Binder Bot» дозволяє зберігати деякі шаблонні повідомлення та отримувати до них доступ майже одразу у будь-якому чаті. Цей бот працює індивідуально з кожним користувачем, тому ці шаблони доступні лише тому, хто їх створив у цілях безпеки.

Цим ботом може користуватися будь-хто, від старости, який зможе одразу надіслати усі контакти викладачів будь-кому хто його попросить, до адміністрації закладів, які повідомляють про необхідність продовження абонементу до зали. Цей бот простий у використанні та не потребує часу для налаштування. Невелика інфографіка для KeyBinderBot представлена на рисунку 1.1.

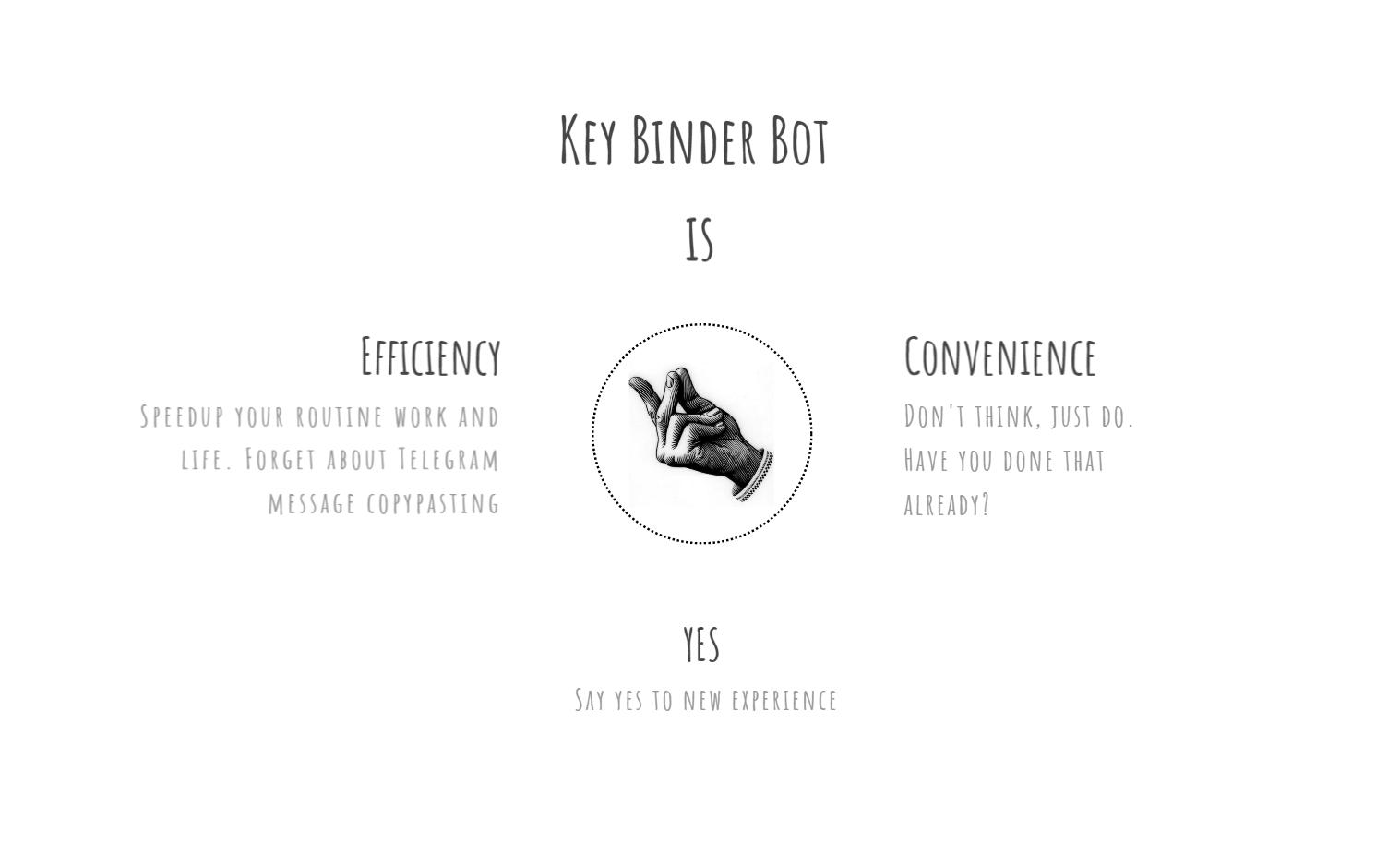


Рис. 1.1 – Інфографіка

* 1. Огляд аналогів

SendPulse – це платформа, що дозволяє створювати ботів для багатьох месенджерів. Вона дозволяє налаштовувати автовідповіді, запускати розсилки і оптимізувати бізнес процеси за допомогою бота [https://sendpulse.ua/features/chatbot/telegram]. Ця платформа потребує деякого розуміння принципу роботи ботів та змушує створювати свого бота для окремої задачі через спеціальній графічний конструктор (рис. 1.2.1). Також цей сервіс є умовно безкоштовним до деякої кількості користувачів.

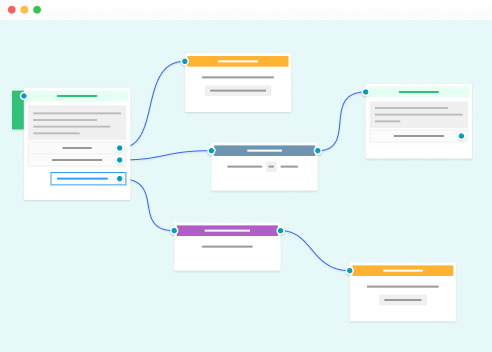


Рис. 1.2.1 – Графічний конструктор SendPulse

На відміну від SendPulse, KeyBinderBot не створює нового бота під кожного користувача, а працює одразу для всіх. Будь-хто може почати користуватися ним коли завгодно та де завгодно.

PrivateBoxBot – це бот, який надає деяке сховище для зберігання даних різних видів, деякі з них представлені на рисунку 1.2.2. На відміну від попереднього аналогу цей бот не змушує створювати окремого бота для кожного користувача, однак це лише сховище, яке не дозволяє проводити операції з даними під час їх використання. Також він є не досить зрозумілим для використання.

На відміну від PrivateBoxBot, KeyBinderBot має набагато простіший інтерфейс, що не потребує жодних додаткових налаштувань та дозволяє проводити деякі зміни у створених шаблонах шляхом використання змінних.

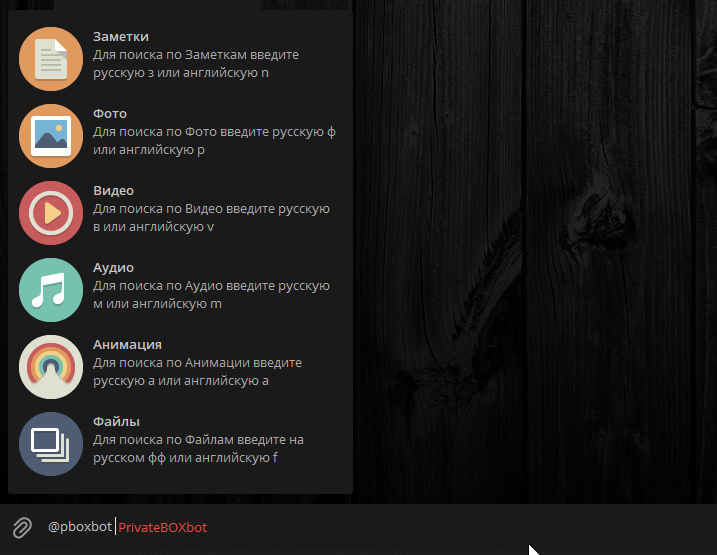


Рис.1.2.2 – Формати даних PrivateBoxBot

Для більш наочного порівняння цих аналогів з даним продуктом була створена порівняльна таблиця 1.2.3.

|  |  |  |  |
| --- | --- | --- | --- |
| Ознака | KeyBinderBot | SendPulse | PrivateBoxBot |
| Гнучкість | Дозволяє зберігати будь-яку інформацію, що підтримується Telegram, та примітивний функціонал для редагування ціх даних під час використання | Майже не має обмеженнь у можливостях та сферах використання | Дозволяє лише зберігати та відтворювати віділені формати даних |
| Поріг входження | Функціонал доступний для будь-кого, для початку використання досить лише написати /start та створити шаблон | Перед початком роботи треба повністю скласти бота за допомогою спеціального графічного інтерфейсу | Перед початком використання треба пройти досить довгий процесс реєстрації. |
| Вартість | Умовно безкоштовний, залежить від кількості збережених шаблонів | Умовно безкоштовний, залежить від потоку користувачів | Умовно Безкоштовний, залежить від максимально обсягу даних, що можуть бути збережені |

Таблиця 1.2.3 – Порівняння аналогів

* 1. Технічне завдання

Технічне завдання (ТЗ) – це затверджений документ, на основі якого виконується розробка проекту. У ньому максимально точно і детально описані вимоги до компонентів та характеристик майбутнього продукту. [https://internetdevels.ua/blog/technical-specification-project-management]

* + 1. Словник
* Нікнейм – Ім’я користувача. Обов’язково є у будь-яких Telegram ботів та необов’язково є у звичайних користувачів.
* Інлайн – режим Telegram боту. Якщо увімкнений, користувач може викликати бота за допомогою набирання його нікнейму та запиту в текстовому полі повідомлення у будь-якому чаті. Запит надходить до боту під час отримання оновлень. Таким чином користувач може запрошувати деякий контент у бота в будь-де з його чатів, груп або каналів без надсилання жодних повідомлень [https://core.telegram.org/bots/inline]
* Інлайн кнопки – кнопки, що прикріплені до деякого повідомлення
* Шаблон – деяке збережене повідомлення користувача, яке може бути відтворено під час використання боту інлайн.
* Ключове слово – деяке слово-ключ за допомогою якого користувач зможе отримувати доступ до деякого шаблону.
  + 1. Перелік команд

Бот має обробляти наступні команди:

* /start
* /help
* /bind
* /unbind
* /list

На кожну з цих команд бот має дати видиму для користувача відповідь, яка має бути локалізована.

* + 1. Початок роботи

Під час первинного додавання бота усі користувачі повинні відправити боту команду /start (це зумовлено системою безпеки Telegram, що не дозволяє ботам писати користувачам першими для запобігання спаму від ботів). У цей момент бот надсилає стандартне вітання та не робить жодних записів у базі.

* + 1. Команда /start

При надсиланні користувачем команди /start бот повинен відправити стандартне вітання, яке має містити посилання на команду /help, при натисканні якого буде автоматично відправлена команда /help.

* + 1. Команда /help

При надсиланні користувачем команди /help бот повинен відправити більш детальну інформацію про існуючі команди та їх призначення.

Самі команди в повідомлені мають бути виділені.

Команди повинні дозволяти користувачу натискати на них задля їх виконання, окрім команд /bind та /unbind бо ці команди мають виконуватися з обов’язковими параметрами.

* + 1. Команда /list

При надсиланні користувачем команди /list бот повинен надіслати повідомлення з переліком усіх шаблонів користувача. Також команда має «таємний» параметр для користувачів, що наявні у колекції адміністрації, user\_id – індекс користувача з яким буде проводитися операція.

Перелік має бути зроблений за допомогою інлайн кнопок кнопки мають бути розташовані у 2 стовбця (якщо кількість непарна – остання кнопка займає 2 стовбця). На кожній кнопці має бути написане одне ключове слово. При натисканні кожна кнопка дає команду ботові надіслати шаблон відповідний до ключового слова на кнопці для перед перегляду.

У випадку, якщо користувач немає жодних збережених шаблонів, він має отримати відповідне повідомлення.

* + 1. Команда /bind

Команда /bind має обов’язковий параметр kwd – ключове слово, яке користувач буде використовувати для доступу до поточного шаблону. Допускається перелік ключових слів розділених комою.

Після ключового слова користувач може вводити будь-який текст з будь-яким форматуванням, яке має бути збережене при відтворенні.

При виконанні команди шаблон має бути доданий до колекції шаблонів та має бути зв’язаний з користувачем з колекції користувачів у базі даних. У випадку, якщо в колекції ще не існує запису користувача, вона повинна бути створена.

Після виконання команди має бути надісланий новий шаблон для перед перегляду або повідомлення про помилку.

* + 1. Команда /unbind

Команда /unbind має обов’язковий параметр kwd – ключове слово, шаблон відповідний якому має бути видалений. Допускається перелік ключових слів розділених комою. Також є «таємний» параметр для користувачів, що наявні у колекції адміністрації, user\_id – індекс користувача з яким буде проводитися операція.

При виконанні видаляє зв’язок користувача та шаблону, видаляє шаблон з колекції шаблонів, якщо жодне з інших ключових слів користувача не має зв’язку з цим шаблоном.

Після виконання команди має бути надіслано повідомлення про успішність операції або про помилку.

* + 1. Ключове слово

Ключове слово має містити лише букви та цифри, є нечутливим до регістру. З одним ключовим словом може бути зв’язаний лише один шаблон.

* + 1. Шаблон

Шаблон може містити форматований текст, файл будь-якого формату або деякий список файлів, таким чином підтримуючи усі типи повідомлень до яких можна додати текст.

Типи повідомлень, що мають підтримуватися:

* Текстові
* Зображення
* Відео
* Файл
* Аудіо
* Медіа група (сукупність файлів одного типу, зображення та відео можуть бути разом)

Текст після ключового слова є необов’язковим для всіх типів повідомлень окрім текстових.

* + 1. Збереження даних

База даних має містити наступні колекції:

* users – список користувачів
* keywords – список шаблонів
* admins – список індексів адміністраторів

Усі файли зберігаються на платформі Telegram, у базу записуються лише індивідуальні ідентифікатори файлів.

* + 1. Програмна частина

Увесь код має бути написаний на мові програмування Python з використанням бібліотеки для розробки Telegram ботів aiogram.

В якості бази даних має бути використана MongoDB.

* + 1. Локалізація

Увесь текст повідомлень має бути локалізованим.

Інформація про локалізацію має братися з повідомлення користувача.

Мови, що мають бути реалізовані:

* Англійська
* Російська

У майбутньому можливе додавання інших мов.

У випадку, якщо мови користувача немає у існуючому переліку, за замовчуванням має бути обрана англійська мова.

* + 1. Відмовостійкість

У випадку помилки на стороні користувача, бот має вказати користувачу на помилку

У випадку внутрішньої помилки, повна інформація про помилку має бути надіслана адміністрації

* + 1. Розгортання

База даних повинна бути доступна онлайн та зберігатися на серверах, що надаються самою MongoDB.

Скрипт програми повинен бути розгорнутий на сервері з підтримкою мови Python не нижче 3.8.

Кожен розробник повинен мати власно створеного бота для роботи. Забороняється використання основного боту.

Параметри підключення для бази даних та токен боту повинні зберігатися у змінних середовища, що відрізняються на сервері та локально.

* 1. Вибір засобів розробки

Для розробки Telegram боту використовувалося середовище розробки PyCharm 2020 від компанії JetBrains.

PyCharm – це спеціальне інтегроване середовище розробки Python (IDE), що забезпечує широкий спектр важливих інструментів для розробників Python, тісно інтегрованих для створення зручного середовища для продуктивного розвитку Python, Інтернету та науки про дані. [https://www.jetbrains.com/help/pycharm/quick-start-guide.html]

Для моделювання бази даних використовувався додаток MongoDB Compass

MongoDB Compass – графічний інтерфейс для роботи з MongoDB, що дозволяє робити всі базові операції CRUD, переглядати та оптимізувати запити та візуально досліджувати наявні дані. [https://www.mongodb.com/products/compass]

В якості мови програмування була використана мова Python.

Python – це інтерпретована, об’єктно-орієнтована мова програмування високого рівня з динамічною семантикою.

Для зручної роботи з Telegram API була використана спеціальна бібліотека для Python Aiogram.

Aiogram – це простий і повністю асинхронний фреймворк для API Telegram Bot, написаний на Python 3.7 з asyncio та aiohttp, що допомагає робити ботів простішими та швидшими.

* 1. Вибір моделі життєвого циклу

При розробці програмного продукту використовувалася інкрементна модель життєвого циклу, під час якої програмне забезпечення розробляється з лінійною послідовністю стадій однак в декілька інкрементів (версій). Таким чином покращення продукту проходить заплановано весь час, доки життєвий цикл розробки не завершиться.

Потреби до системи визначаються в самому початку роботи, після чого процес розробки проводиться у вигляді послідовності версій, кожна з яких являє собою закінчений працездатний продукт.

Ця модель є досить зручною тому, що спираючись на технічне завдання, є можливість створити деякий працездатний продукт на початкових етапах розробки.

Для розбиття завдань за пріоритетами використовувалася техніка Story Mapping.

Story Mapping – це метод організації користувацьких історій для створення більш цілісного уявлення про те, як вони вписуються у загальний користувацький досвід. [https://www.productplan.com/glossary/story-mapping]

Результат розбиття завдань за пріоритетами вказаний на рисунку 1.5.1.

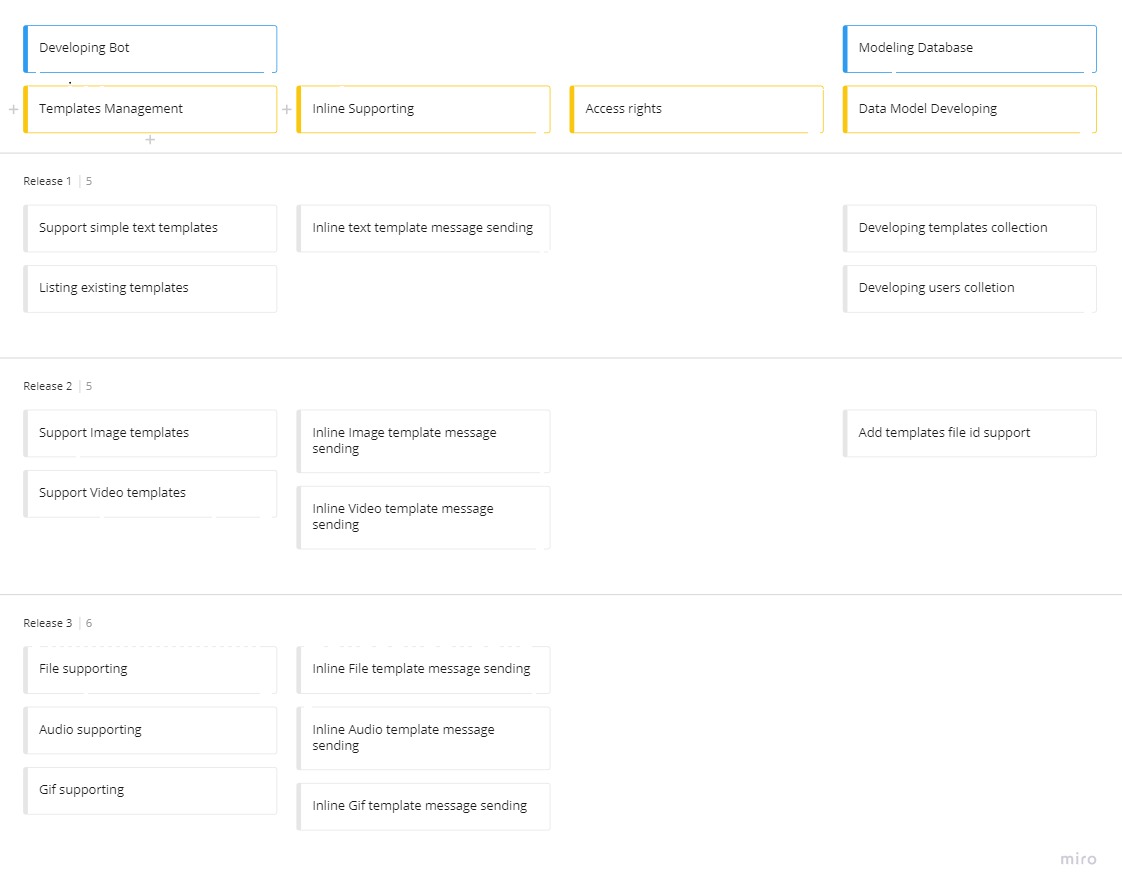


Рис. 1.5.1 – Пріоритетність задач за методом Story Mapping

Після виділення пріоритетності завдань, вони були розбиті на спринти за принципом 1 реліз – 1 спринт. Кожен спринт триває 2 тижні. Для розбиття цих завдань та подальшої роботи з ними був використаний веб-додаток Clubhouse. Який дозволяє зв’язати дошку з деяким git репозиторієм та контролювати прогрес деякої історії шляхом зміни статусів обраної гілки. Приклад дошки з Clubhouse продемонстровано на рисунку 1.5.2

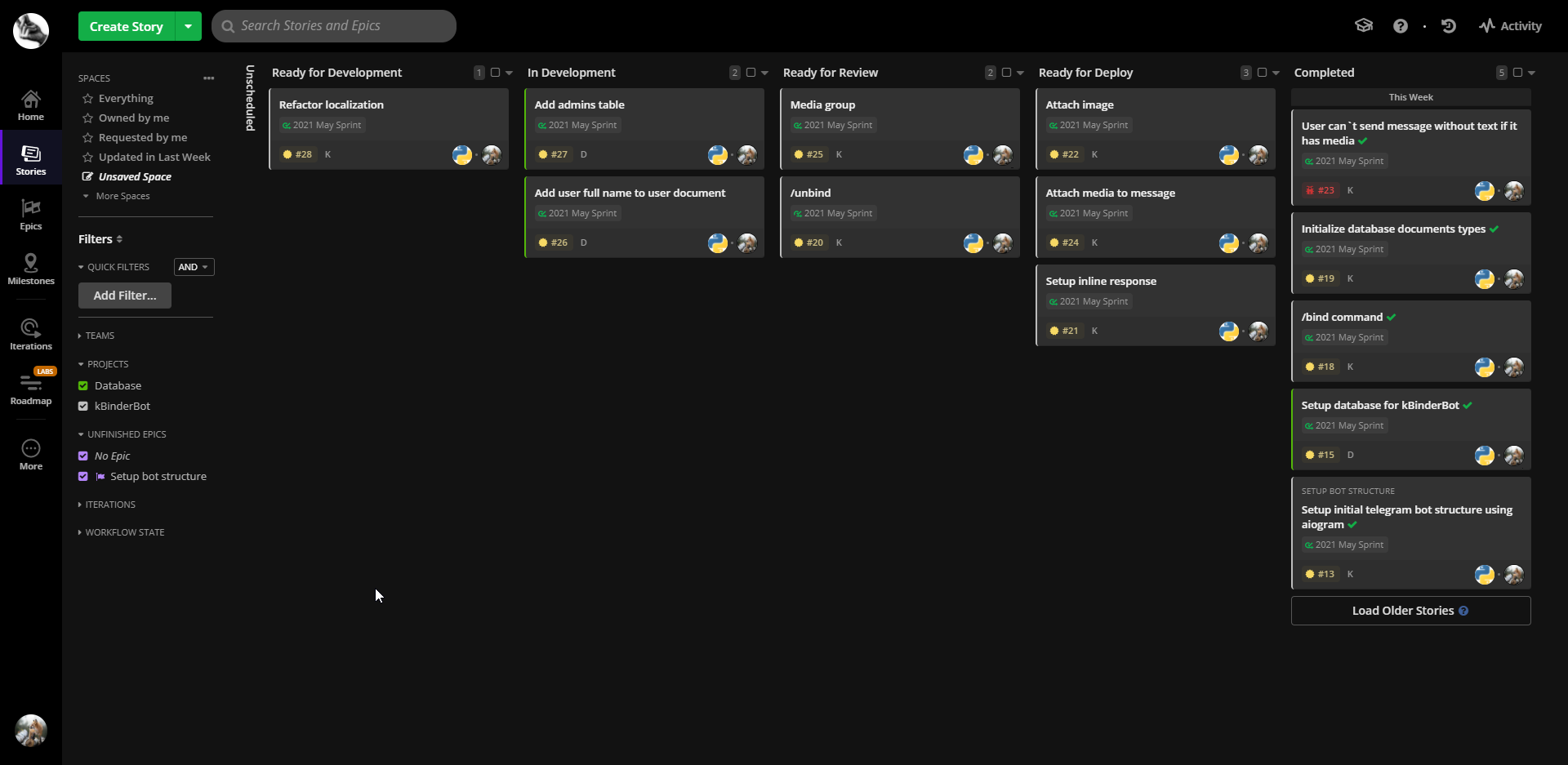


Рис. 1.5.2 – дошка Clubhouse

Для початку роботи з дошкою треба обрати завдання та знайти на ньому поле, у якому зазначена спеціальна назва гілки завдяки якій сервіс буде відстежувати статус історії (рис. 1.5.3).

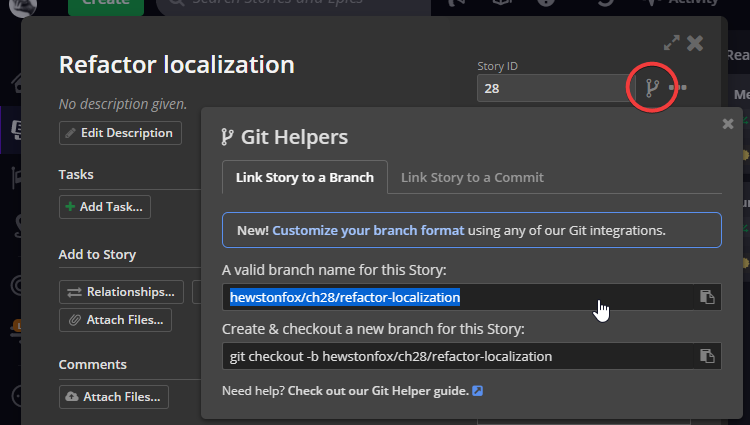


Рис. 1.5.3 – Отримання корекної назви гілки

Статус історії визначається за правилами нижче:

* Одразу після створення гілки історія переходить із статусу «Готова до розробки» у статус «У розробці».
* Коли від гілки поступає запит на злиття з гілкою dev історія переходить у статус «Готова для перевірки»
* Після злиття цієї гілки з dev історія переходить у статус «Готова для розгортання»
* Після злиття цієї гілки з гілкою master, історія переходить у статус «Виконано»