Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій

Кафедра програмних систем і технологій

УДК *004.9* *На правах рукопису*

ВИПУСКНА КВАЛІФІКАЦІЙНА БАКАЛАВРСЬКА РОБОТА

Тема: «Розробка віртуального асистенту для автоматизації процесу найму на роботу»

Спеціальність – 121 “Інженерія програмного забезпечення”

ПОЯСНЮВАЛЬНА ЗАПИСКА

Студент

ІПЗ-42_____/Олександра ГОЛОВАН/

Науковий керівник

к.ф.м.н.,доц._____/ Сергій ПОЛЯКОВ/

Консультант

з питань нормоконтролю

фахівець_____/ Тамара ЧАПОВСЬКА/

Допускається до захисту

Завідувач кафедри

д.т.н.,проф._____ /Олексій БИЧКОВ/

Київ – 2021

Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій

Кафедра програмних систем і технологій

Освітньо-кваліфікаційний рівень бакалавр

Спеціальність 121 “Інженерія програмного забезпечення”

ЗАТВЕРДЖЕНО

Зав. кафедри програмних систем і технологій

_____ (Олексій БИЧКОВ)

(підпис) **(прізвище та ініціали)**

ЗАВДАННЯ
НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ

Голован Олександрі Олегівні

(прізвище, ім’я, по-батькові)

1. Тема бакалаврської роботи “Розробка віртуального асистенту для автоматизації процесу найму на роботі ” **,** керівник проекту (роботи) Поляков Серній Анатолійович к. ф. м. н.**,** доцент _____ затверджені наказом вищого навчального закладу від **“** ___ ” _____ 2021 р. **№** _____

2. Строк подання студентом роботи _____ 2021 р.

3. Вихідні дані до проекту (роботи) Теоретичні концепції та моделі побудови віртуального асистенту

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

1. Опис проблеми та способи її вирішення

2. Обрана середа розробки

3. Створення чат-боту

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

1. Список доступниз для створення об'єктів у Corezoid (рис. 1.4.2, ст. 19)
2. Діграма станів «User Profile» (рис. 2.3.1, ст. 29)
3. Процес «Router» (рис. 2.3.3, ст. 31)
4. Процес «Send Message» (рис. 2.3.4, ст. 32)

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Створення чат-боту	Поляков Сергій Анатолійович		

7. Дата видачі завдання _____ 2021 р.

Керівник _____ (Сергій ПОЛЯКОВ)

Завдання прийняв до виконання _____ (Олександра ГОЛОВАН)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назви етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Збір та вивчення літератури	21.01.21	виконано
2	Визначення функціональних можливостей майбутнього ПЗ	14.02.21	виконано
3	Вивчення інформації про API, які будуть використовуватися	04.03.21	виконано
4	Розробка ПЗ	28.04.21	виконано
5	Опис процесу розробки ПЗ	18.05.21	виконано

Студент – бакалавр _____ (Олександра ГОЛОВАН)

Керівник роботи _____ (Сергій ПОЛЯКОВ)

АНОТАЦІЯ УКРАЇНСЬКОЮ МОВОЮ

Випускна кваліфікаційна бакалаврська робота: 69 с., 25 рис., 1 таблиця, 4 додатки

Прізвище та ініціали студента: Голован О.О.

Тема: Розробка віртуального асистенту для автоматизації процесу найму на роботу

Спеціальність (шифр та назва): 121 «Інженерія програмного забезпечення»

Установа, де відбудеться захист: Факультет інформаційних технологій

Місто: Київ

Рік: 2021

Предмет дослідження: спрощення процесу найму на роботу.

Результати дослідження: було створено чат-бот для меседжеру Telegram, де користувач матиме змогу надсилати своє резюме і проходити співбесіду. Окрім цього, за допомогою чат-боту можна буде переглядати товари та замовляти їх через кошик.

Ключові слова: ЧАТБОТ, COREZOID, TELEGRAM, ПРИЙОМ НА РОБОТУ.

АНОТАЦІЯ РОСІЙСЬКОЮ МОВОЮ

Выпускная квалифицированная бакалаврская работа: 69 с., 25 рис., 1 таблица, 4 дополнения

Фамилия и инициалы студента: Голован А. О.

Тема: Разработка виртуального ассистента для автоматизации процесса приема на работу

Специальность (шифр и название): 121 «Инженерия программного обеспечения»

Учреждение, где состоится защита: Факультет информационных технологий

Город: Киев

Год: 2021

Предмет исследования: упрощение процеса прийома на работу.

Результаты исследования: был создан чат-бот для мессседжера Telegram, где пользователь имеет возможность отправлять свое резюме и проходить собеседование. Помимо этого, с помощью чат-бота возможно просматривать товари и заказывать их.

Ключевые слова: ЧАТ-БОТ, COREZOID, TELEGRAM, ПРИЕМ НА РАБОТУ.

АНОТАЦІЯ АНГЛІЙСЬКОЮ МОВОЮ

Graduation qualified bachelor's work: 69 pages, 25 pictures, 1 table, 4 appendices

Student name: Sasha Holovan

Work name: Development of virtual assistant for hiring process automation

Specialty (code and name): 121 «Program Engineering»

Institution of work defensing: Faculty of Information Technologies

City: Kyiv

Year: 2021

Subject of research: simplification of the recruitment process.

The results of the research: creation of the chatbot for the Telegram messenger, where user's has the opportunity to send their resume and pass an interview. In addition, with the help of a chatbot it is possible to view products and make an order.

Keywords: CHATBOT, COREZOID, TELEGRAM, HIRING

ЗМІСТ

ПЕРЕЛІК ОСНОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ 10

ВСТУП 11

РОЗДІЛ 1

Теоретична частина

1.1 Основні відомості про меседжери. Обрані меседжери для роботи 12

1.2 Що таке чат-боти та їх можливості 13

1.3 Способи створення чат-ботів 15

1.4 Обрана мова програмування Corezoid. Інтерфейс. Вбудовані логіки 16

1.5 JavaScript. Взаємозв’язок з Corezoid 25

РОЗДІЛ 2

Практична частина

2.1 Створення чат-боту для працевлаштування 26

2.2 Функціонал чат-боту 27

2.3 Архітектура проекту 28

2.4 Процеси 33

2.4.1 Процес «/catalogueOfGoods» 34

2.4.2 Процес «/jobOffers» 37

2.4.3 Процес «/basket» 39

2.4.4 Процес «/viewTheResumes» 41

2.4.5 Процес «Zoom Meeting» 44

2.4.6 Процес «DialogFlow» 46

ВИСНОВКИ 48

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 50

ДОДАТОК А 51

ДОДАТОК Б 54

ДОДАТОК В 56

ДОДАТОК Г 57

ПЕРЕЛІК ОСНОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ

ОС	–	Операційна система
Лоукод (з англ. low-code)	–	Візуальне програмування, яке зазвичай потребує мінімальне написання коду за допомогою вбудованих конструкцій коду (наприклад, умовна конструкція)
Вузол (нода)	–	Універсальний обчислювальний елемент, який описує стан об'єктів, може мати такі характеристики, як: лічильник, логіка, черга тощо
Браузер	–	Програма за допомогою якої можливо шукати та переглядати інформацію в Інтернеті;
БД	–	База даних
Меседжер	–	Програмне забезпечення, яке дозволяє спілкуватися в реальному часі за допомогою інтернету, технології Bluetooth або між пристроями які підключені між собою локально
API (повна назва «Application Programming Interface»	–	Інтерфейс за яким виконується взаємозв’язок між двома програмами
Інтеграція	–	Процес об'єднання даних з усіх різних джерел та їх представлення в єдиному виді
ПЗ	–	Програмне забезпечення
Заявка	–	Набір постпаючих параметрів типу JSON у визиваємий процес
JS	–	JavaScript
Кросплатформність	–	Здатність програми працювати на приятроях з різними операційними системами та виду

ВСТУП

В наш час використання таких програмних засобів, як меседжери зростає з кожним днем. Спілкування з далекими родичами, друзями, створення нових знайомств або навіть корпоративне спілкування – все це можна робити на своєму пристрою у реальному часі. Але що, якщо я скажу Вам, що це далеко не всі можливості сучасних меседжерів?

Автоматизація звичних для нас процесів, наприклад, замовлення їжі або бронювання квитків до кінотеатру, вже не є чимось новим для нас усіх та виконується за декілька натисків у додатку. Але нащо встановлювати багато різних програм до свого пристрою або весь час відкривати десятки сторінок у браузері, якщо це все можна робити у меседжері, де люди зазвичай проводять більшість свого часу при використанні гаджетів, за допомогою чат-ботів.

Однією з досі не вирішених проблем в наш час є процес прийняття на роботу. Перед тим, як нарешті отримати омріяну посаду, майбутньому співробітникові треба витратити досить багато часу на: співбесіди, відправку резюме та заповнення анкет, підпис документів та багато іншого. Важко не погодитися, що це не завжди зручно і взагалі не приносить нам жодного задоволення. Я вважаю, що було б доречно створити щось інноваційне з цього плану, а саме створити електронного асистенту, за допомогою якого кожен бажаючий зміг би проходити всі ці етапи влаштування на робоче місце прямо у себе на телефоні, не виходячи при цьому з власної домівки.

РОЗДІЛ 1
ТЕОРЕТИЧНА ЧАСТИНА

1.1 Основні відомості про меседжери. Обрані меседжери для роботи

Насправді, меседжери виникли ще дуже давно у вигляді SMS-листування але все зараз вони мають набагато більшу популярність та функціонал. По-перше, вони не потребують додаткових витрат за кожне повідомлення, яке ви надсилаєте; по-друге, через те, що зазвичай вони потребують зв’язок з Інтернетом, листування здійснюється в реальному часі, відповідь від свого співбесідника можна отримати буквально менше ніж за секунду, як воно було надіслано вам; по-третє, меседжери підтримують надсилання файлів будь-яких типів, не тільки фото або відео, а також аудіо файлів, архівів тощо; по-четверте, за допомогою використання чат-ботів, функціонал звичайного меседжеру можна зростити у десятки разів.

Отже, як біло сказано вище, меседжери існують вже давно але таку стрімку популярність вони почали набирати тільки зараз. За даними сайту minfin.com.ua, найпопулярнішими меседжерами в Україні за даними станом на травень 2020 рокі, є наступні:

- Viber;
- Facebook Messenger;
- Telegram;
- WhatsApp;
- Skype.

В рамках цієї роботи я буду працювати з таким меседжером, як Telegram. Такий вибір я можу аргументувати тим, що це є один із найпопулярніших меседжерів, а також, на відміну від Facebook Messenger та Viber, які за мають друге та третє місце по популярності, він не потребує проходження верифікації при створенні свого чат-боту. Telegram має дуже детальну документацію, підтримує оплату з Apple Pay та Google Pay, відправку точки на карті, альбомів фото та відео файлів та багато іншого функціоналу, який може бути використан у чат-боті.

1.2 Що таке чат-боти та їх можливості

Чат-бот – програма, яка імітує розмову з реальною людиною та має орієнтований для користувача інтерфейс. Зазвичай спілкування з такими програмами відбувається за допомогою вбудованих команд, написання тексту, спеціальної клавіатури з вшитими під кнопками командами або навіть голосових повідомлень. Чат-ботів можна зустріти переважно у меседжерах. Зараз їх функціонал майже не поступається функціоналу повноцінної програми але не потребує встановлення на пристрій, тим самим не займає зайвого об’єму пам’яті, не потребує швидкого Інтернету та постійного підключення до нього. Окрім цього, не можна не сказати про те, що підтримка таких програм займає набагато менше ресурсів та не потребує багато часу для створення.

Функціонал чат-ботів дуже багатий. До основних можливостей можна віднести:

- відповідь на питання користувачів;
- збір та зберігання даних користувачів (наприклад, опитування);
- розсилка новин користувачам або інших повідомлень;
- навчання;
- відправка повідомлень на електрону пошту або на мобільний пристрій;
- покупка товарів та послуг.

Зазвичай, найрозповсюдженішими чат-ботами є ті, за допомогою яких можна замовити їжу (див. рис. 1.2.1), купити квитки, переглянути новини, погоду або іншу інформацію.

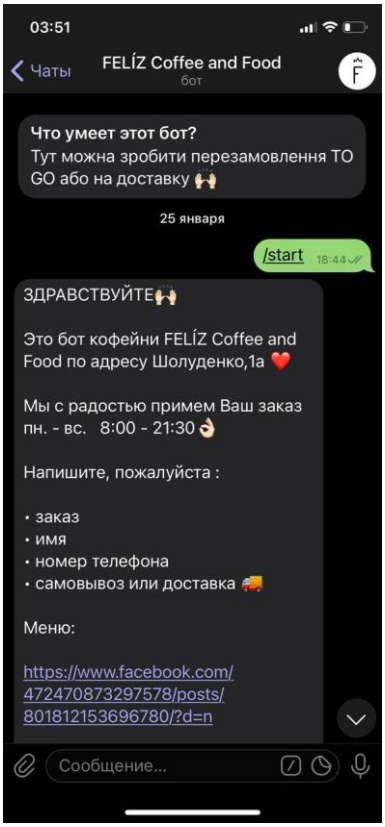


Рис 1.2.1. Чат-бот закладу «FELIZ». За допомогою нього можна легко замовити їжу та доставку

Більшість функціональних можливостей виконується за допомогою інтеграцій з різноманітними API. Серед популярних сервісів, які частіше всього використовуються для інтеграцій, можна виділити наступні:

– Google Sheets – запис даних до таблиць;
– eSputnik – відправка SMS-повідомлень, а також до електронної пошти;
– БД – запис інформації даних до різних баз даних, наприклад: MongoDB.

1.3 Способи створення чат-ботів

Як і будь-яка програма, чат-боти також створюються за допомогою різних мов програмування та інших інтрументів. Для їх створення підійде будь-яка мова програмування але для більшості це будуть саме:

– Python;
– Java;
– PHP;
– Node.js.

Останнім часом набуває популярність створення та підтримка чат-ботів на мовах лоу-коду. До таких мов можна віднести: Corezoid та Node-RED. Цей спосіб створення є менш популярним у світі але дуже затребуваний на території України. Наприклад, більшість банків використовує саме цю технологію.

Самою великою перевагою використання саме лоу-коду є візуальна розробка (див. рис. 1.3.1). З його допомогою можна вирішувати прості задачі з мінімальним знанням основ програмування, що робить процес створення більш швидким, легшим та більш дешевшим. Також, через його візуальне відображення дуже легко відстежувати шлях користувачів ПЗ у програмі.

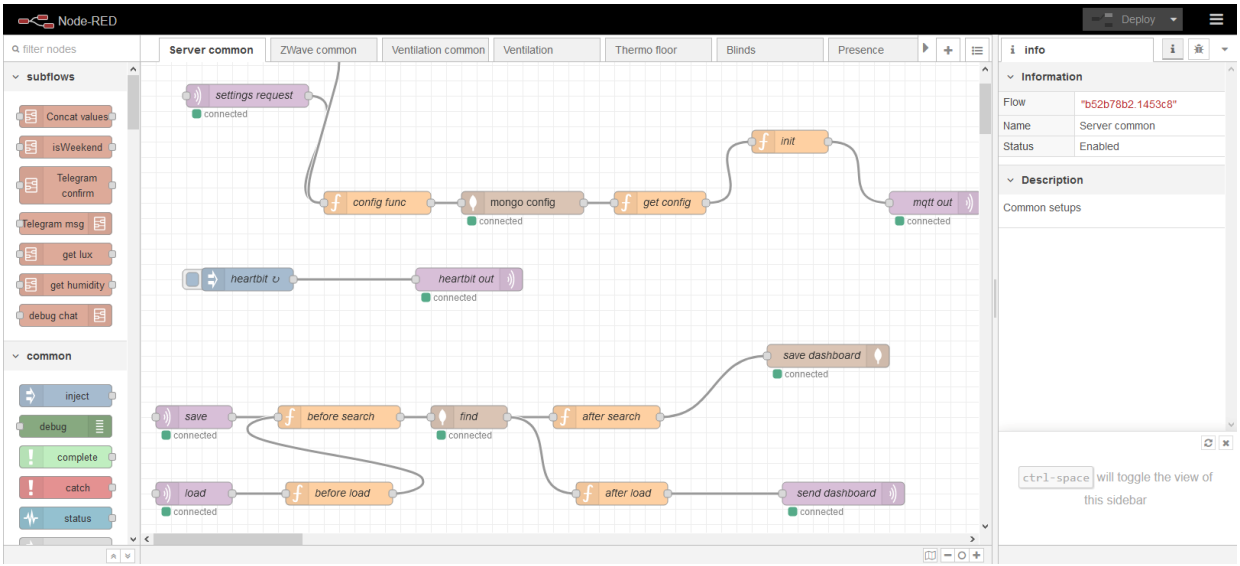


Рис 1.3.1. Середа розробки на мові Node-RED. Інтерфейс

1.4 Обрана мова програмування Corezoid. Інтерфейс. Вбудовані логіки

Corezoid (див. рис. 1.4.1) – це універсальна метамова, яка дозволяє швидко створювати нові і управляти існуючими API в єдиному інтерфейсі. Сам Corezoid було написано на мові Erlang. Використовує базу даних PostgreSQL. Технологія не є безкоштовною але усім бажаючим надається пробний період у розмірі 30 днів. Програмування виконується у вікні браузеру на офіційному сайті Corezoid, що робить цей інструмент кросплатформним. Можливе використання мови JavaScript або Erlang для розширення базового функціоналу створення ПЗ.

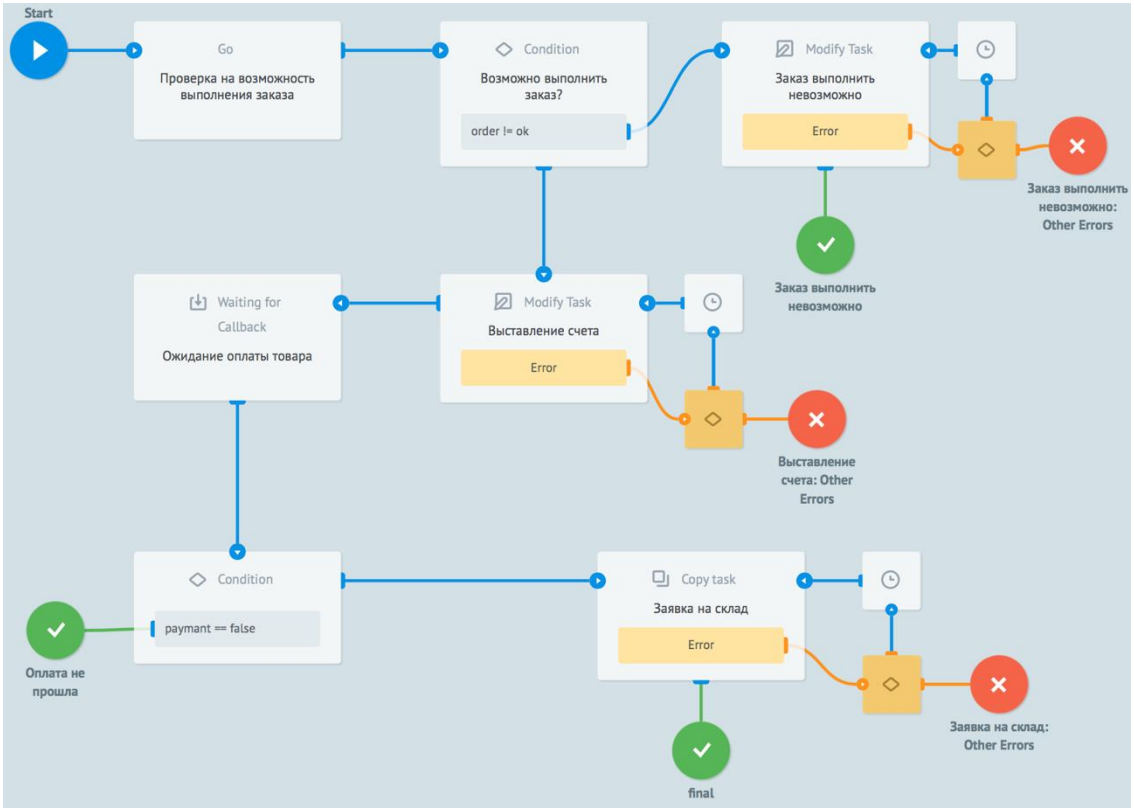


Рис 1.4.1. Інтерфейс процесу на Corezoid. Логіка замовлення товару

Corezoid з’явився на початку 2013 році. Розроблявся він командою розробників з компанії PrivatBank. Пізніше, команда відділилась та продовжила вдосконалювати свій продукт вже під керівництвом новоутвореної компанії Middleware. Технологія постійно оновлюється та вдосконалюється. Остання версія продукту наразі 5.6.1.

Corezoid ще тільки набирає свою популярність але не дивлячись на це, вже зараз його використовують деякі закордонні компанії. До списку компаній, що використовують цю технологію, входять наступні:

- VISA;
- MascterCard;
- METRO;
- D.Tєк;
- ПУМБ;
- Банк ВОСТОК;
- Таскомбанк;
- Rakuten Viber;
- PrivatBank.

За допомогою мови Corezoid можливе створення мобільних додатків, веб-застосунків, чат-ботів та багато іншого. Серед переваг можна виділити наступне: легкість в освоєнні, не потребує поглиблених знань у сфері програмування; можливість реалізації, як простих задач так і складних; зручне створення бізнес логіки; цілодобова підтримка; документація не тільки англійською, а також російською мовою; можливість отримання офіційного сертифікату (Junior, Middle, Senior) володіння мовою; кросплатформність; зручний та зрозумілий інтерфейс; можливість використання мови JavaScript або Erlang для збільшення функціональних можливостей мови; влаштована БД; не потребує встановлення до свого пристрою; можливість експорту даних у csv форматі (Excel або Google Sheets); можливість перегляду, на якому кроці «завис» користувач.

В кожному проекті у Corezoid можливо створити 7 типів (див. рис. 1.4.2):

- Folder (з англ. Folder – папка) – каталог всередені проекту;
- Process (з англ. Process – процес) (див. рис. 1.4.3) – деяка програмка логіка;
- The state diagram (з англ. The state diagram – діаграма станів) (див. рис. 1.4.4) – аналог БД, там можуть зберігатися заявки;

– Dashboard (з англ. Dashboard – панель, дошка) (див. рис. 1.4.5) – дошка з діаграмами, данні до яких можна обирати, аналогічне є у, наприклад, Microsoft Excel;
– Bot Platform (з англ. Bot Platform – бот платформа) – стандартна чат-бот платформа Corezoid, яка підтримує прийом повідомлень від користувача, оброблює їх, відправляє йому повідомлення у відповідь, має стандартний набір шаблонів для клавіатур, стандартну локалізацію;
– Database (з англ. Database – база даних) – підтримує підключення бази даних;
– From file (з англ. From file – з файлу) – підтримує імпорт об’єктів з файлу типу JSON.

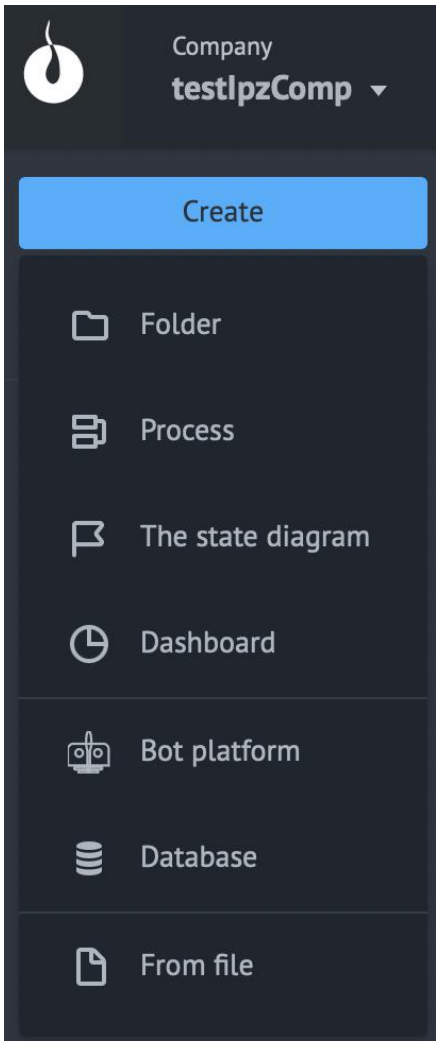


Рис 1.4.2. Створення об’єктів у проекті Corezoid

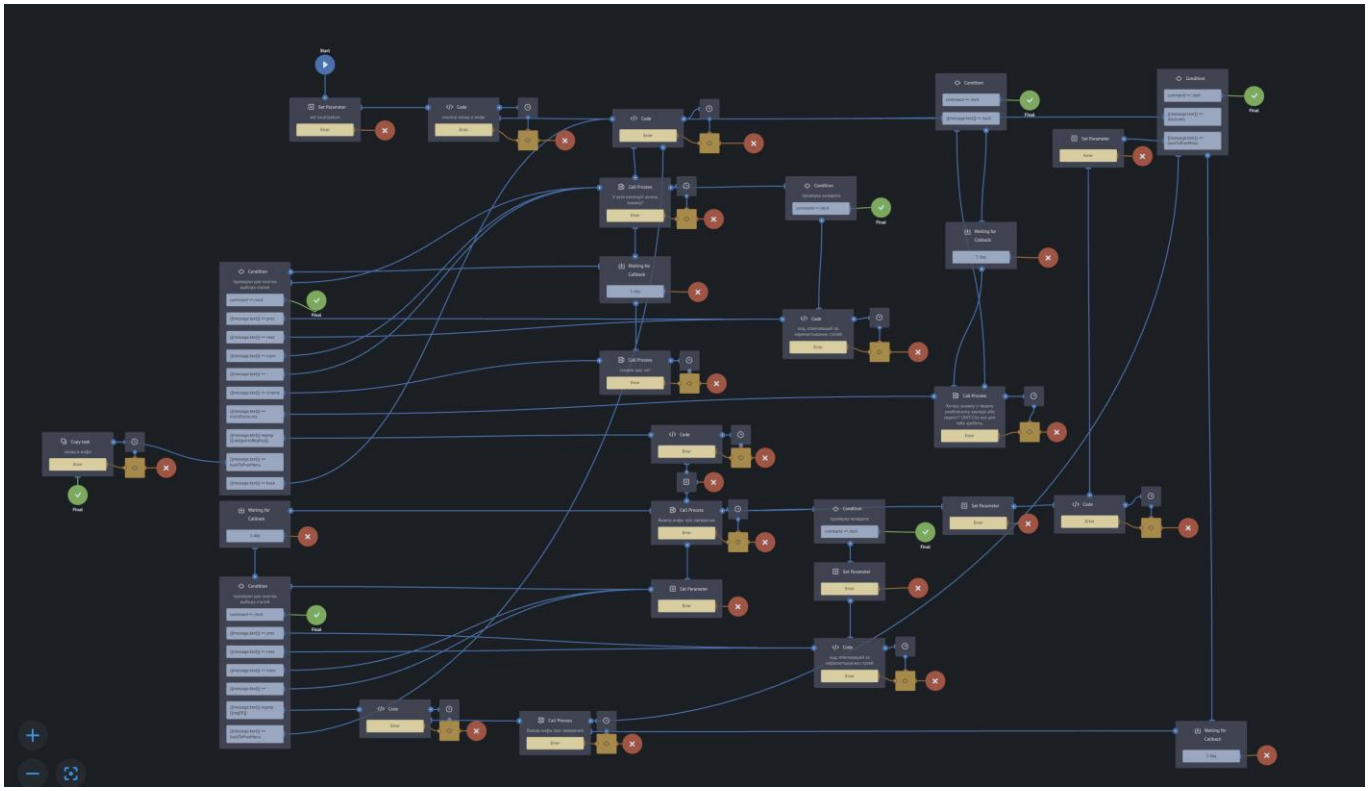


Рис 1.4.3. Процес Corezoid

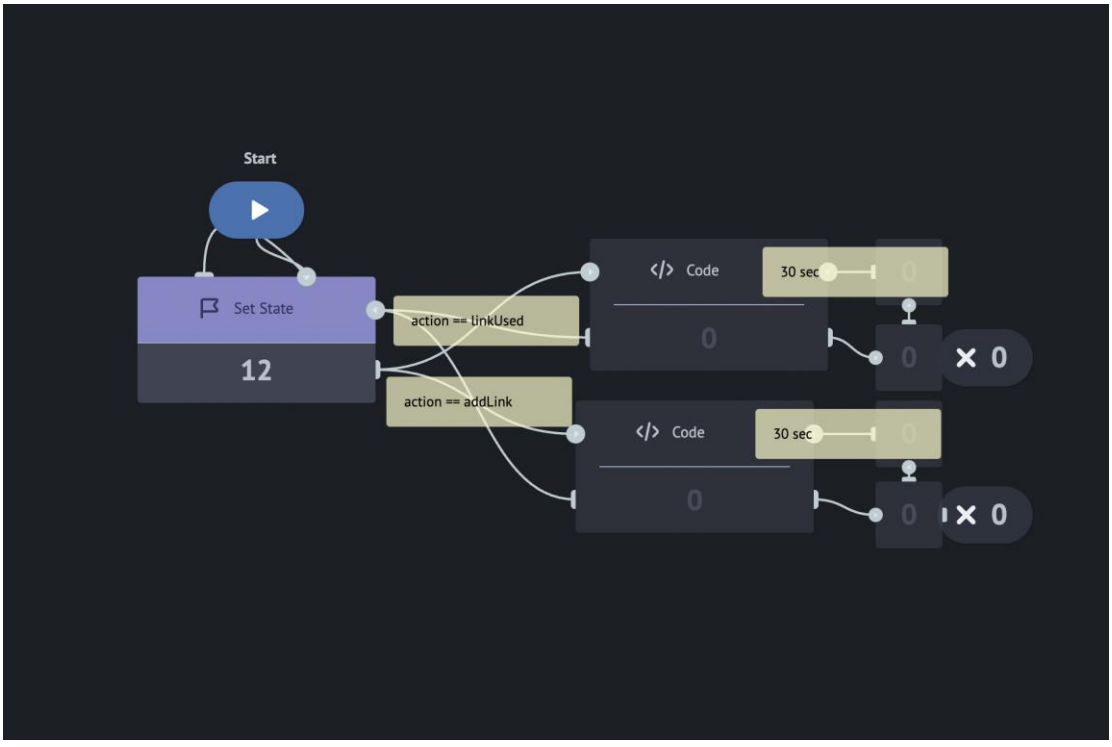


Рис 1.4.4. Діграма станів Corezoid

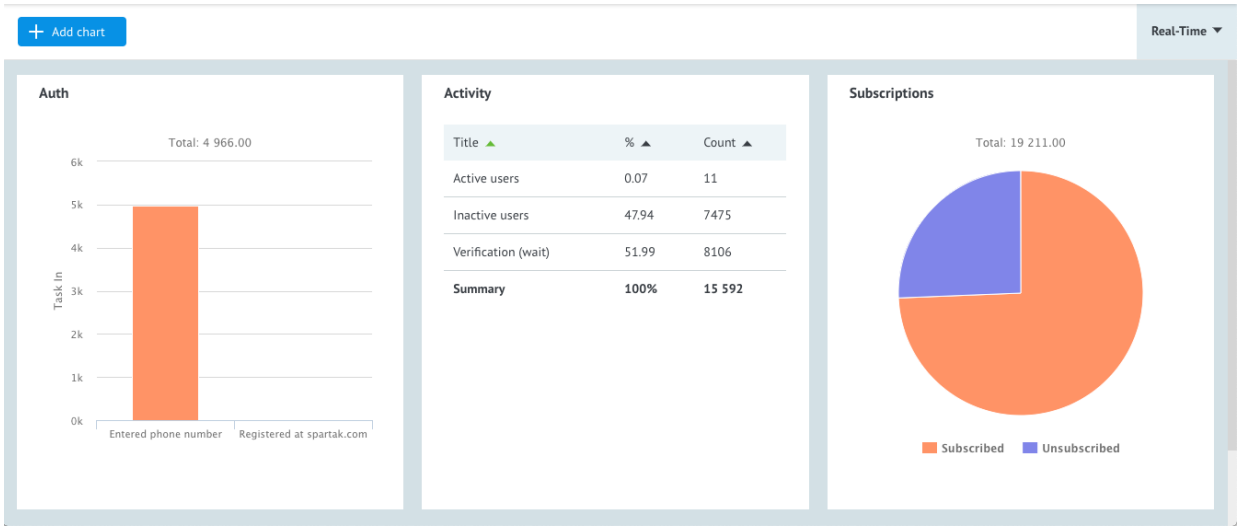


Рис 1.4.5. Діаграми (дашборди) Corezoid

Найчастіше, при розробці деякого проекту, створюються об’єкти двох типів: процес та діаграма станів.

Процес:

- має доступ до повного набору вбудованих логік;
- має обмеження на максимальну кількість заявок в одному процесі – 100000 заявок.

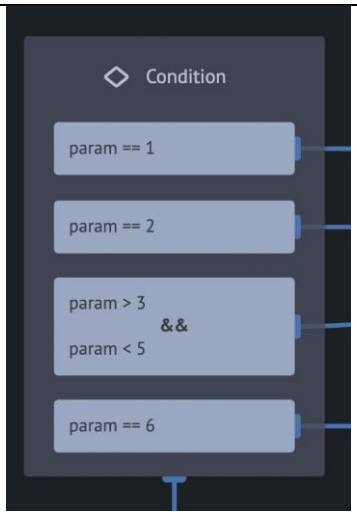
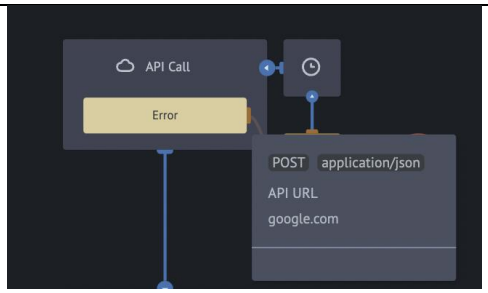
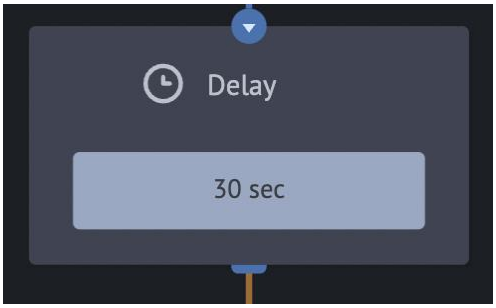
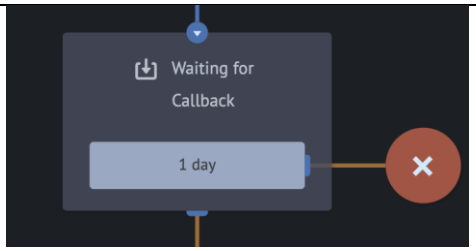
Діаграма станів:

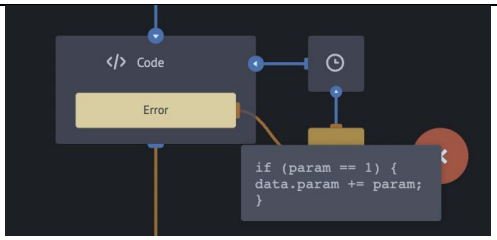
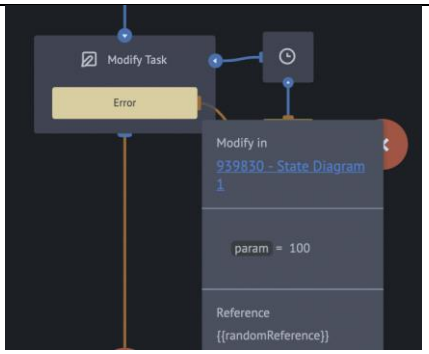
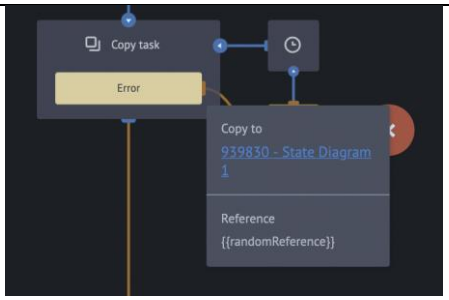
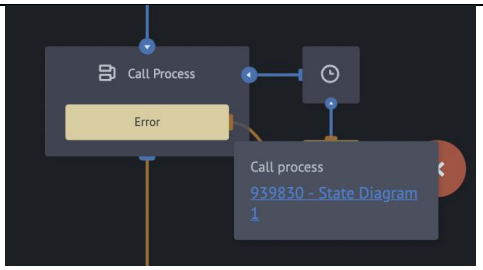
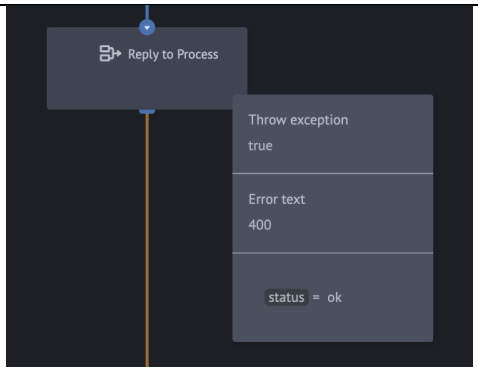
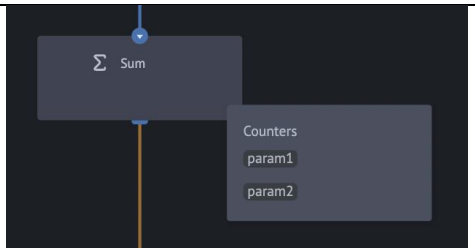
- зберігання даних об’єктів у вузлах;
- отримання даних об’єктів за допомогою спеціальних функцій;
- доступна обмежена кількість логік;
- немає обмежень на кількість заявок у процесі.

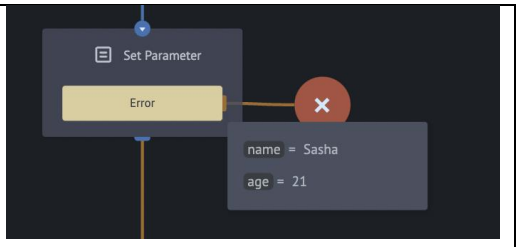
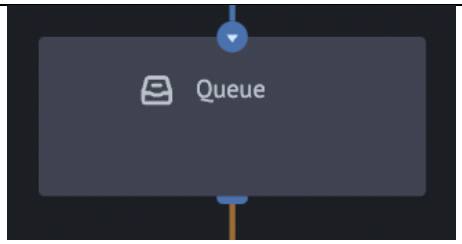
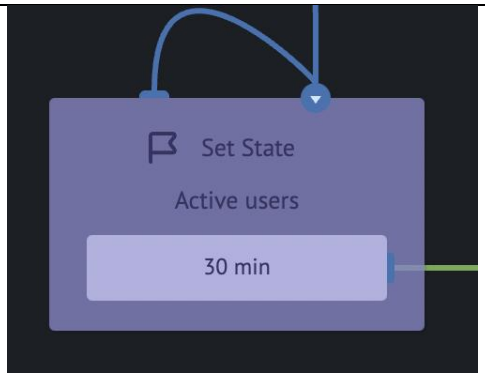
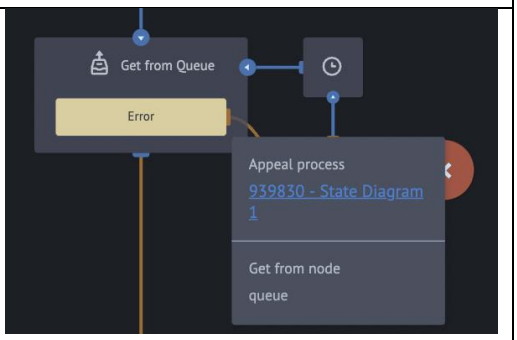
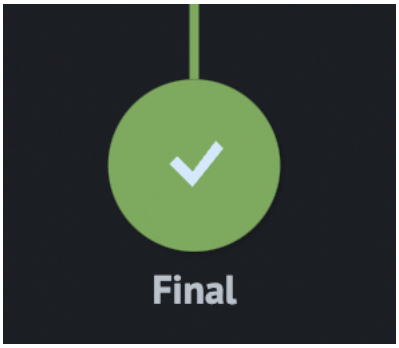
Як ми бачимо, процес – це деяка логіка, через яку проходить заявка, діаграма станів – аналог БД, де можна зберігати заявки, які надходять до процесів та згодом брати з них інформацію або навпаки, модифікувати дані. Яскравим прикладом є стандартні діаграми станів Localization та Attachments. У першій зберігається заявка з усіма текстами, які черпає програма, щось схоже є у інших мовах програмування. Друга зберігає у собі заявки з клавіатурами для користувачів.

Наразі Corezoid підтримує 16 вбудованих логічних вузлів. Повний список з описом кожної з логік можна знайти нижче у таблиці 4.1.

Таблиця 1.4.1. Список доступних логік

Назва логіки	Опис	Вигляд
Condition	Умовна логіка. Підтримує умови: =, !=, >, <, regEX.	
API Call	Виклик API (можливо вказати параметри для запиту).	
Delay	Затримка заявки на деякий визначений час. Після скінчення таймеру, заявка піде дальше по процесу або діаграмі станів.	
Waiting For Callback	Очікування на відповідь від зовнішньої системи.	

Code	Додавання додаткової логіки на мовах JavaScript та Erlang.	
Modify Task	Зміна (оновлення) заявки в іншому процесу по її назві (унікального ідентифікатора).	
Copy Task	Копіювання заявки до іншого процесу.	
Call Process	Виклик універсального процесу.	
Reply To Process	Відповідь на виклик універсального процесу.	
Sum	Сумування деяких значень заявки.	

Set Parameter	Установка параметрів, зміна значення параметрів у заявках.	
Queue	Логіка черги.	
Set State	Логіка зберігання заявок, аналог БД.	
Get From Queue	Вилучення заявок з черги.	
End: Success	Кінцевий вузол, який зберігає успішні заявки процесу, носить інформативний характер. Заявки в ньому зберігаються деякий час, після чого видаляються та їх не можна переглянути.	

End: Error	Кінцевий вузол, який зберігає заявки з помилками, носить інформативний характер. Заявки в ньому зберігаються деякий час, після чого видаляються та їх не можна переглянути.	

1.5 JavaScript. Взаємозв’язок з Corezoid

JavaScript – об’єктно орієнтована мова програмування, яку ми можемо зустріти найчастіше всього у вікні свого браузеру. Кожного разу, коли ми відкриваємо сторінку у браузері, натискаємо на кнопку, бачимо анімацію – це все робота саме JavaScript. До сильних сторін цієї мови можна віднести:

- простота мови, дуже легка для розуміння;
- постійні оновлення;
- можливість розроблювати на цій мові навіть у звичайному текстовому редакторі та за допомогою браузеру;
- багатий функціонал, який можливо використати на власний веб-застосунках.

JavaScript було створено у 1992 році однією американською компанією. Не зважаючи на префікс «Java», JS не має жодного відношення до цієї сови програмування. Справа в тому, що спочатку ця мова мала назву «LiveScript» але під впливом популярної на той час мови Java, було вирішено перейменувати LiveScript на JavaScript, як посилання на те, що це є молодшим братом мови Java.

Corezoid та JS дуже пов’язані між собою. По-перше, саме JS ввикористовується для розширення базової логіки та написання більш складних функцій, які не передбачені стандартною платформою. По-друге, заявки до Corezoid надходять саме у JSON, текстовому форматі, який було засновано на JS. В наступних підрозділах буде продемонстровано приклади використання мови JS у середовищі Corezoid.

РОЗДІЛ 2

ПРАКТИЧНА ЧАСТИНА

2.1 Створення чат-боту для працевлаштування

Темою моєї дипломної роботи є створення чат-боту для спрощення процесу найму на роботу. В наш час, коли майже усі звичні нам речі можна зробити прямо у своєму смартфоні, для звичайної співбесіди нам все одно потрібно йти до офісу майбутньої компанії, стояти у чергах, когось чекати. Іноді, такий процес може зайняти не один день або навіть тиждень. Саме через це було вирішено створити систему працевлаштування, яка не потребує виходити з власного будинку.

Було переглянуто багато варіантів реалізації такої проблеми: веб-додатки, ПЗ на смартфон або інший гаджет, чат-бот. Згідно до 1 розділу, чат-бот є найактуальнішим та найзручнішим вирішенням даної задачі. По-перше, для процесу працевлаштування нам не потрібно буде встановлювати зайве ПЗ на свій пристрій, а також переходити на різноманітні веб-сторінки у пошуках необхідної нам інформації, все буде «під рукою» – у меседжері де, зазвичай, люди проводять більшість свого часу за гаджетом. По-друге, процес створення такого додатку є більш швидким, простим та легким у підтримці, з такою задачею може справитися навіть одна людина.

Чат-бот створюватиметься для філій магазинів, які надають користувачам високий асортимент товарів для придбання. Для компанії, яка володіє такими магазинами, підтримка такого чат-боту є гарним рішенням, бо це:

- сприяє зростанню популярності до своїх магазинів за рахунок такого віртуального асистенту;
- зменшує штаб, адже для найму людей онлайн, не потрібно буде утримувати зайвих менеджерів на кожній з філій, легше буде набрати значно меншу кількість у вигляді онлайн-операторів;
- більш дешева підтримка такого виду продукту.

2.2 Функціонал чат-боту

Чат-бот матиме наступний функціонал:

- каталог товарів за категоріями;
- додавання товарів до кошику;
- замовлення товарів з кошику;
- видалення товарів з кошику;
- перегляд доступних вакансій;
- надсилання свого резюме на обрану вакансію;
- співбесіда у чат-боті за допомогою інтегрованих сервісів;
- «живе» спілкування;
- відображення актуальний акцій та пропозицій;
- розсилка акцій усім користувачам.

Було вирішено, що для коректної реалізації представленого вище функціоналу, у чат-боті буде використовуватися рольова модель, адже для кожної ролі буде виділено свій набір доступних функцій. В цілому, асистент матиме 2 ролі, а саме:

- користувач – будь-хто, хто вперше зайшов до бота та ще не пройшов реєстрацію. Матиме наступний функціонал:
    - перегляд товарів за категоріями;
    - перегляд знижок та акцій;
    - перегляд доступних магазинів;
    - додання товарів до кошику;
    - видалення товарів з кошику;
    - замовлення товарів;
    - відправка резюме на роботу.
- оператор – той, хто перевіряє документи від майбутнього співробітника та проводить співбесіду. Має наступні можливості у чат-боті:
    - перегляд анкет та резюме;
    - співбесіда.

2.3 Архітектура проекту

Коли користувач вперше відкриває чат-бот, заявка з його даними попадає у процес ресівер (з англ. receiver – одержувач). Таких процесів може бути декілька, в залежності від того, скільки меседжерів підтримується чат-ботом. Наприклад:

– Telegram Receiver;
– Viber Receiver;
– Facebook Receiver.

Ці процеси дивляться, що саме чат-бот отримав від користувача: текст, документ, контакт, зображення, відео, та інші об’єкти, які можна тільки надіслати у діалогове вікно.

Після цього, заявка йде до головного процесу (з англ. main – головний). Тут ми дивимося на змінну event (з англ. event – подія), яка може примати такі значення як: start, message, command. Незалежно від значення цієї змінної, ми намагаємося створити профіль користувача (з англ. user profile – профіль користувача) у діаграмі станів Users Profiles (див. рис. 2.3.1), де зберігаються профілі усіх користувачів, які хоч раз починали діалог з чат-ботом. Кожна заявка з користувачем у діаграмі зберігається за назвою channel_chatId (див. рис. 2.3.2), що є назва меседжеру, з якого від користується чат-ботом, та унікальний код користувача.

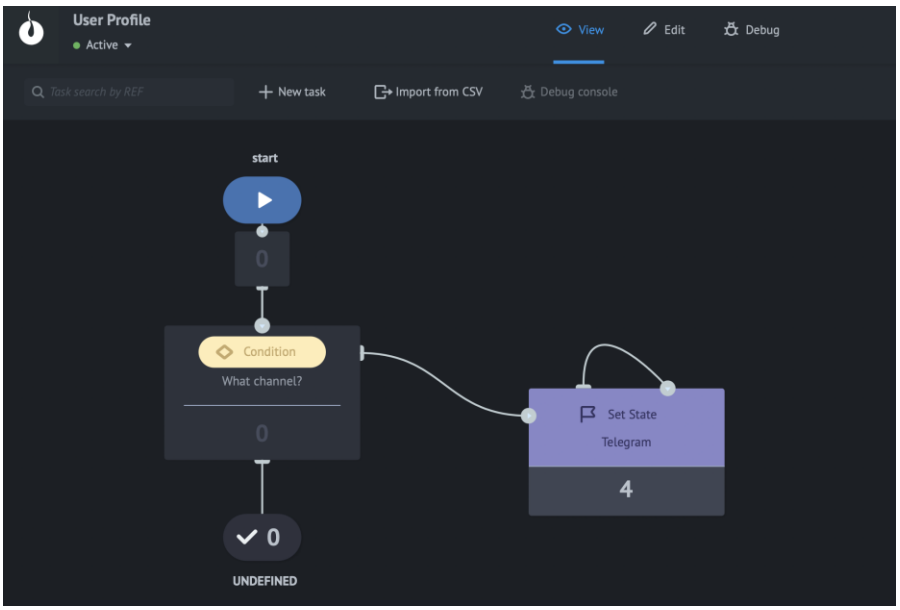


Рис 2.3.1. Діаграма станів «User Profile», яка зберігає 4 користувачів чат-боту

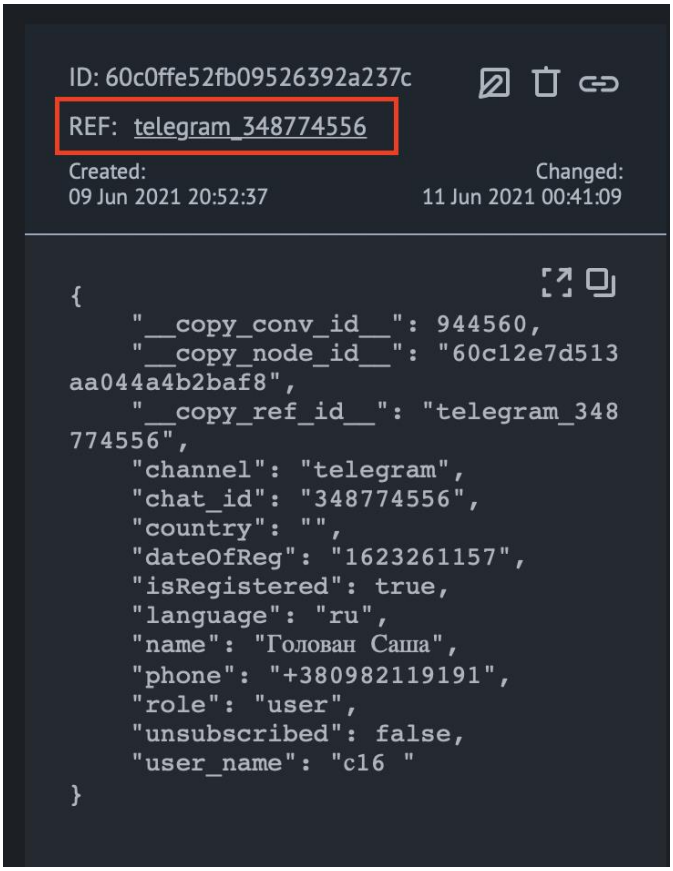


Рис 2.3.2. Приклад назви заявки у діаграмі станів «User Profile»

Третій процес, до якого надходить заявка користувача – роутер (з англ. router – маршрутизатор) (див. рис. 2.3.3). Як вже зрозуміло з назви – цей процес дивиться, до якого далі процесу користувача слід перенаправляти. Паралельно з цим, він перевіряє, чи можливе використання такої команди у поточній ролі користувача. Після цього користувач нарешті попадає до того процесу, до якого він звертався (наприклад, він натиснув кнопку «кошик» якій еквівалентна команда /basket, роутер дивиться на назву команди та перенаправляє користувача до цього процесу).

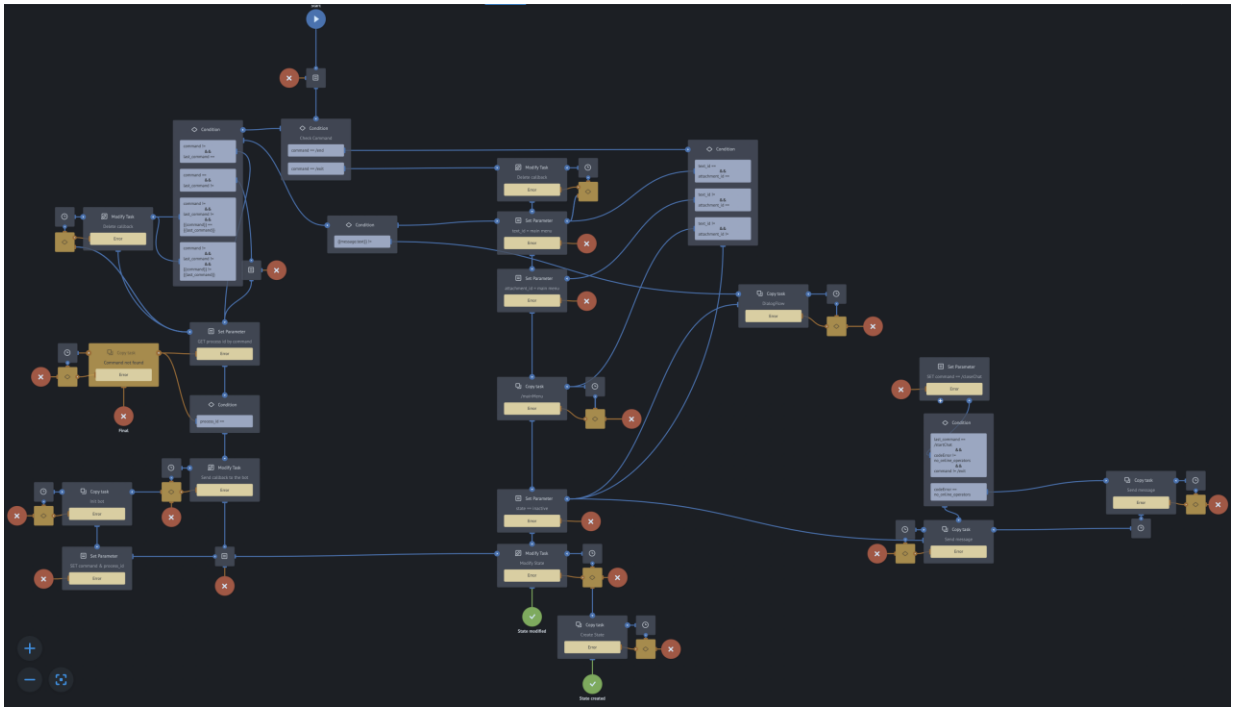


Рис 2.3.3. Процес «Router»

Коли з командою було визначено і користувач потрапив до потрібного процесу, ми повинні відправити йому належний текст та клавіатуру (за необхідністю). Для цього було створено процес відправки повідомлень (з англ. send message – відправка повідомлення) (див. рис. 2.3.4). Цей процес приймає на вхід обов'язково наступні параметри:

- chat_id – ідентифікатор користувача у боті, так ми будемо знати кому відправляти повідомлення;
- channel – канал, з якого звертається користувача, там ми будемо знати звідки прийшов користувач;
- text_id – ідентифікатор тексту, так ми будемо знати, який саме текст відправляти (про це трохи пізніше). Може бути замінений на параметр text;
- text – текст повідомлення, яке має бути надіслане;
- attachment_id – ідентифікатор клавіатури, яка буде відображатися користувачеві.

– localization – номер діаграмі станів, з якої потрібно буде брати текст за ключем, визначеним у text_id. Без цього параметру буде обрана стандартна діаграма з локалізацією.

Сам процес визначається з мовою інтерфейсу користувача, аби потім відправити потрібний текст з, формує клавіатуру (наприклад, можна зробити її динамічною), якщо коротко – процес який повністю відповідає за інформацію, що ми відправляємо користувачу.

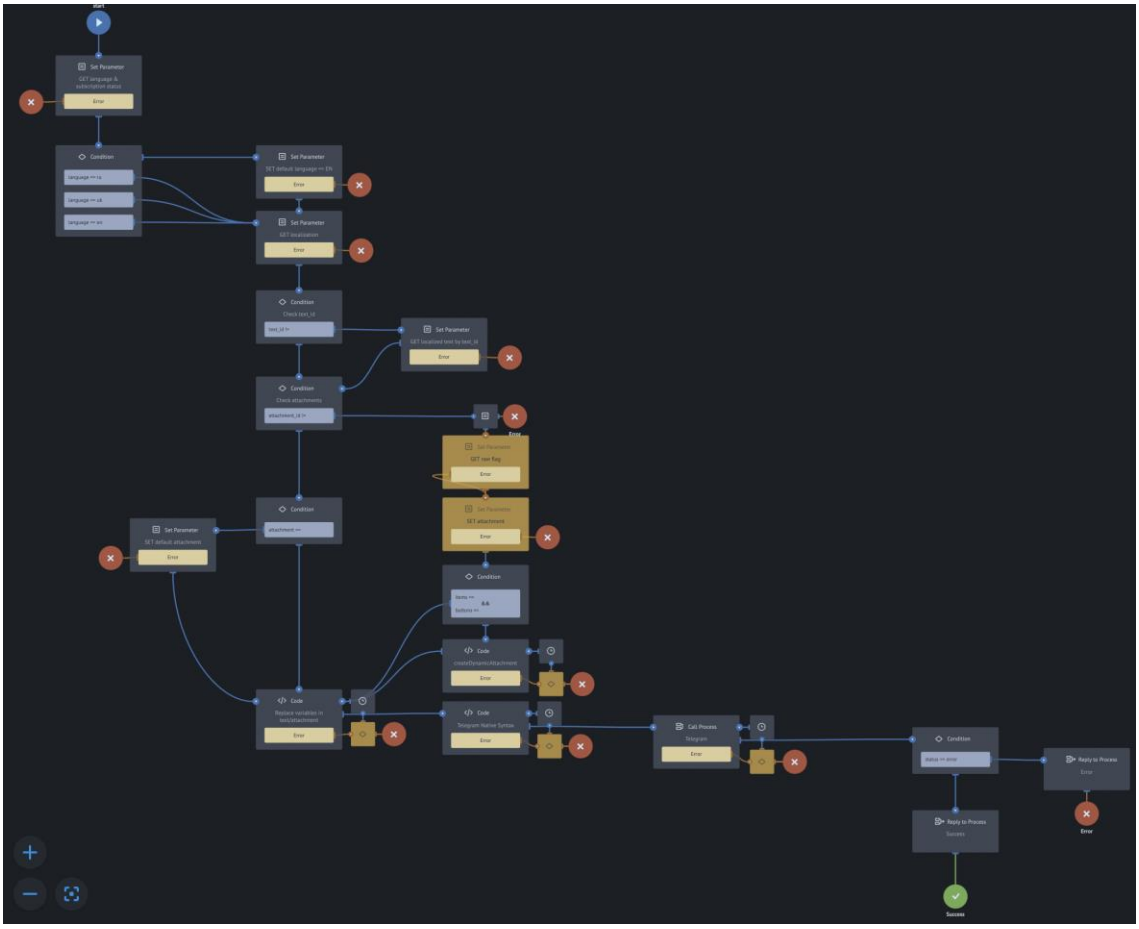


Рис 2.3.4. Процес «Send Message»

Зазвичай, текст повідомлення можна вказати на вхід до процесу у параметрі «text» але це не є гарним рішенням. По-перше, це не є зручним. Якщо нам потрібно буде змінити текст, ми мусимо йди до конкретного процесу, з якого цей текст надходить, шукати потрібний вузол та змінювати текст прямо там. По-друге, якщо бот буде підтримувати більше однієї мови, це буде дуже складно реалізувати. Саме

через це зазвичай використовують таку діаграму станів, як «Localization» (з англ. localization – локалізація). Ця діаграма є базовою, та містить у собі лише одну заявку з масивом об’єктів – текстами (див. рис. 2.3.5).

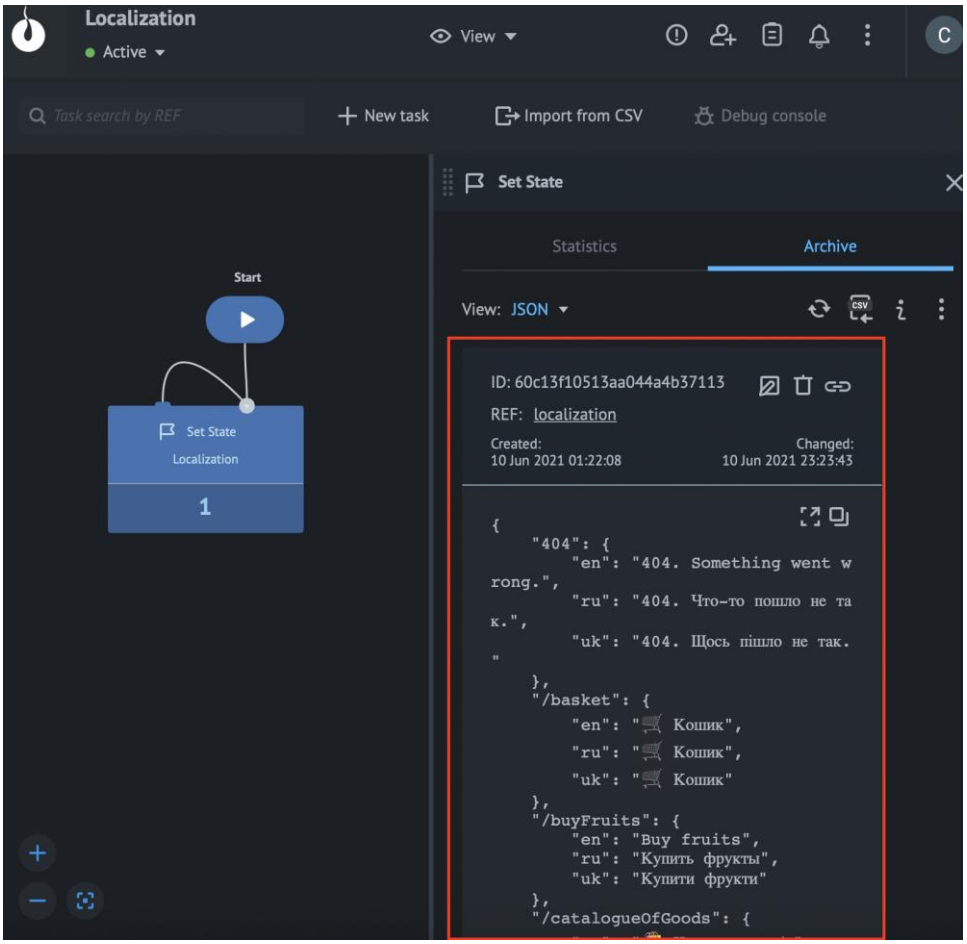


Рис 2.3.5. Заявка з локалізацією чат-боту у процесі «Localization»

Аби надіслати потрібну клавіатуру користувачеві, Corezoid дивиться на параметр у attachment_id та йде до стандартної діаграми attachments (з англ. attachments – вкладення). В цій діаграмі станів знаходяться заявки з клавіатурами.

2.4 Процеси

Процеси – деякі окремі модулі проекту. Це можуть бути процеси з викликом API та його обробкою, процеси, які надсилають повідомлення користувачам, проще кажучи – щось схоже з поняттям «класи» у об’єктно-орієнтованому програмуванні. Наприклад, в нас є процес «інформація», який виводить користувачеві клавіатуру з кнопками: «новини», «погода», «гороскоп». В даному випадку ми маємо щонайменше 4 процеси: «інформація», куди буде потрапляти користувач та ще 3 процеси згідно до кнопок на клавіатурі, до яких ми будемо надсилати нашу заявку. В цих підпроцесах можуть у свою чергу звертатися до інших процесів, наприклад, процес з погодою звертається до процесу з викликом API погоди, куди ми передаємо місто користувача. Основними процесами в проекті, що розробляється, є:

- /catalogueOfGoods – відображення каталогу товарів;
- /jobOffers – доступні вакансії та надсилання свого резюме;
- /discounts – знижки та акційні пропозиції;
- /basket – кошик з товарами, замовлення та видалення товарів з кошику;
- /shopLocation – локації доступних магазинів;
- /viewTheResumes – керування резюме користувачів: відхилення або призначення їм співбесіди;
- /mainMenu – відображення клавіатури згідно в залежності від ролі користувача;
- Zoom Meeting – формування посилання-запрошення до конференції сервісу «Zoom»;
- dialogFlow – інтеграція з API DialogFlow, для «живого» спілкування з чат-ботом.

2.4.1 Процес «/catalogueOfGoods»

Процес «catalogurOfGoods» відповідає за відображення каталогу товарів користувачеві (див. рис. 2.4.1.1).

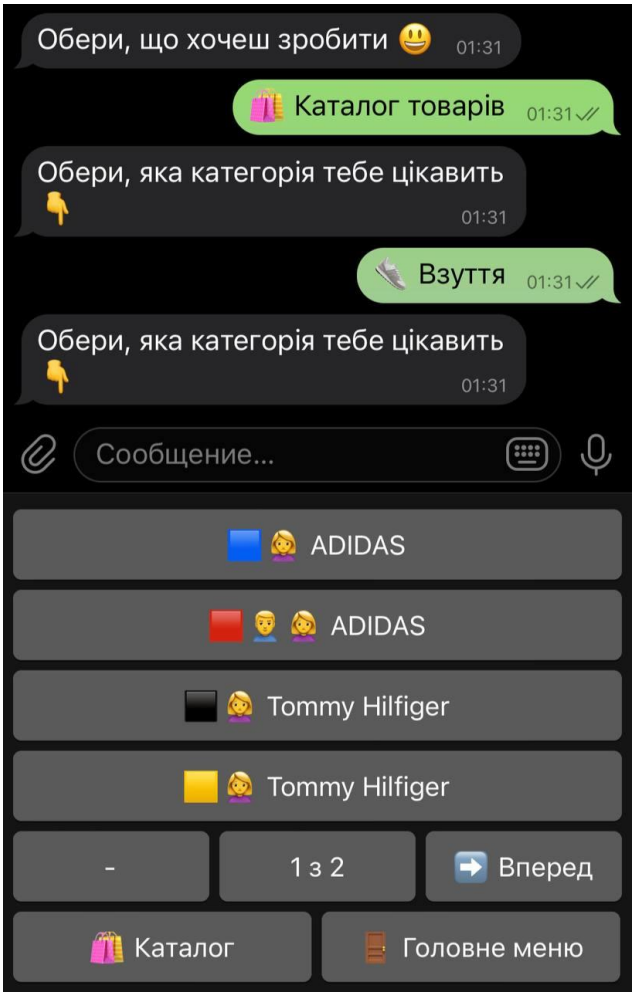


Рис 2.4.1.1. Каталог товарів

Для зручності у перегляді товарів, була використана динамічна клавіатура з пагінацією (див. Додаток А).

Спочатку, ми виводимо користувачеві усі можливі категорії товарів. Після вибору, ми формуємо динамічну клавіатуру, в яку передаємо такі параметри, як:

– items – масив об’єктів, в яких знаходяться кнопки товарів;
– maxItemsCountToShow – максимальна кількість кнопок з товарами на сторінці;
– currentPage – поточна сторінка.

При натиску кнопок «Вперед» або «Назад», ми збільшуємо або зменшуємо параметр «currentPage» на 1 та виводимо наступний список кнопок на сторінку. В залежності

від обраної категорії, ми дістаємо масив з кнопками з діаграми станів «Goods» (див. рис. 2.4.1.2).

Рис 2.4.1.2. Масив кнопок товарів з категорії «Взуття»

Після натиску на необхідний товар, наприклад, на кнопку «bootsAdidas», ми додаємо до назви кнопки слово «Info» та отримуємо «bootsAdidasInfo» – опис товару, який ми дістаємо з локалізації. Таке рішення було прийнято для оптимізації процесу і його динамічності. Таким чином, нам не потрібно було додавати вузли «Condition» для кожної можливої кнопки що значно зменшило розмір процесу (див. рис. 2.4.1.3).

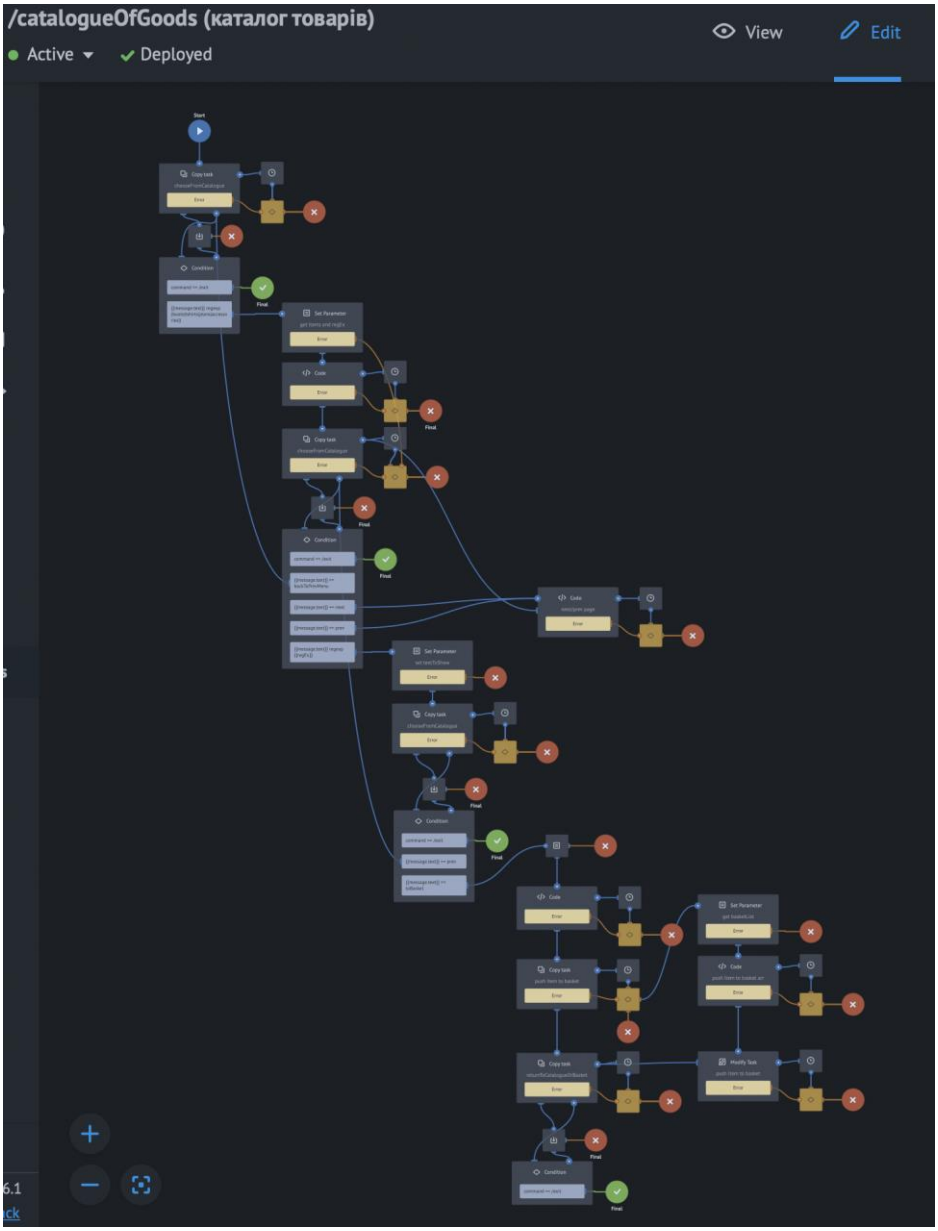


Рис 2.4.1.3. Процес «catalogueOfGoods»

2.4.2 Процес «/jobOffers»

Процес «/jobOffers» відповідає за відображення користувачеві списку доступних вакансій та надсилання резюме та інших файлів від користувача до оператора на співбесіду (див. рис. 2.4.2.1, рис. 2.4.2.2).

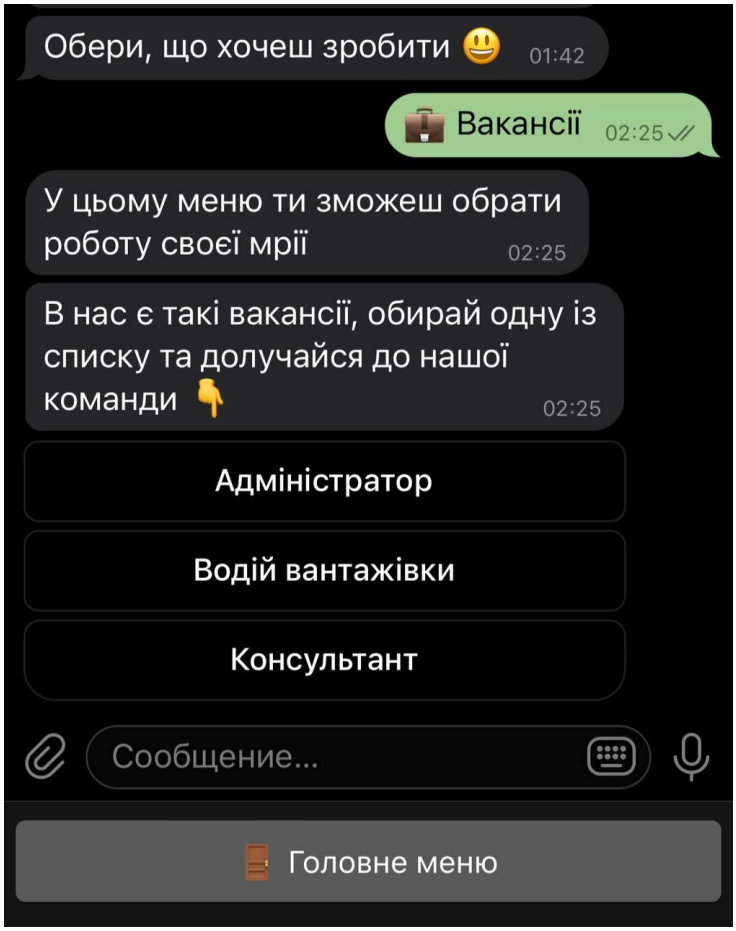


Рис 2.4.2.1. Вибір доступних вакансій зі списку

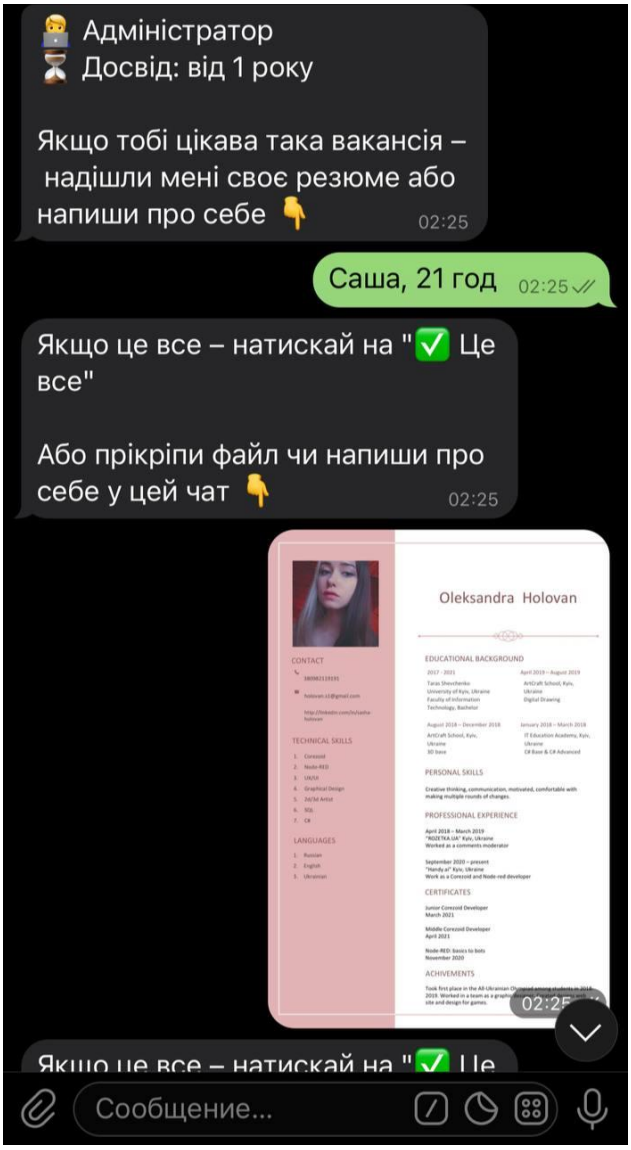


Рис 2.4.2.2. Надсилання свого резюме та іншої інформації до оператору

Після вибору доступних вакансій, користувач може написати дещо про себе, надіслати будь-який файл та навіть аудіо чи відео повідомлення (вбудована функція меседжеру Telegram). Всю цю інформацію ми зберігаємо до масивів «aboutYourself» та «resumeData» – текстова інформація про користувача та прикріпленні файли відповідно (див. Додаток Б).

Коли користувач закінчить заповнювати інформацію про себе, ми перевіряємо, чи зареєстрований він у нашій базі (у випадку, коли він вже подавав своє резюме раніше), якщо ні – просимо його вказати номер телефону та своє ім’я. Наступним кроком буде зберігання даних користувача до діаграми станів «Candidates». В цій даграмі станів ми зберігаємо одну заявку, в якій знаходиться масив з користувачами.

2.4.3 Процес «/basket»

Процес «/basket» відповідає за відображення користувачеві товарів у кошику та можливість їх придбання або видалення зі списку (див. рис. 2.4.3.1 та рис. 2.4.3.2).

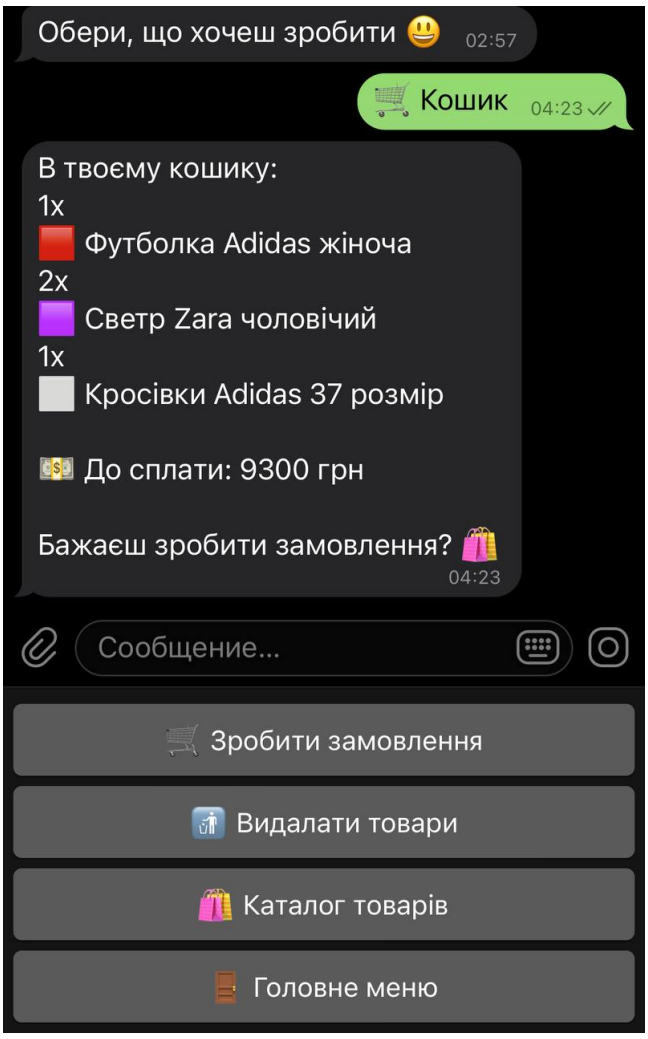


Рис 2.4.3.1. Кошик з товарами

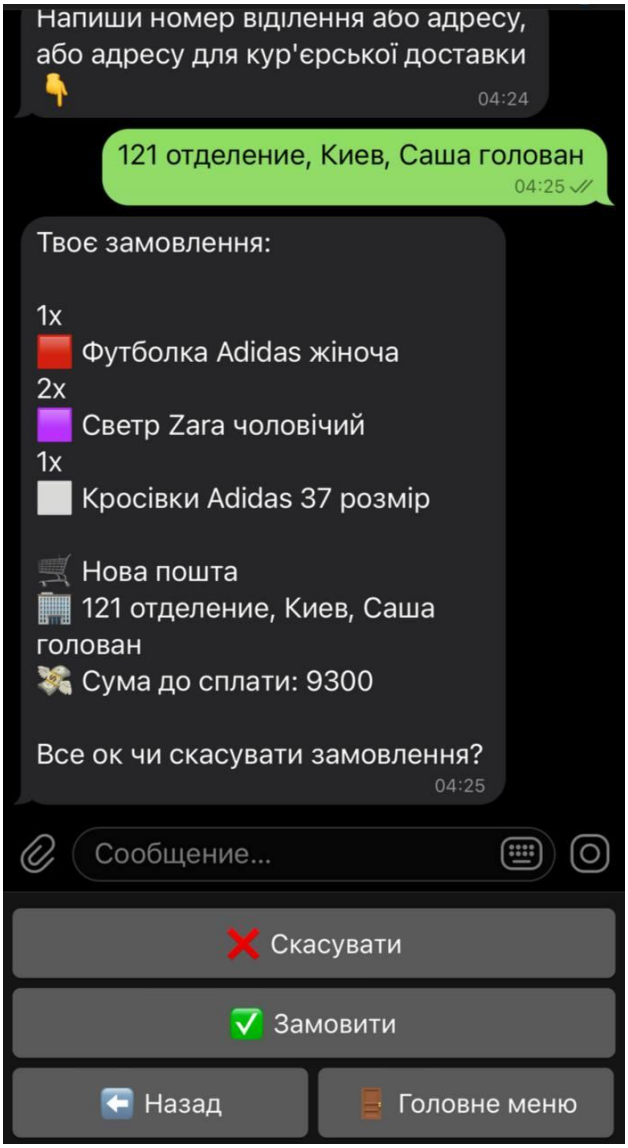


Рис 2.4.3.2. Оформлення замовлення на пошту

Коли користувач попадає до цього процесу, ми перевіряємо, чи є в його кошику товари. У випадку, якщо кошик пустий – пропонуємо йому перейти до каталогу товарів та щось туди добавити. Якщо в кошику вже є товари, ми так само пропонуємо йому кнопку «Каталог товарів», а також кнопки «Зробити замовлення» та «Видалити товари». При натисканні на першу кнопку, користувач може обрати спосіб отримання товару (поштою або обрати магазин зі списку) та сказати необхідну для цього інформацію. Друга кнопка пропонує користувачеві видалити якусь позицію зі списку або очистити весь список.

2.4.4 Процес «/viewTheResumes»

Процес «/viewTheResumes» відповідає за перегляд надісланих до оператора резюме (див. рис. 2.4.4.1). Доступ до цього процесу має лише оператор.

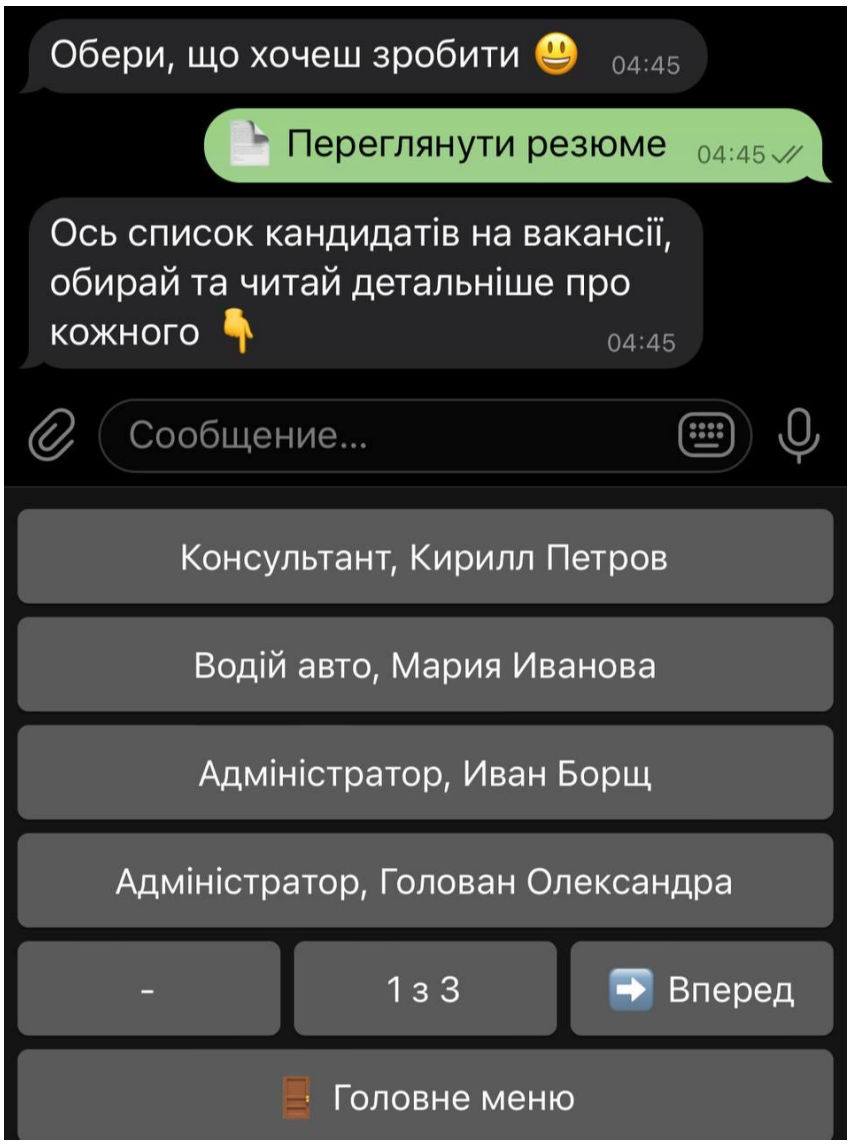


Рис 2.4.4.1. Перегляд кандидатів на відкриті вакансії

При переході до цього процесу, ми дістаємо масив з усіма кандидатами з діаграми станів «Candidates» та перевіряємо, чи не пустий він. Якщо в ньому все-таки немає кандидатів – надсилаємо оператору повідомлення, що поки що ніхто не надав своє резюме. Якщо він не порожній – формуємо кнопки (див. Додаток В) з назвою вакансії та ім’ям кандидата за допомогою динамічною клавіатури, як і у розділі 2.4.1.

Після вибору потрібного кандидата, оператор бачить інформацію, що вказав про себе претендент на посаду, а також усі файлі, які він прикріпив. Якщо оператор згоден з даною кандидатурою, він натискає кнопку «Назначити співбесіду» (див. рис. 2.4.4.2), отримує згенероване посилання до конференції сервісу «Zoom» (див. рис. 2.4.4.3), яке також надходить в той самий момент до кандидата (див. рис. 2.4.4.4).

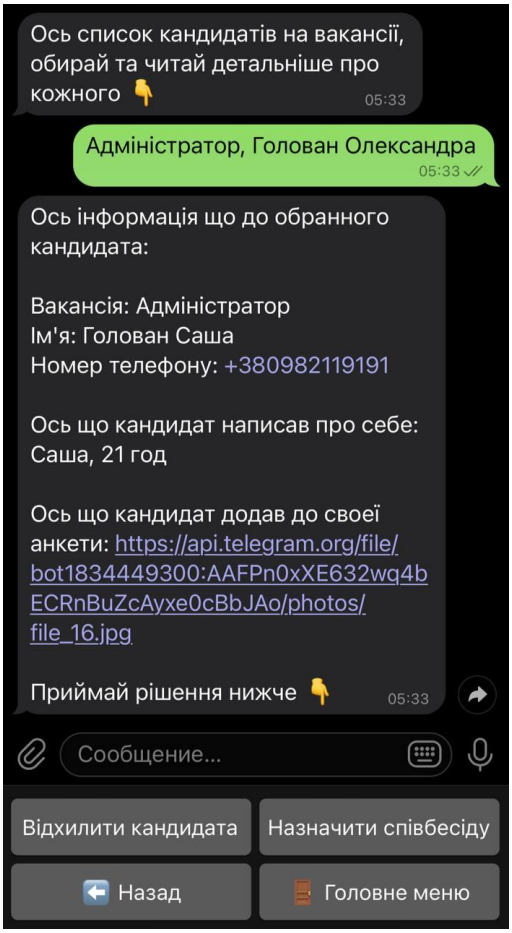


Рис 2.4.4.2. Перегляд резюме кандидата

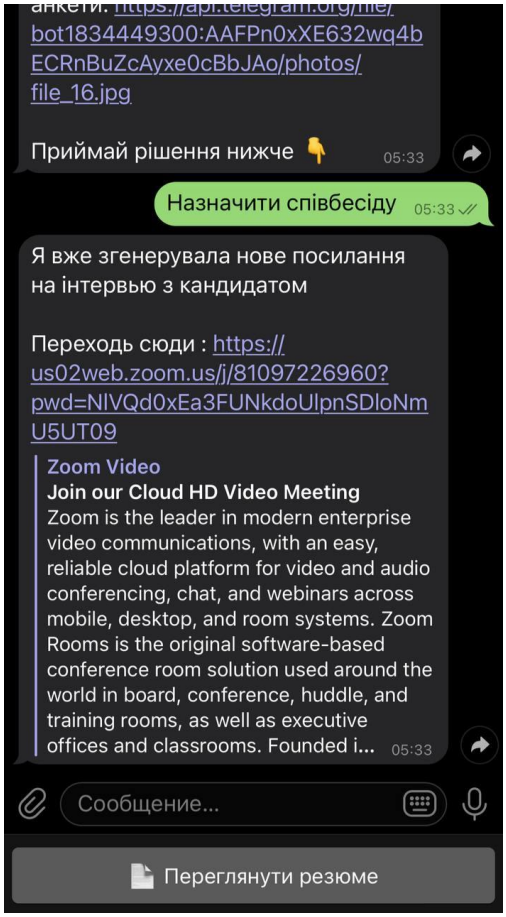


Рис 2.4.4.3. Отримання посилання на співбесіду

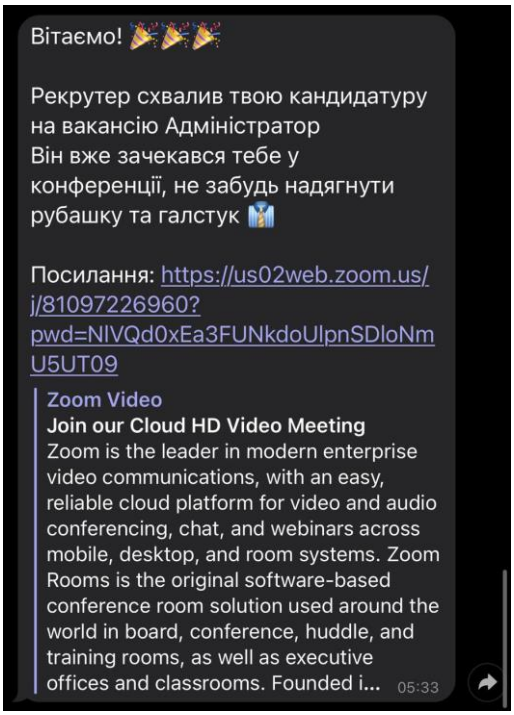


Рис 2.4.4.4. Сповіщення кандидатові про початок співбесідо у чат-боті

2.4.5 Процес «Zoom Meeting»

Процес генерує посилання на конференцію сервісу «Zoom». Це відбується за допомогою вузла «API Call» у процесі (див. рис. 2.4.5.1).

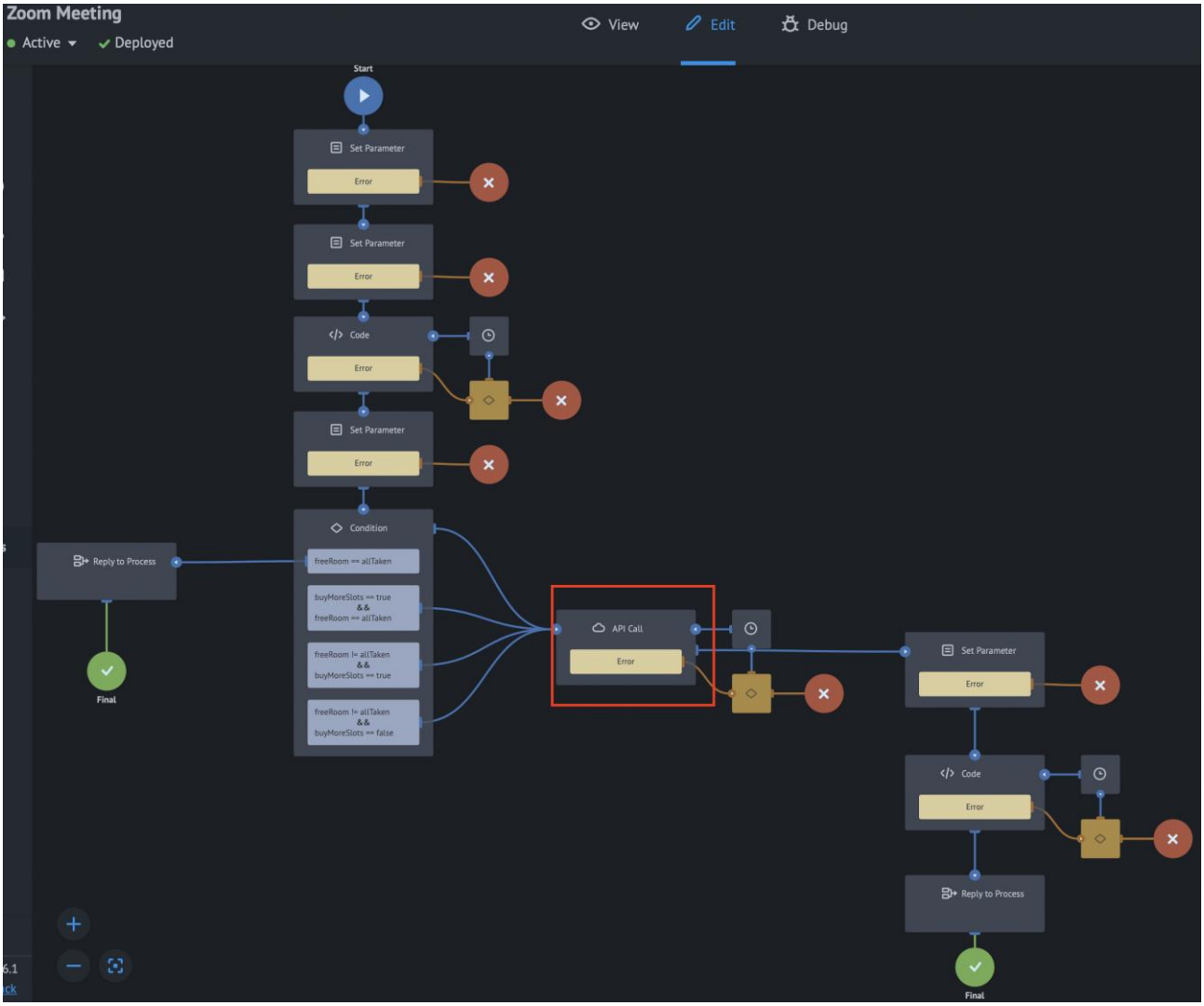


Рис 2.4.5.1. Процес «Zoom Meeting»

У вузлі ми звертаємося до посилання виду: https://api.zoom.us/v2/users/{{login}}/meetings, де {{login}} – пошта, за якою буде закріплена конференція. На цю адресу нам потрібно передавати наступні параметри(див. рис. 2.4.5.2):

- topic – назва конференції;
- start_time – час початку конференції;
- duration – тривалість конференції;

– password – пароль від Zoom аккаунту;
– settings – налаштування конференції, за замовчуванням в процесі встановлене значення {"join_before_host":true}, що означає можливість користувача підключитися до моменту підключення оператору;
– timezone – часова зона.

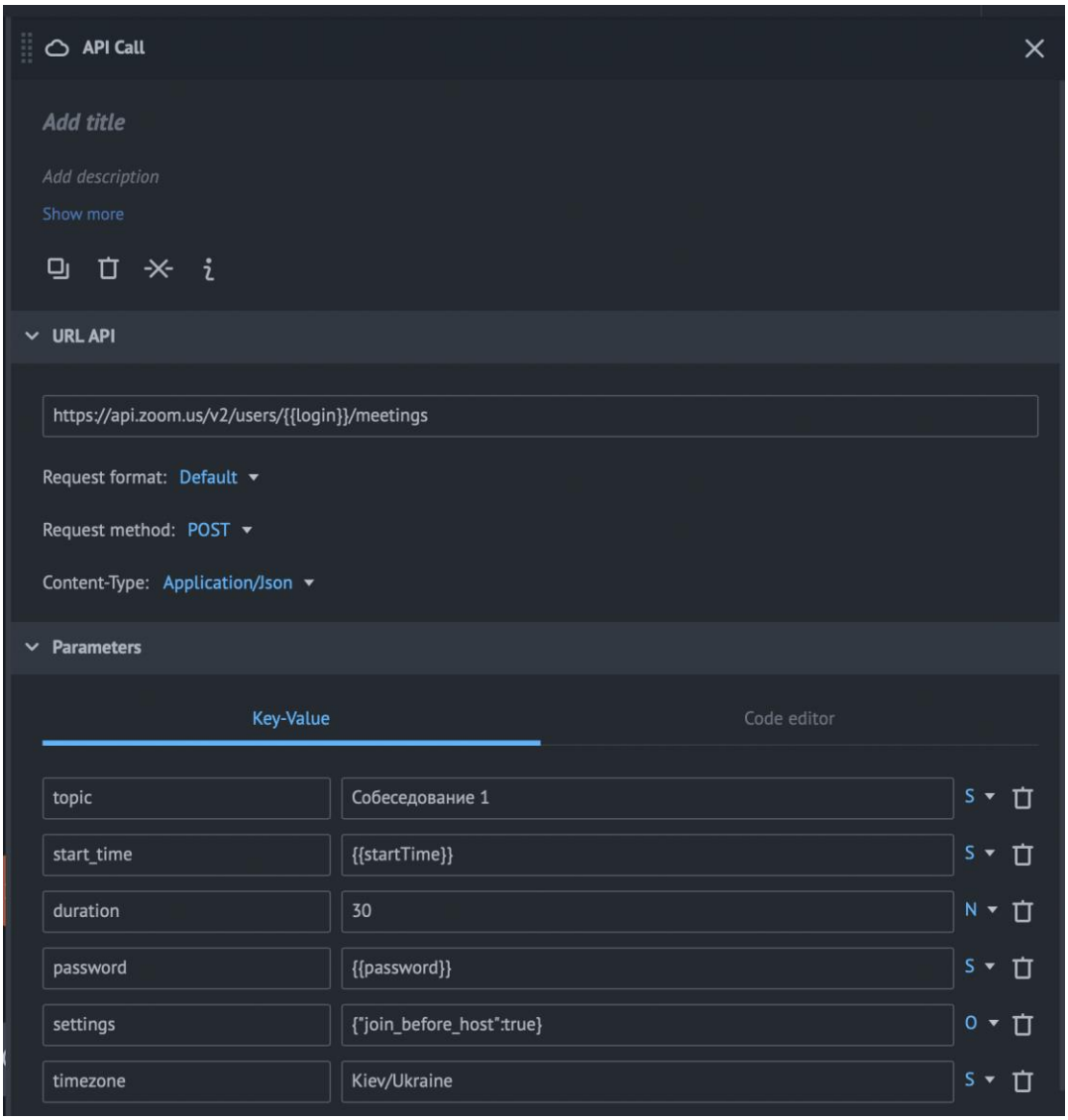


Рис 2.4.5.2. Параметри для генерування посилання на конференцію

Якщо посилання було згенеровано успішно – у заявці з’явиться параметр «start_url», який ми відправляємо до процесу «/viewTheResumes». У іншому випадку нам прийде об’єкт зі статусом «error» та код помилки, що виникла.

2.4.6 Процес «DialogFlow»

Процес «DialogFlow» за допомого якого у боті відбувається так зване «живе» спілкування. Під цим мається на увазі, що користувач може написати будь-яке слово або речення до діалогу з чат-ботом, та отримати відповідь на своє питання. Аби автоматизувати цей процес та не створювати дуже блоки коду з умовними конструкціями, було вирішено скористуватися сервісом «DialogFlow». Цей сервіс допомагає відтворювати такий випадок за допомогою функціоналу, чат-бота можна навчити відповідати на запитання користувача по ключовим словам. Наприклад, для таких фраз, як: «хочу переглянути каталог товарів», «каталог товаров», «товари», «список товаров», ключовим словом є «товари». Потрібно виділити його, як ключове, а далі чат-бот буде сам орієнтуватися, що саме відповідати на такі фрази. Також, додаток дозволяє нам переглядати усі можливі фрази, які користувачі писали до чат-боту та аналіз його відповіді їм.

Для підключення даного сервісу я скористувалася вузлом API Call (див. Рис. 2.4.6.1).

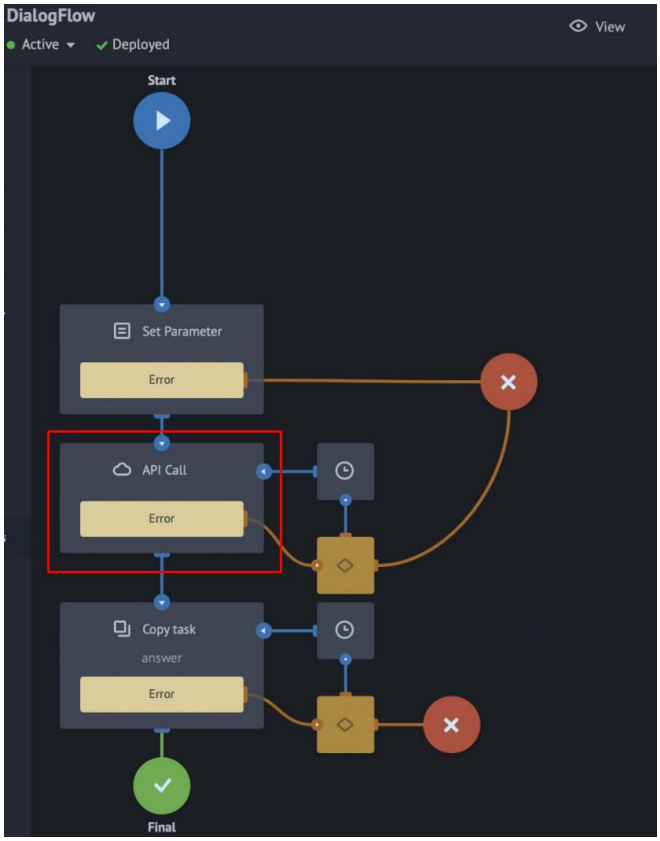


Рис 2.4.6.1. Процесс «DialogFlow»

У вузлі ми звертаємося до посилання виду: https://dialogflow.googleapis.com/v2beta1/projects/test-aphq/agent/sessions/{{token}}:detectIntent – де, {{token}}– попередньо згенерований унікальний ключ доступу до сервісу. До цієї адреси ми надсилаємо наступні дані: {"query_input": {"text": {"text": "{{question}}","language_code": "ua-UA"}}}. Питання, яке пише користувач, ми передаємо у {{question}}.

ВИСНОВКИ

Метою даного проекту було створення ПЗ, яке має спростити процес прийому на роботу. Для реалізації цієї задумки було обрано створити ПЗ – чат-бот. Цей вибір було зроблено через: популярність чат-ботів на нашому ринку, простоту у реалізації та підтримці продукту, зручність у використанні. Через стрімку популярність меседжерів у всьому світі, розробники надали їм можливість підтримувати стороннє ПЗ – чат-ботів. Чат-боти – програма, яка імітує розмову з реальною людиною, найчастіше за допомогою них можна замовити собі їжу, забронювати білети, переглянути новини, модифікувати файли (наприклад, чат-бот що конвертує формат pdf у формат doc). Саме за допомогою чат-ботів нам не потрібно встановлювати додаткове ПЗ на свої пристрої або переходити на безліч веб-сторінок для пошуку необхідної інформації – все це можна зробити у єдиному додатку, де багато з нас, скоріш за всього, проводить більшість свого екранного часу.

Чат-ботів із замовленням їжі та бронюванням квитків ми вже всі бачили. А що якби користувачі мали змогу влаштовуватися на роботу, не виходячи не тільки зі свого будинку, а ще і з одного єдиного додатку на своєму пристрої? Саме це мені і довелося реалізувати в рамках даної роботи. По-перше, це значно економить не тільки вас час, а ще і час роботодавця. Нащо їхати кудись, якщо можна надіслати всі необхідні документи та пройти співбесіду прямо у своєму смартфоні? По-друге, це економить, як ваші гроші на проїзд, так і гроші роботодавця, адже утримання декількох онлайн-операторів обійдеться значно дешевше ніж утримання менеджерів на кожній філії. По-третє, це просто зручно. Як було вказано раніше, нам не потрібно нікуди їхати, ми можемо пройти усі етапи працевлаштування за декілька натисків у чат-боті.

Було обрано найпопулярніший меседжер в Україні – Telegram. Цей меседжер має велику аудиторію та не потребують додаткового дозволу на створення чат-ботів. Також, в нього є досить гарна документація та велике спільнота розробників.

Для створення такого додатку я обрала технологію Corezoid і JavaScript. Corezoid – метамова, яка має візуальний вигляд, влаштований аналог БД та логічні

модулі (вузли). Цю технологію зазвичай використовують в парі з JavaScript для поширення базового функціоналу та написання додаткових модулів. Свій вибір я можу обґрунтувати тим, що Corezoid – дуже проста для розуміння мова; розробка на ній проходить прямо у вікні браузеру та не потребує додаткового встановлення на свою власну машину; має власний аналог БД у вигляді так званих діаграм станів; вже знаходиться на хмарному сховищі; має багато базових модулів «вузлів», таких як, наприклад, виклик API, тому самостійно нічого писати не потрібно; саме через влаштовані модулі, розробка на ній проходить значно швидше, і підтримувати такий продукт набагато легше.

Результатом даної роботи стало створення віртуального асистенту для спрощення процесу працевлаштування. В архітектурі чат-боту використовується рольова модель. Таке рішення було прийняте у зв’язку з тим, що окремі множини користувачів мають різні функціональні можливості. Наразі в боті доступні 2 ролі: користувач та оператор. Даний чат-бот має на меті значно полегшити процес прийому на роботу, бо все, починаючи від відправки резюме, до підтвердження користувача, як співробітника, виконується вдома у одному лише середовищі. Таким чином, це зекономить час не лише кандидатові на посаду, а в тому числі і самій компанії; значно зменшить витрати компанії на заробітну плату менеджерам; підвищить інтерес до компанії через таку часткову відмову від бюрократії та перехід до цифрового формату.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Документація Corezoid [Електронний ресурс] – Режим доступу до ресурсу: https://doc.corezoid.com/docs.
2. Мартин Р. Чистый код. Создание, анализ и рефакторинг / Роберт Мартин., 2008.
3. Что такое мессенджер, для чего он нужен и как им пользоваться — 6 самых популярных мессенджеров в мире [Електронний ресурс] // https://ktonanovenkogo.ru/voprosy-i-otvety/messendzher-chto-ehto-takoe-prostymi-slovami.html
4. Bots: An introduction for developers [Електронний ресурс] – Режим доступу до ресурсу: https://core.telegram.org/bots.
5. Chatbot: What is a Chatbot? Why are Chatbots Important? [Електронний ресурс] – Режим доступу до ресурсу: https://www.expert.ai/blog/chatbot/.
6. JavaScript [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Web/JavaScript.
7. Dialogflow [Електронний ресурс] – Режим доступу до ресурсу: https://cloud.google.com/dialogflow/docs.
8. Zoom Developer Platform [Електронний ресурс] – Режим доступу до ресурсу: https://marketplace.zoom.us/docs/guides.

ДОДАТОК А

Код для формування динамічної клавіатури з пагінацією у процесі «Send Message»:

```
var channel = data.channel;
var attachment = data.attachment;
var items = [];
var itemOptions = [];
var buttons = [];
var itemsPattern = attachment.items;
var buttonsPattern = attachment.buttons;
if (attachment.apple_pay_request) {
   var paymentItemsPattern =
attachment.apple_pay_request.payment.paymentRequest.lineItems;
};
// PAGINATION
var totalPages = 0;
var maxItemsCountToShow = 0;
var currentPage = parseInt(data.currentPage) || 0;
var currentOffset = 0;
if (!data.attachment.buttons) {
   data.attachment.buttons = [];
}
if (data.maxItemsCountToShow != undefined && data.maxItemsCountToShow != '') {
   data.attachment.maxItemsCountToShow = data.maxItemsCountToShow;
};
// Get max. items count to render
if (!data.attachment.maxItemsCountToShow) {
   if (attachment.type === "inline_keyboard") {
      maxItemsCountToShow = 5;
   } else {
      maxItemsCountToShow = 10;
   }
} else {
   maxItemsCountToShow = data.attachment.maxItemsCountToShow;
};
// Increment page controller
if (currentPage !== 0) {
   currentOffset = maxItemsCountToShow * currentPage;
} else {
   currentOffset = maxItemsCountToShow;
```

```
}
// Get total pages
totalPages = Math.ceil(data.items.length / maxItemsCountToShow);
// Slice array by pagination offset
itemOptions = data.items.slice(currentOffset - maxItemsCountToShow, currentOffset);
// Create dynamic keyboard
function wrapButtons(buttons) {
    var wrapped = [];
    var clone = buttons.slice();
    while (clone.length >= 2) {
        wrapped.push([clone.shift()[0], clone.shift()[0]]);
    }
   return wrapped;
         };
// Create pagination buttons
if (totalPages > 1) {
    var paginationButtons = createPaginationButtons(currentPage, totalPages, channel);
    paginationButtons.forEach(function(button) {
        if (channel === 'imessage') {
            data.attachment.buttons.push(button);
        } else {
            if (attachment.type === "keyboard") {
                data.attachment.buttons.push(button);
            } else {
                data.attachment.buttons.unshift(button);
            }
        }
    });
};
var paginationPatterns = {
    "telegram": [
        [{
            "text": "{{t'prev}}",
            "callback_data": "prev"
        },
        {
             "text": currentPage + " {{t'of}} " + totalPages,
              "callback_data": "none"
         },
         {
              "text": "{{t'next}}",
               "callback_data": "next"
          }]
    ],
```

```
};
var exitButton = {
    "telegram": [{
        "text": "{{t'/exit}}",
        "callback_data": "/exit"
    }]
};
```

ДОДАТОК Б

Код, який відповідає за зберігання інформації, яку надсилає користувач:

```
data.aboutYourself = [];
data.resumeData = [];
if (data.message.text != null) {
data.aboutYourself.push(data.message.text);
} else if (data.message.url != null) {
data.resumeData.push(data.message.url);
};
if (data.message.text != null) {
data.aboutYourself.push(data.message.text);
} else if (data.message.url != null) {
data.resumeData.push(data.message.url);
};
data.application = {
   "job": data.chosenJobTitle,
   "id": data.channel + "_" + data.chat_id,
   "name": data.name,
   "phone": data.phone,
   "aboutYourself": data.aboutYourself,
   "resumeData": data.resumeData
};
function removeNull(array) {
   return array.filter(x => x !== null);
};
switch (data.action) {
   case "addCandidate":
      data.candidates.push(data.application);
      delete data.application;
      delete data.action;
      break;
   case "deleteCandidate":
      if (data.candidates.length <= 1) {
         data.candidates = [];
      } else {
         for (var i = 0; i < data.candidates.length; i++) {
            if (data.candidates.id === data.uid) {
               delete data.candidates[i];
            }
         }
      }
```

```
        data.candidates = removeNull(data.candidates);
        delete data.job;
        delete data.id;
        delete data.action;
        break;
}
```

ДОДАТОК В

Код, який відповідає за формування кнопок для відображення списку кандидатів оператору:

```
data.reg = "(";
if (data.candidates.length == []) {

} else {
    data.buttons = [];
    data.candidates.forEach(element => {
        data.buttons.push({
            "text": element.job + ", " + element.name,
            "callback": element.id
        });
        data.reg += element.job + ", " + element.name + "|";
    });
    data.reg = data.reg.slice(0, -1);
    data.reg += ")";
    data.maxItemsCountToShow = 4;
    data.currentPage = 1;
    data.limiter = Math.ceil(data.buttons.length / data.maxItemsCountToShow);
}
```

ДОДАТОК Г

Taras Shevchenko National University of Kyiv

Development of virtual assistant for hiring process automation

SOFTWARE ARCHITECTURE DOCUMENT (SAD)

Content Owner: Oleksandra Holovan

DOCUMENT NUMBER: 1.0

RELEASE: 1.0

RELEASE DATE: 01.06.2021

Contents

1. Summary 59

2. Introduction 60

3. Methods 61

4. Architecture Representation 62

5. Processes 64

6. Testing 65

7. Results 66

8. Discussion 67

9. Conclusion 68

1. Summary

Background: Virtual assistant is the best way to improve business processes of company, due to his fast development, cheapness, and ease of support

Objective: Followed article can be used by different companies for better market-appearance and performance. Providing this system could be very useful for enterprises in HR-processes, shopping, newsletters, and communication with employees

Methods: Virtual assistant was developed on Corezoid Process Engine, including messengers API`s, like Telegram, Viber, and others. Using JavaScript code, and nodes of Corezoid. Assistant was built on client-server architecture.

Results: As a result we get a usefull Assistant for HR-processes, and shopping, that can be scaled up to another messengers, and get new functional

Conclusions: The proposed architecture, is an essential part of building customer service model.

2. Introduction

Nowadays, the use of software such as messengers is growing every day. Communicate with distant relatives, friends, make new acquaintances or even corporate communication – all this can be done on your device in real time. But what if I told you that this is not all the capabilities of modern messengers?

Automating the processes we are used to, such as ordering food or booking tickets to the cinema, is no longer something new for all of us and is performed with a few clicks in the application. But why install many different programs to your device or open dozens of pages in a browser all the time, if all this can be done in the messenger, where people usually spend most of their time using gadgets, using chatbots.

One of the still unresolved issues today is the recruitment process. Before finally getting the job of your choice, the future employee needs to spend a lot of time on: interviews, sending resumes and filling out questionnaires, signing documents and much more. It is difficult to disagree that this is not always convenient and does not bring us any pleasure at all. I think it would be appropriate to create something innovative in this regard, namely to create an electronic assistant, with which anyone could go through all these stages of employment directly at home, without leaving their own home.

3. Methods

Our assistant are hosted and developed on Corezoid process engine, using their bot platform(development packet for chat-bots), we are using standart Corezoid nodes, and messengers API’s such as Telegram, Viber, and others. Also we are using API of Zoom, and Google DialogFlow.

We are get json answer from user to our bot, editing it, and sending answer to user. All communication with user and bot are looks like a json dataframes, the functions and editing the answers is a JavaScript code.

4. Architecture Representation

The Virtual Assistant is developed as a processes in Corezoid, which are hosted completely in Corezoid engine.

When the user first opens the chat-bot, the application with his data enters the process of the receiver. There can be several such processes, depending on how many messengers are supported by the chatbot. Example:

- Telegram Receiver;
- Viber Receiver;
- Facebook Receiver.

These processes looks what the chat-bot received from the user: text, document, contact, image, video, and other objects that can only be sent to a dialog box.

After that, the application goes to the main process. Here we look at the event variable, which can take values such as: start, message, command. Regardless of the value of this variable, we try to create a user profile in the Users Profiles, which stores the profiles of all users who have started a chatbot dialogue at least once. Each application with a user in the chart is stored under the name channel_chatId, which is the name of the messenger from which the chatbot is used, and the unique user code.

The third process, which receives the user's request - the router. As the name implies, this process looks at which process the user should be redirected to. In parallel, it checks whether it is possible to use such a command in the current user role. After that, the user finally gets to the process he was referring to (for example, he clicked the "basket" button which is equivalent to the /basket command, the router looks at the command name and redirects the user to this process).

Once the command has been identified and the user has entered the desired process, we must send him the appropriate text and keyboard (if necessary). To do this, the process of sending messages was created. This process necessarily accepts the following parameters as input:

- chat_id - user ID in the bot, so we will know who to send messages to;

- channel - the channel from which the user accesses, there we will know where the user came from;
- text_id - text ID, so we will know exactly what text to send (more on this later). Can be replaced by the text parameter;
- text - the text of the message to be sent;
- attachment_id - the ID of the keyboard that will be displayed to the user.
- localization - number of the state diagram from which it will be necessary to take the text on the key defined in text_id. Without this parameter, a standard chart with localization will be selected.

The process itself is defined with the language of the user interface, and then send the desired text with, forming a keyboard (for example, you can make it dynamic), in short - a process that is fully responsible for the information we send to the user.

Normally, you can specify the text of the message to enter the process in the "text" parameter, but this is not a good solution. First, it is not convenient. If we need to change the text, we have to go to the specific process from which this text comes, find the right node and change the text right there. Second, if the bot supports more than one language, it will be very difficult to implement. This is why a state diagram such as "Localization" is usually used. This diagram is basic, and contains only one application with an array of texts objects.

To send the desired keyboard to the user, Corezoid looks at the parameter in attachment_id and goes to the standard attachments diagram. This state diagram contains applications with keyboards.

5. Processes

Processes are some separate project modules. These can be processes that call the API and process it, processes that send messages to users, simply put, something similar to the concept of "classes" in object-oriented programming. For example, we have a process of "information", which displays the user a keyboard with buttons: "news", "weather", "horoscope". In this case, we have at least 4 processes: "information", where the user will get and 3 more processes according to the buttons on the keyboard, to which we will send our application. These subprocesses can in turn refer to other processes, for example, the weather process refers to the process with the weather API call, where we pass the user city. The main processes in the project being developed are:

- /catalogOfGoods - display of the catalog of goods;
- /jobOffers ¬– available vacancies and sending your resume;
- /discounts - discounts and promotional offers;
- /basket - basket with goods, ordering and removing goods from the basket;
- /shopLocation - locations of available stores;
- /viewTheResumes - managing users' resumes: rejecting or assigning them an interview;
- /mainMenu - display the keyboard according to the user role;
- Zoom Meeting - forming a link-invitation to the conference of the service "Zoom";
- dialogFlow - integration with the DialogFlow API, for "live" communication with the chatbot.

6. Testing

Virtual assistant is passed all test-cases, such as buying products, adding products to basket, applying for vacancy, get zoom link, answer to candidate from operator, and others. So as a conclusion of testing, our assistant is finished it, and has no more bugs.

In user experience testing, our focus group of 5 people mastered using the assistant without any problems.

7. Results

As a result, we have developed Virtual Assistant for Telegram, that can be helpful for companies to sell their products, manage personal, communicate with personal and hire new people

8. Discussion

Virtual assistant – such an automated system that can automate business processes, hire and onboard new personal, and selling products online by their own.

It is an easy and quick way to reduce company costs and speed up the recruitment and sales process.

Nowadays chat-bots are used by many companies, it is convenient for both the director and the employee, from Ukrainian companies such as:

- DTEK – assistant for training and advanced training of employees;
- Ukrzaliznitsa - corporate news assistant for employees;
- UNIT.City – smart city assistant, and bot for manage your employees;
- Svoboda Slova – assistant for sociological surveys with a reward system;
- Intertop – assistant for hiring new personal.

9. Conclusion

The main reason of this project was to create software that should simplify the recruitment process. To implement this idea, it was chosen to create a software - chatbot. This choice was made due to: the popularity of chatbots in our market, ease of implementation and maintenance of the product, ease of use. Due to the rapid popularity of messengers around the world, the developers have given them the opportunity to support third-party software - chatbots. Chatbots are a program that simulates a conversation with a real person, most often they can be used to order food, book tickets, view news, modify files (for example, a chatbot that converts pdf to doc). With chatbots, we don't need to install additional software on our devices or go to many web pages to find the information we need - all in a single application, where many of us are likely to spend most of our screen time.

We have all seen chatbots with ordering food and booking tickets. And what if users were able to get a job without leaving not only their home, but also from a single application on their device? This is what I had to implement in this work. First, it saves not only you time, but also the time of the employer. Why go somewhere if you can send all the necessary documents and pass the interview directly on your smartphone? Secondly, it saves both your travel money and the employer's money, because the maintenance of several online operators will be much cheaper than the maintenance of managers at each branch. Third, it's just convenient. As mentioned earlier, we do not need to go anywhere, we can go through all stages of employment with a few clicks in the chatbot.

The 2 most popular messengers were selected: Viber and Telegram. These messengers have a large audience and do not require additional permission to create chatbots. Also, they have pretty good documentation and a large community of developers.

To create such an application, I chose Corezoid and JavaScript technology. Corezoid - metalanguage, which has a visual appearance, arranged analog database and logic modules (nodes). This technology is usually used in conjunction with JavaScript to distribute the basic functionality and write additional modules. I can justify my choice by the fact that Corezoid is a very easy to understand language; development on it takes place directly in

the browser window and does not require additional installation on your own machine; has its own analogue database in the form of so-called state diagrams; already in cloud storage; has many basic modules of "nodes", such as, for example, an API call, therefore it is not necessary to write anything independently; It is through the arranged modules that development on it is much faster, and it is much easier to maintain such a product.

The result of this work was the creation of a virtual assistant to simplify the employment process. The role model is used in the chatbot architecture. This decision was made due to the fact that individual sets of users have different functionality. There are currently 4 roles available in the bot: user, registered user, operator, administrator. This chatbot aims to greatly facilitate the hiring process, because everything from sending a resume to confirming the user as an employee, is done at home in one environment. Thus, it will save time not only for the candidate for the position, but also for the company itself; significantly reduce the company's cost of salaries to managers; will increase interest in the company through such a partial abandonment of bureaucracy and the transition to a digital forma.