МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри

_____Литвиненко О.Є.

«___» _____2021 р.

ДИПЛОМНИЙ ПРОЕКТ

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ
"БАКАЛАВР"

Тема: Програмний модуль голосового помічника для платформи *Android*

Виконавець: Чубарєв Д. М,

Керівник: Кучеров Д.П.

Нормоконтролер: Тупота Є.В.

Київ 2021

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки, комп'ютерної та програмної інженерії

Кафедра комп'ютеризованих систем управління

Освітнього ступеня бакалавр

Напрям (спеціальність) 123 "Комп'ютерна інженерія"
(шифр, найменування)

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____Литвиненко О. Є.

«___»_____2021 р.

ЗАВДАННЯ
на виконання дипломної роботи (проекту)

Чубарєва Дмитра Миколайовича
(прізвище, ім'я, по батькові)

1.Тема роботи: Програмний модуль голосового помічника для платформи *Android*

затверджена наказом ректора від "04" лютого 2021 року № 135 /ст.

2. Термін виконання роботи: з 17.05.2021 до 20.06.2021

3. Вихідні дані до роботи: 1) вимоги до змісту модуля;
2) основні операції в модулі

4.Зміст пояснювальної записки (перелік питань, що підлягають розробці):
1) аналіз можливостей голосових асистентів;
2) методика тестування умінь голосового асистента;
3) реалізація програмного модулю голосового асистента та його тестування.

5.Перелік обов'язкового графічного матеріалу:
1. Функціональні можливості голосового асистенту ;
2. Наявні засоби тестування голосового асистента;
3. Методика тестування умінь голосового асистента;
4. Схема алгоритму роботи голосового асистенту

6. Календарний план

№ п/п	Етапи виконання дипломного проекту	Термін виконання етапів	Примітка
1	Провести аналіз літератури за темою дипломного проекту та аналіз існуючих систем	17.05.21-19.05.21	
2	Провести аналіз принципів роботи голосових асистентів	19.05.21-23.05.21	
3	Розробити модель роботи системи	24.05.21-26.05.21	
4	Розробити алгоритми основних операцій голосового асистенту	27.05.21-29.05.21	
5	Розробити програмні засоби	30.05.21-02.06.21	
6	Написати пояснювальну записку	03.06.21-10.06.21	
7	Підготувати презентацію та графічні матеріали	11.06.21-14.06.21	

7. Дата видачі завдання «_17_» травня 2021 р.

Керівник дипломного проекту_____Кучеров Д.П.
(підпис)

Завдання прийняв до виконання_____Чубарєв Д.М.
(підпис студента)

РЕФЕРАТ

Пояснювальна записка до дипломного проекту “Програмний модуль голосового помічника для платформи *Android*”: 69 с., 25 рис., 22 літературних джерела, 1 додаток.

ГОЛОСОВИЙ АСИСТЕНТ, МОБІЛЬНИЙ ДОДАТОК, ФРЕЙМВОРК, ТЕСТУВАННЯ, ШТУЧНИЙ ІНТЕЛЕКТ.

Об’єкт дослідження – голосовий асистент і методи тестування його умінь.

Предмет дослідження – програмний модуль голосового помічника для платформи *Android.*

Мета дипломної роботи – збільшення швидкості та якості тестування умінь голосового помічника.

Метод дослідження – розробка модуля голосового помічника та фреймворку для автоматизації тестування умінь голосового помічника.

В процесі роботи, був зроблений глибокий аналіз процесу розробки голосових помічників на прикладі голосового помічника *Alexa*. В результаті чого виявилося, що ручне тестування розробленого уміння займає багато часу і може містити багато помилок.

Результати роботи можуть бути використані при розробці умінь для голосового помічника *Alexa* в середовищі *.NET* для автоматизації процесу тестування розробленого уміння, що зменшить час тестування та збільшить його якість.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

МН – машинне навчання

ДКП – довга короткострокова пам`ять

ШІ – штучний інтелект

ОПМ – обробка природної мови

КІ – командний інтерфейс

API – application public interface

ООП – об`єктно орієнтовне програмування

UAT – user acceptance testing

OAT – operational acceptance testing

DRY – don’t repeat yourself

ISP – interface segregation principle

JSON – JavaScript Object Notation

URL – Uniform Resource Locator

DLL – Dynamic Link Library

HTTPS – HyperText Transfer Protocol Secure

SSL – Secure Sockets Layer

ВСТУП

Для більшості з нас було б чудово мати помічника, який завжди прислухається до вас, передбачатиме всі ваші потреби і вживатиме необхідних заходів для їх задоволення. Це все тепер доступно завдяки асистентам зі штучним інтелектом, а саме – голосовим помічникам.

Кожен п'ятий житель США володіє розумною колонкою, а це 47 000 000 чоловік. Помічник може створити нагадування, список справ, будильник, таймер, прочитати новини, включити музику, подкаст, замовити доставку, купити квитки в кіно і викликати таксі. Все це «навички» або «skills» помічників. Ще їх називають голосовими додатками. Для Alexa і Google Assistant таких додатків на 2018 рік розроблено 70 000.

У 2017 році Starbucks запустив функцію на замовлення кави додому для Amazon Alexa. Крім того, що виросли замовлення на доставку, про це написали всі можливі ЗМІ, створивши класний PR. Наприклад Starbucks пішли Uber, Domino's, MacDonald's, і навіть у прального порошку Tide з'явився свій skill для Alexa.

Як у Starbucks, голосове додаток виконує одну-дві функції: замовляє каву, ставить будильник або викликає кур'єра. Щоб спроектувати щось подібне, не обов'язково бути міжконтинентальної корпорацією. Ідея, проектування, тестування, розробка і реліз схожі на аналогічні етапи в світі мобільного розробки, але з особливостями для голосу. Детально про процес розповів Павло Гвай: від ідеї до публікації, з прикладами реальної гри, з історичними вставками і розбором світу голосової розробки.

У розмовному додатку канал взаємодії з користувачем будується через розмову: Усний - з розумною колонкою, або через письмовий, наприклад, з Google Assistant. Крім колонки, пристроєм взаємодії може бути екран, тому розмовні програми ще й графічні.

Правильно говорити розмовне додаток, а не голосове, але це вже усталений термін, і я теж буду його використовувати.

У голосових додатків є важлива перевага перед мобільними: їх не треба завантажувати та встановлювати. Досить знати назву, і асистент сам все запустить.

Все тому, що нема чого завантажувати - і розпізнавання мови, і бізнес-логіка - все додаток живе в хмарі. Це величезна перевага перед мобільними додатками.

Історія голосових помічників почалася з Interactive Voice Response - інтерактивної системи записаних голосових відповідей. Можливо, ніхто не чув цей термін, але все стикалися, коли дзвонили в техпідтримку і чули робота: «Натисніть 1, щоб потрапити в головне меню. Натисніть 2, щоб дізнатися детальніше »- це і є IVR система. Почасти, IVR можна назвати першим поколінням голосових додатків. Хоча вони вже частина історії, але дечого можуть нас навчити.

Більшість людей при взаємодії з IVR-системою намагаються зв'язатися з оператором. Це відбувається через погане UX, коли взаємодія заснована на жорстких командах, що просто незручно.

Це підводить нас до основного правилом хорошого розмовного додатки.

Гарне розмовне додаток взаємодіє з користувачами не через суворі команди, а через живий, природний розмову, схожий на спілкування між людьми.

Розмова з додатком повинен бути більше схожий на дзвінок в піцерію для замовлення, ніж на спілкування командами з чат-ботом. Досягти такої ж гнучкості, як в розмові між людьми, не вийде, але говорити з додатком комфортно і природною мовою - цілком.

У цьому теж перевага голосу перед графічними додатками: не потрібно вчитися користуватися. Моя бабуся не вміє заходити на сайти або замовляти піцу через додаток, але викликати доставку через колонку зможе. Ми повинні

використовувати цю перевагу і підлаштовуватися під те, як кажуть люди, а не вчити їх говорити з нашим додатком.

Голосовий світ крутиться навколо віртуальних помічників: Google Assistant, Amazon Alexa і Аліса.

Все влаштовано майже як в мобільному світі, тільки замість платформ iOS і Android тут Аліса, Google Assistant і Alexa, замість графічних додатків - голосові, з власними назвами або іменами, і у кожного помічника свій внутрішній магазин голосових додатків. Знову ж таки, говорити «додаток» неправильно, так як у кожної платформи свій термін: у Аліси - «навички», у Алекси - «skills», а у Google - «actions».

Щоб запустити skill, я прошу асистента: «Алекса, передай Starbucks, що я хочу кави!», Алекса знайде додаток кав'ярні в своєму магазині і передасть йому розмову. Далі розмова йде не між Алексой і користувачем, а між користувачем та програмою. Багато плутаються і думають, що з ними продовжує говорити асистент, хоча у додатки вже інший голос.

Для користувача у додатку немає ніякої графічної частини - все виглядає як набір діалогу. Зовні може здатися, що додаток - це проста річ, створити його просто, але це не так. Етапи розробки такі ж, як і у мобільних додатків.

Розробка ділиться на дві частини: розробка системи розуміння мови і написання логіки. Два перші етапи специфічні, так як додатки розмовні, а два останніх - стандартні.

Об’єкт дослідження – голосовий асистент і методи тестування його умінь.

Предмет дослідження – програмний модуль голосового помічника для платформи *Android*.

Мета дипломної роботи – збільшення швидкості та якості тестування умінь голосового помічника.

Метод дослідження – розробка модуля голосового помічника та фреймворку для автоматизації тестування умінь голосового помічника.

РОЗДІЛ 1
АНАЛІЗ МОЖЛИВОСТЕЙ ГОЛОСОВИХ АСИСТЕНТІВ

1.1. Загальні відомості про голосового асистента на прикладі програми *Amazon Alexa*

UX і UI-дизайнери все ще зі скепсисом дивляться в бік голосових інтерфейсів. Одним здається, що це маркетинговий хайп, який скоро зійде нанівець. Інші не користуються голосовими асистентами і тому впевнені, що голос - це незручно і неприродно. Але поки вони сумнівалися, склалася самостійна професійна сфера - зі своїми секретами, паттернами і механіками (і навіть ринком праці). Разом з UX-архітектором Just AI Катериною Юлиної розбираємося, як підступитися до голосових технологій і що в своєму мисленні повинен змінити UX-дизайнер традиційних інтерфейсів, беручись за голосові.

Але для початку трохи ретроспективи. На дворі 1995 рік. Microsoft випускає ОС Windows 95, і в світі відбувається революція. Пам'ятаю, як батьки та інші дорослі говорили про користь і шкоду комп'ютера. А у вихідні ми всією сім'єю ходили в гості до друзів, щоб пограти в «Косинку» та «Сапера».

У 2000-му вийшла Nokia 3310. Світ знову змінився, на цей раз завдяки телефонам з кнопковим інтерфейсом. Пізніше з'явилися смартфони з стилусами. У мене теж був такий. Тикати стилусом в екран було дуже круто. Відразу якось виділяєшся на тлі людей з кнопковими телефонами. Але «кращий стилус - це ваш власний палець», говорив Стів Джобс. У 2007 році Apple починає продавати iPhone - і з тих пір люди Тапа і свайпа, а тикати стилусом вже давним-давно стрьомно.

А потім понеслося: 2011 рік - Apple презентує Siri, в 2014-му Amazon випускає Alexa і Amazon Echo, в 2016-му виходить Google Assistant, в 2017-м - «Аліса» від «Яндекса» ... Перед розробниками і бізнесом відкрилися двері в екосистеми асистентів, як колись - в магазини мобільних додатків. І скоро тільки ледачий (і недалекоглядний!) Не робитиме навички для голосових помічників.

Неможливо уявити, щоб user experience (UX, призначений для користувача досвід) будувався в вакуумі. Голосовий навик народжується в конкретній ситуації, де він корисний і органічний, - безглуздо створювати голосовий сценарій, а потім думати, де б його застосувати. Важливо цю ситуацію виявити і майстерно її обіграти.

Голос зручніше, ніж веб- або мобільні додатки тоді, коли нам потрібна конкретна функція для вирішення конкретного завдання. Чому? Тому що не потрібно чекати завантаження сайту, скролл сторінки, шукати по меню, натискати кнопки. Сайти та додатки багатофункціональні. Голосовий навик повинен бути заточений під одиничний кейс, під «тут і зараз».

У лютому 2019 американський канал TNT транслював матчі NBA. У перервах між іграми «Лос-Анджелес Лейкерс» і «Бостон Селтікс» коментатор Ерні Джонсон оголошує, що почався продаж лімітованих кросівок Nike і що глядачі можуть попросити Google Assistant забронювати для них пару за $ 350.

Через шість хвилин кросівки зникли. Більше 15 тисяч людей забронювали їх за допомогою голосового помічника (число замовлень перевищила кількість доступного товару). Ідеальною ситуацією для продажу баскетбольної взуття виявилася справжня гра.

Ще в 2017 році Starbucks придумала, як за допомогою голосового помічника скоротити час очікування замовлення і шлях клієнта до бажаної чашці кави. Можна сказати «Alexa, order my Starbucks» («Алекса, замов мій« Старбакс »») і забрати улюблений напій в найближчому закладі вже через кілька хвилин. Не потрібно стояти в черзі і чекати, коли бариста викрикне ваше ім'я. Не потрібно заходити в додаток, шукати свій звичайний замовлення, підтверджувати вибір. Потім цей же кейс обіграла Ford: Alexa вбудували в мультимедійну

систему SYNC3, так що у водіїв з'явилася можливість замовляти і оплачувати голосом свій Starbucks прямо в салоні автомобіля, не відволікаючись від дороги.

А ось колеги з компанії Alan AI розповіли мені, як виявили прикладної юзкейс - вже не для маркетингових, а для enterprise-завдань. Технічному персоналу, який обслуговує ліфти в США, доводиться читати тонни документації, заповнювати ряд бланків, вносити дані про ремонт, звітувати про виконання завдань - і робити все це в різних системах обліку. На писанину йде багато часу, в яке ліфтер фактично не працює. Alan AI застосували технології голосового штучного інтелекту, щоб користувачі-ліфтери могли заповнювати бланки голосом в ході роботи або під час поїздки на об'єкт.

Голосовий UX - це не про картинки, а про контекст ситуації. Завдання дизайнера - детально вивчити контекст і зрозуміти, що користувач захоче в конкретній ситуації.

1.2. UX-патерни в голосі

Отже, один навик - одна функція. Якщо водій замовляє каву, то він вирішує цю конкретну задачу. Закінчив з кавою і хоче дізнатися відстань до Місяця? За це відповідає вже інший голосовий навик. Робити швейцарський ніж в одному навику - погана затія.

Скролл.UX- і UI-дизайнери постійно сперечаються з маркетологами про те, скільки інформації показувати на першому екрані. Немає нічого гіршого, ніж чекати подгрузку контенту, нескінченно прокручуючи сторінку. У діалогових інтерфейсів теж є перший екран і, на відміну від інтернету, відсутня скролл. Він не потрібен, адже голосовий асистент - це набір навичок. Користувач за командою активує навик і запускає одну-єдину функцію.

Модальні вікна і кнопки.Друга назва модальних вікон - діалог. У чому суть модальних вікон? Підтвердити або відмовитися від наміру виконати операцію. У реальному житті люди висловлюють наміри, кажучи так чи ні, і для цього не потрібні кнопки.

Мультимодальні.«Розумні» колонки і екрани, смартфони, що говорять іграшки для дітей, «розумні» будинки, приладова панель в автомобілі - пристрої визначають контекст використання. Те, що користувачеві зручно робити вдома через «розумну» колонку, буде відрізнятися від використання голосового помічника в автомобілі. Один і той же асистент на різних платформах володіє різним набором умінь.

Визначте контекст використання і пристрій. Перевірте, чи можна комбінувати голос з іншим видом взаємодії на цьому пристрої.

Кросплатформеність. Інший аспект - як зробити один навик під кілька асистентів. Механіка і логіка можуть зберігатися, але диявол буде ховатися в деталях реалізації та шляхи користувача до навички. Подивимося, як на одній і тій же платформі виглядає шлях до навичкам «Аліси» та діям Google Assistant.

«Аліса» на iOS. Користувачам «Аліси» потрібно спочатку завантажити додаток «яндекс.браузер», дозволити йому використовувати локацію і мікрофон, натиснути значок асистента і сказати активаційну фразу навички. Наприклад, «Запусти навик« Так, пане »». Другий і наступний досвід користувача складається вже з трьох кроків: відкрити «яндекс.браузер», натиснути на кнопку виклику асистента, сказати активаційну фразу.

Google Assistant на iOS. Користувач завантажує додаток Google Assistant, авторизується через акаунт, дозволяє відправку повідомлень і погоджується на доступ до мікрофона. Якщо в телефоні за замовчуванням обрана англійська мова, то в інтерфейсі додатка треба вибрати російську, попросити асистента голосом поміняти мову і сказати активаційну фразу для виклику дії. Тут вона інша - «Говорити з додатком« Так, пане »». Потім шлях скорочується - вдруге користувач відкриє додаток Google Assistant і просто озвучить команду.

Не знаючи активаційну фразу (а команди виклику одного і того ж навички в різних екосистемах будуть відрізнятися), навик не запустиш - і в плані UX це одне зі слабких місць голосових асистентів. Але над завданням зручного skills discovery (як правильно донести до споживачів інформацію про нові навичках) зараз працюють і Google, і «Яндекс», і інші компанії, так що з часом знаходити корисні і актуальні в певний момент навички буде простіше.

І, до речі, VUI-дизайнери наполягають, що навик з однією і тією ж механікою для різних асистентів - все ж окремі проекти.

1.3. Проектування інтерфейсу в сім етапів

Я питала VUI-дизайнерів, колег з Just AI і користувачів конструктора навичок Aimylogic, з чого вони взагалі починають роботу над сценарієм. Всі говорять «з ідеї». Описати її можна у вільній формі в текстовому редакторі.

Павло Гвай, засновниктortu.io- інструменту для проектування голосових додатків: «Найшвидший і дешевий спосіб зрозуміти, як буде будуватися розмова між користувачем і вашим додатком, - написати приклад діалогу. Це текстовий файл, що описує роботу будь-якого флоу. За форматом приклади діалогів нагадують сценарій до фільму, де всі репліки розписані по ролям ».

Розкажіть, що робить бот. Використовуйте ємні фрази середньої довжини. Закінчуйте фразу закритим питанням: так користувач зрозуміє, що від нього хочуть.

погано- "Привіт! Я Активити. Не можу прожити жодного дня без спорту. Він моє все, моє життя, моє натхнення! А ще у мене великий досвід в якості тренера і сотні вправ в базі! З радістю поділюся своїми улюбленими вправами з тобою! »

добре- "Привіт! Я Активити, бот-спортсмен. З радістю порекомендую тебе комплекс вправ. Хочеш, розповім про йогу? »

Опрацюйте шляху користувача

Мовою програмістів намір - Интент (англійське intent). Аналогія з класичними інтерфейсами - формалізований або неформалізовані запит. З формалізованим працювати простіше, а неформалізовані може перетворити життя на кошмар.

Дизайнери використовують блок-схему при роботі з шляхами користувача. Її завдання - допомогти описати логіку програми. Блок-схема складається з

кроків діалогу від імені користувача та системи, іноді на неї додають логічні елементи - виклики API, роботу з контекстом.

Блок-схема в Miro

Часто блок-схема описує основні розвилки в голосовому навику. Деякі дизайнери покривають блок-схемою кожну деталь навички. Павло Гвай робити цього не рекомендує, тому що вона дуже швидко перестане бути читабельною, а внесення будь-яких правок буде займати багато часу.

Найпростіший спосіб опрацювати шлях користувача - пройти разом з ним від початку до кінця і подивитися, в яких точках з'являться умови і відхилення від основного сценарію.

створіть навігацію

На якому б етапі користувач не знаходився, у нього завжди повинна бути можливість почати спочатку, повернутися назад, просунутися вперед і відповідати варіативно. Не змушуйте його запам'ятовувати команди.

Приклад з життя. Вже згаданий мною навик «Так, пане» (його аудиторія в «Алісі» - 650 тисяч осіб, до того ж він став одним із самих популярних ігор для Google Assistant не тільки в Росії, але і в світі) - це атмосферна гра з музикою , гідним синтезом мови, цікавими сюжетами, ігровий механікою. На старті використовувати міг керувати грою тільки за допомогою «Так», «Ні» і «Досить». За сюжетом гравець - пане, власник земель і селян, потрапляє в ситуацію «Ваша світлість! У наших благодатних краях знайдені великі поклади міді! Зволите почати будувати шахту? ». Можна було б і просто відповісти так, але хочеться ж вжитися в роль і говорити, як личить великосвітської особі.

- Ваша світлість! У наших благодатних краях знайдені великі поклади міді! Зволите почати будувати шахту? - зволив!

VUI-дизайнер виявив цей момент і навчив навик краще розуміти користувачів, надавши їм більше свободи при відповіді.

Випишіть приклади діалогів і робіть відповіді бота різноманітнішими

Мої колеги по Just AI пишуть приклади діалогів в таблицях. Їм так зручніше, але це далеко не найзручніший варіант - описувати і логіку, і можливі

переходи. Хтось пише сценарії в Word. Якогось єдиного формату та регламенту опису немає.

Користувачів сильно дратує, коли асистент починає повторюватися. Гіпотезу підтверджує Неллі Камаева, дизайнер Alan AI. Під час тестування навички для дітей вона бачила, як швидко хлопці втрачали інтерес, зіткнувшись з одним і тим же відповіддю.

Варто передбачити кілька синонімічних за змістом реплік, які буде чути користувач, потрапляючи на один і той же крок сценарію. VUI-дизайнери рекомендує використовувати від трьох до десяти варіантів однієї фрази.

Загляньте в «смітник» для нерозпізнаних намірів

«Смітник», або catch-all мовою програмістів, - це місце, куди з різних причин потрапляють фрази пользователей.- Я з ким взагалі говорю? Ти робот?! - Ви мене розкусили. Хочете продовжити спілкування? Фраза «Я з ким взагалі говорю? Ти робот?!" потрапить в catch-all (якщо ви її не передбачили сценарієм). «Ви мене розкусили. Хочете продовжити спілкування? » - репліка за замовчуванням при подібних ситуаціях. Подумайте заздалегідь, ніж допомогти користувачеві, що провалився в catch-all.

Згадайте співрозмовників, які наводили на вас нудьгу. З ними нецікаво, розмова продовжувати не хочеться, ми їх називаємо нудними. Те ж саме відбувається і при спілкуванні з навичками. «Аліса», Alexa і Google Assistant мають свої Speech Toolkit з широким набором чоловічих і жіночих голосів, акцентів і звукових ефектів для пожвавлення мови бота на кшталт покашлювання і пошмигіванія.

Якщо ви хочете вразити користувача і залучити його в розмову, помарудитеся зі стилістикою мови, попрацюйте над синтезом: розставте наголоси, паузи та інтонації. Це копітка робота, але, повірте, результат того вартий. На внутрішньому воркшоп колега створила навик для Alexa - послухайте фрагмент:

- Hey, man. Awful weather, yeah? Wanna get drunk? Let your old friend Joe mix you a good Irish drink. Ok? - And now when you feel better, take your effing ass and go to the Joe's pub. There you will never drink alone. I'm waiting for you.

Ще один спосіб працювати з голосом - записати аудіо з професійними акторами. Довго, дорого, втрачається гнучкість, якщо ви захочете щось додати в сценарій, зате ефектно. Навик може говорити голосами знаменитостей, політиків і героїв з фільмів (ну, нашуозвучку гри «Світ Лавкрафта»ви вже могли чути, а от як Google Assistant говорить голосом лауреата Греммі Джон Ледженд -ти до).

Для одного з бізнес-кейсів ми теж використовували записи голосів професійних акторів. Тільки 0,5% клієнтів змогли запідозрити (навіть не здогадатися, а запідозрити), що з ними розмовляє бот.

Підібрати звуки можна з готових бібліотек або створити свої (Alexa вибаглива до розширення файлів, тому доведеться повозитися з конвертацією). У тій же грі «Так, пане» для «Аліси» звуки використовуються для створення атмосфери: іржання коня, ремствування натовпу, зловісний сміх, волинка. А недавно в «Алісі» з'явився навик «Птахи Росії», де в бесіді можна почути, як співають і щебечуть жайворонок, горобець або інші птахи.

Дизайн голосового інтерфейсу не обмежується блок-схемами і сухим текстом. У дизайнера є все для створення навичок, які залучають користувача в процес.

І ще поради від VUI-дизайнерів

Не вчіть інтерфейсу.Мова - звичний і зрозумілий інтерфейс. Не треба вчити людину говорити. Він це вміє.

погано- «Щоб прослухати повідомлення заново, скажіть« Прослухати заново ». Щоб перейти до наступного повідомлення, скажіть «Перейти до наступного повідомлення» ».

добре- «Послухаємо повідомлення ще раз або перейдемо до наступного?»

Давайте закриті питання. Рекомендую уникати відкритих питань та відкритого кінця висловлювань бота, користувача потрібно направити до дії.

погано- "Привіт! Я Симфоні, бот-меломан. З радістю порекомендую тобі альбом і розповім про нього ».

добре- "Привіт! Я Симфоні, бот-меломан. З радістю порекомендую тобі альбом і розповім про нього. Хочеш дізнатися про трек дня? »

Уникайте канцеляризмів. Очевидна рекомендація, яку мало хто слідує. Ніхто з нас не хоче читати складний і перевантажений текст, а вже слухати його тим більше нестерпно.

Погано - «Важливо мати на увазі, що подальші альбоми цього виконавця стануть платиновими, що дозволяє зробити висновок про успішність даного дебютного альбому як засобу ефектного виходу на міжнародну арену».

Добре - «Дебютний альбом привернув до виконавця увагу всього світу. Недарма його наступні платівки двічі стали платиновими! »

Тестируйте навик в тиші, на вулиці, в шумному приміщенні, говорите з різною інтонацією і з різною швидкістю. Навіть в самому тихому місці щось може піти не так. Можливо, комусь процес тестування здасться нудним, але, запевняю вас, це не так. Навику потрібен справжній краш-тест!

З особистого досвіду. На одному з воркшопів я проектувала фітнес-навик: за задумом Alexa спочатку робила інструктаж, потім включалася музика і людина повторював вправу. Я з азартом тестувала все на собі: стрибала і бігала під нарізану музику, змінювала довжину треків, повторювала вправи безліч разів і в кінцевому рахунку UX навички мене задовольнив.

Користувачі - провокатори. Вони перевірять реакції навички не по темі: навик для замовлення піци, а користувач - начебто дуже хитрий - запитає про суші. Придумайте гідну відповідь. В Just AI є чек-лист мату, який використовують при тестуванні.

Проговаривайте все, що ви придумали. Послухайте своїми вухами все, що буде чути користувач. Попросіть колег зачитати і навіть розіграти сценарій. Записуйте мова, повертайтеся до записів, експериментуйте.

Але з першого разу все в навику ви не передбачте. Прийміть це і змиріться. Вашій навику потрібно донавчання. Читайте діалоги і аналізуйте логи. Як це робити за допомогою Python, в одному з випусків «Школи Аліси» розповідає Дар'я Сердюк, NLP Research Engineer Just AI.

1.4. Висновки до розділу

Проведено голосового принципів побудови голосового асистента. Описано основі відомості про голосові асистенти.

Проведено аналіз наявних інструментів для автоматизованого тестування умінь голосового асистента і виявлено, що вони покриваються дуже вузькі сфери розробки програмного забезпечення.

В результаті аналізу було вирішено, що потрібно розробити власну методику тестування умінь голосового асистента, а також спроектувати фреймворк для автоматизації тестування умінь згідно розробленої методики.

РОЗДІЛ 2
МЕТОДИКА ТЕСТУВАННЯ УМІНЬ ГОЛОСОВОГО АСИСТЕНТА

2.1. Вбудовування голосового помічника в мобільний додаток

Голос - найважливіший інструмент для людей з порушеннями зору.

Голос корисний при проходженні нами сценарію «form-filling» і зручний для заповнення майже будь-яких довгих форм, що вимагають від користувача певного обсягу інформації. І такі форми присутні в більшості мобільних додатків.

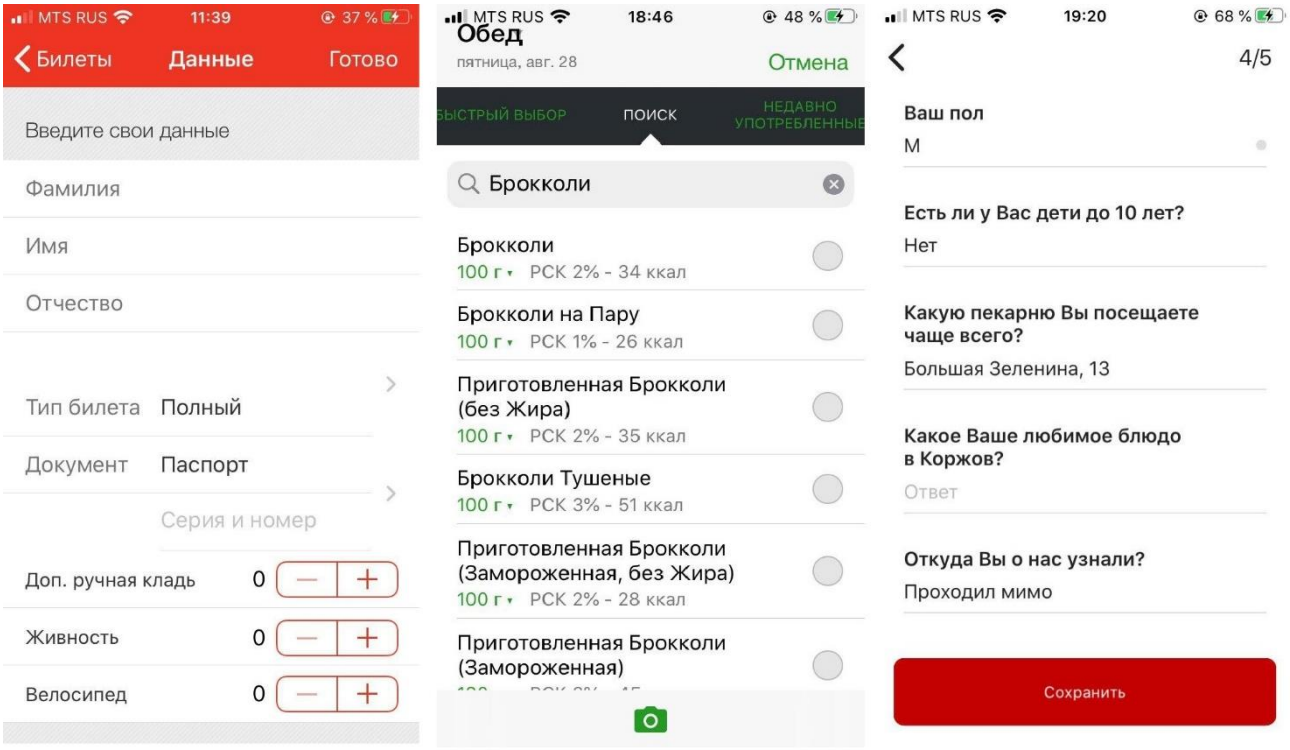


Рис. 2.1. Голосовий помычник

Правильний підхід - вбудовувати асистента бесшовно в уже існуючий функціонал додатка, в інтерфейсі якого буде відбуватися заповнення форми, щоб людина могла просто перевірити, що він все правильно сказав, і натиснути ОК.

Ми вирішили показати, як це можна зробити, на прикладіHabitica- це опенсорсний додаток, написане майже на чистому Котлін. «Хабітіка» відмінно підходить під кейс з голосовим асистентом - тут теж для того, щоб завести нову задачу, потрібно заповнити досить об'ємну форму. Спробуємо замінити цей тоскний процес однією фразою з навідними питаннями?

2.1.1. Aimyboxяк SDK

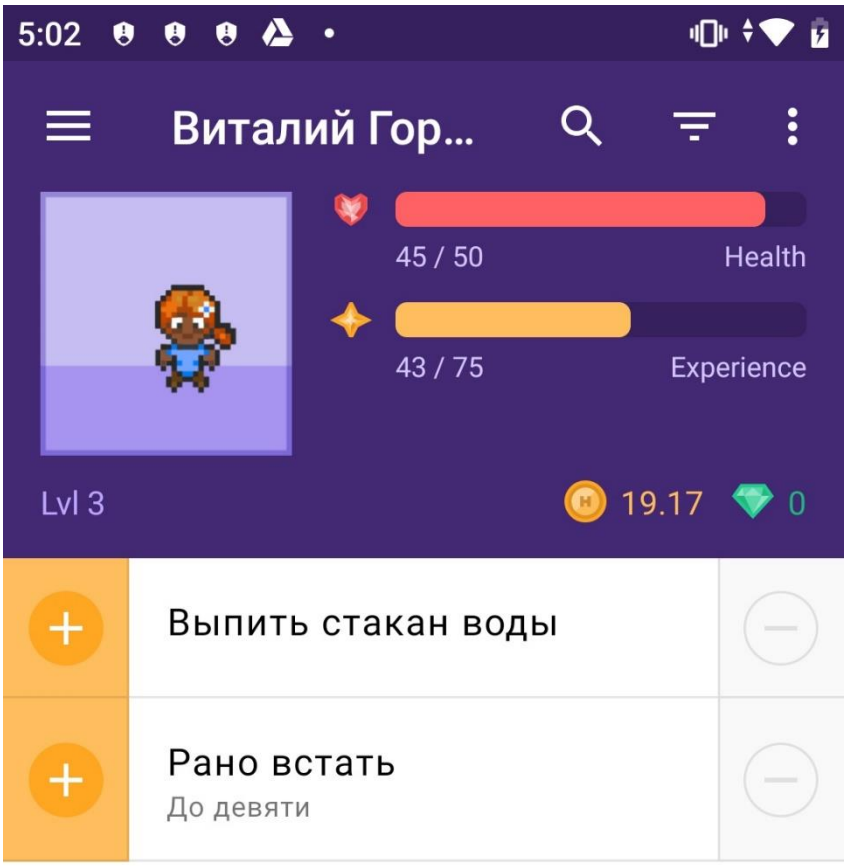
Використано Aimyboxяк SDK для побудови діалогових інтерфейсів. З коробки Aimybox даєSDKасистента і лаконічний ікастомізіруемий UI (Який при бажанні можна і зовсім переробити). При цьому в якості двигуніврозпізнавання,синтезуіNLPможна вибрати з уже наявних або створити свій модуль.

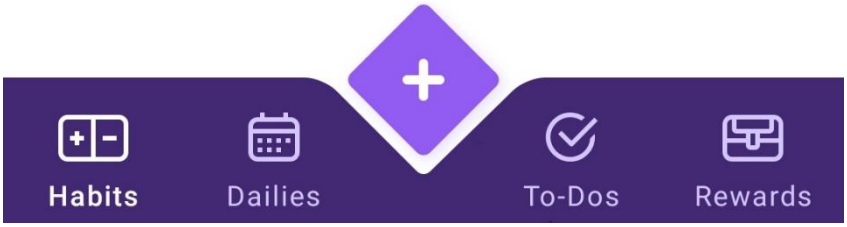
По суті, Aimybox реалізує архітектуру голосового помічника, стандартизуючи інтерфейси всіх цих модулів і правильним чином організовуючи їх взаємодія. Таким чином, впроваджуючи це рішення, можна значно скоротити час на розробку голосового інтерфейсу всередині свого застосування. Детальніше про Aimybox можна прочититутаботут.

Інструмент для створення сценарію.Сценарій писатимемо наJAICF(Це опенсорсний і абсолютно безкоштовний фреймворк для розробки голосових додатків від Just AI), а ІНТЕНТ розпізнавати за допомогоюCaila(NLU-сервіс) вJAICP(Just AI Conversational Platform). Про них докладніше розповім в наступній частині туторіал - коли дійдемо до їх використання.

Смартфон.Для тестів нам знадобиться смартфон на Android, на якому ми будемо запускати і тестувати «Хабітіку».

Користувався IDE Android Studio:



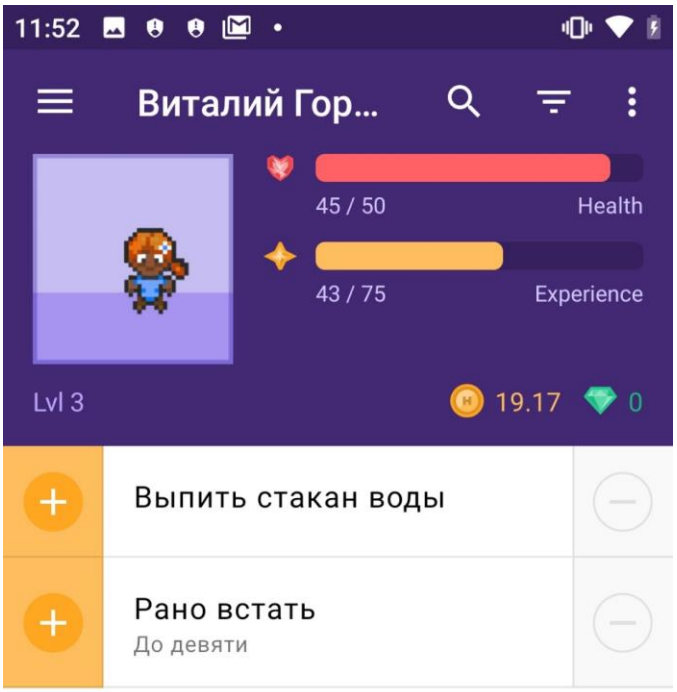


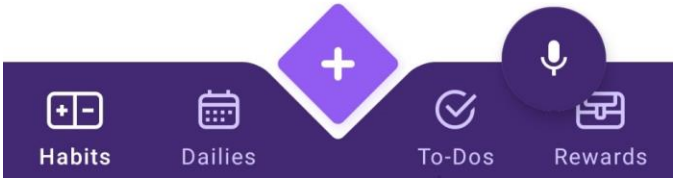
Для початку додамо залежності Aimybox.

Додаємо AimyboxProvider при ініціалізації класу.

```
//contains all HabiticaApplicationLogic except dagger componentInitialisation
abstract class HabiticaBaseApplication : MultiDexApplication(), AimyboxProvider {
    var refWatcher: RefWatcher? = null
    @Inject
    internal lateinit var lazyApiHelper: ApiClient
    @Inject
    internal lateinit var sharedPrefs: SharedPreferences
    @Inject
    internal lateinit var crashlyticsProxy: CrashlyticsProxy
```

Тепер вбудовуємо фрагмент в mainActivity.kt. Попередньо вставляємо ФреймЛейаут в activity_main.xml, прямо під фреймлейаутом з id bottom_navigation

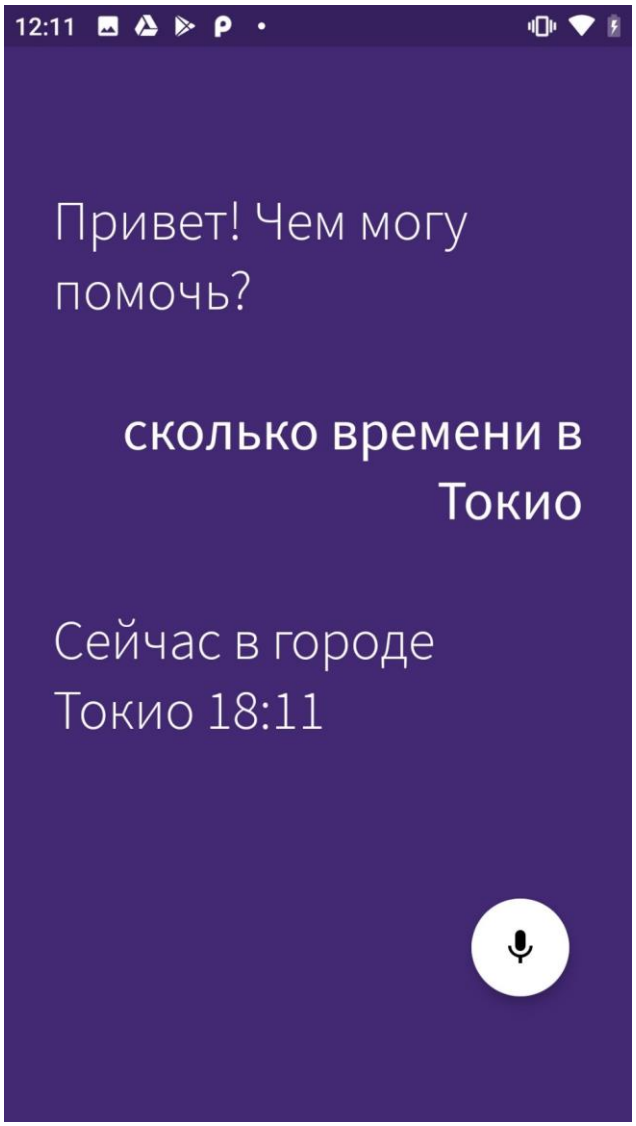




Тепер підключимо туди асісстента і перевіримо, чи нормально він відповідає. Для цього нам знадобиться консоль Aimybox.

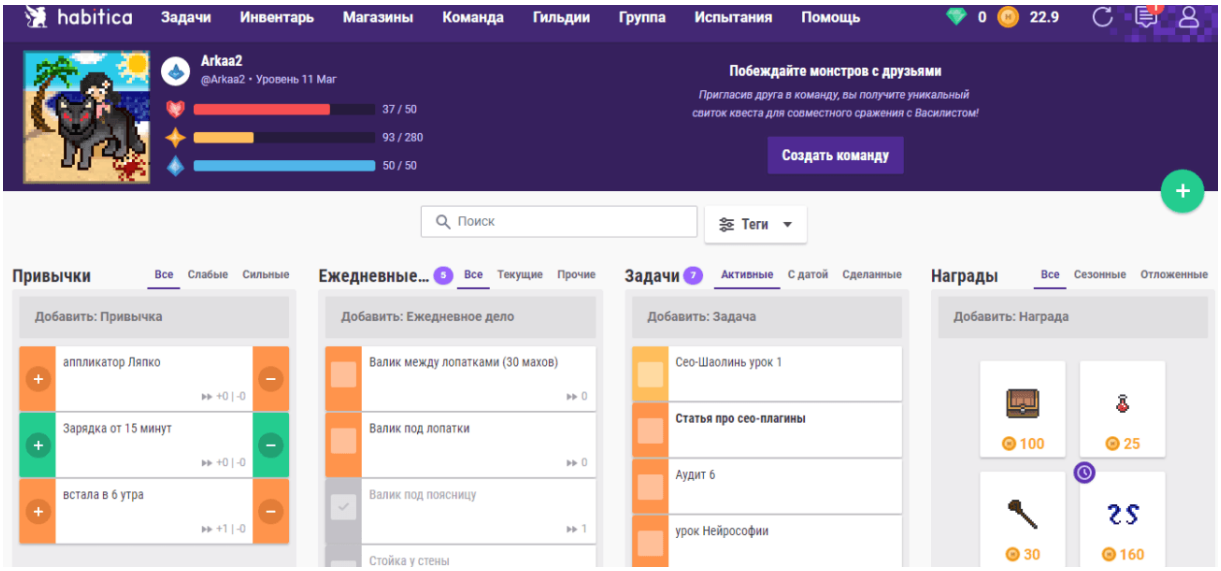
Почнемо з того, що зайдемо в app.aimybox.com під нашим акком гітхаба, зробимо новий проект, підключимо пару навичок (я підключив DateTime для тесту) і спробуємо поставити відповідні питання в асисстент. Тут же в налаштуваннях, в правому верхньому куті, беремо apiKey, який вставляємо в createAimybox замість YOUR KEY.

Тільки напис англійською, давайте поміняємо вітальне повідомлення в strings.constants.xml.



2.2. Голос у мобільному додатку

Як швидко і бесшовно вбудувати голосовий інтерфейс в ваш мобільний додаток? І як навчити асистента всьому, що воно вміє? ВМинулого разуми взяли опенсорсний лайфстайл-додатокHabiticaі показали, як додати в нього помічника і запив базовий голосової сценарій «з коробки» (уточнення прогнозу погоди і часу). А тепер перейдемо до більш просунутому етапу - навчимося викликати голосом певні екрани, робити складні запити з NLU і form-filling за допомогою голосу всередині програми.



Отже, Habitica - це додаток для вироблення хороших звичок з елементами Гейміфікація: підтримка ваших життєвих цілей у вигляді звичок, щоденних справ і завдань заохочується нагородами. І зараз ми навчимо голосового асистента, якого самі ж в додаток і поселили, як створювати і заповнювати таски, шкідливі звички і нагороди голосом, а не вручну.

Почнемо з найпростішого - логіки на стороні додатки. Ми хочемо по голосовій команді відкривати, наприклад, настройки або вікно зміни характеристик. Відкриваємо AndroidManifest і знаходимо відповідні активують. Знаходимо PrefsActivity, який відповідає за настройки, FixCharacterValuesActivity, який відповідає за зміну характеристик персонажа, і до купи знаходимо активують, по якій відкривається профіль і інформація про програму, FullProfileActivity і AboutActivity.

Згідно з документацією, нам потрібно вносити клієнтську логіку в клас, успадкований від CustomSkill. По-перше, зазначимо, що нам потрібно реагувати тільки на відповідь від бота, що містить в response.action "changeView". У response.intent ми будемо передавати безпосередньо команду, куди саме переходити - і в залежності від цього викликати активують. Ну і не забудемо перед цим знайти контекст програми:

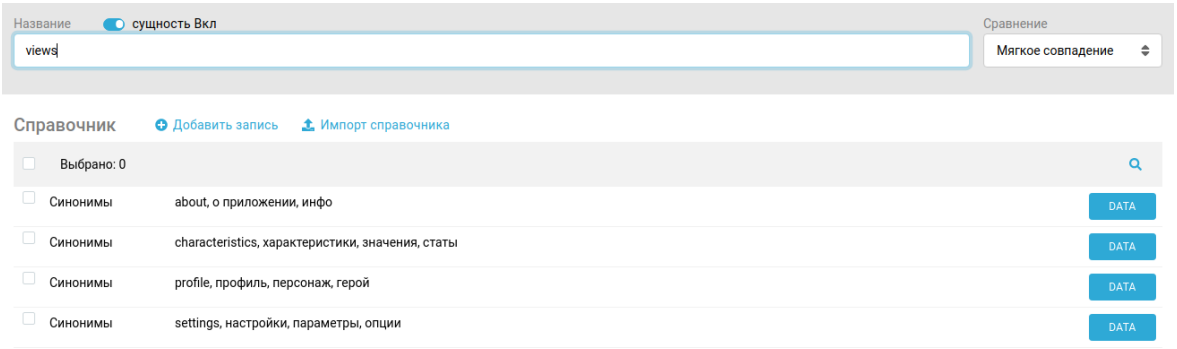
2.2.1. Заповнення форм

Навик ми будемо писати наJAICF (Це опенсорсний і абсолютно безкоштовний фреймворк для розробки голосових додатків від Just AI на Kotlin).

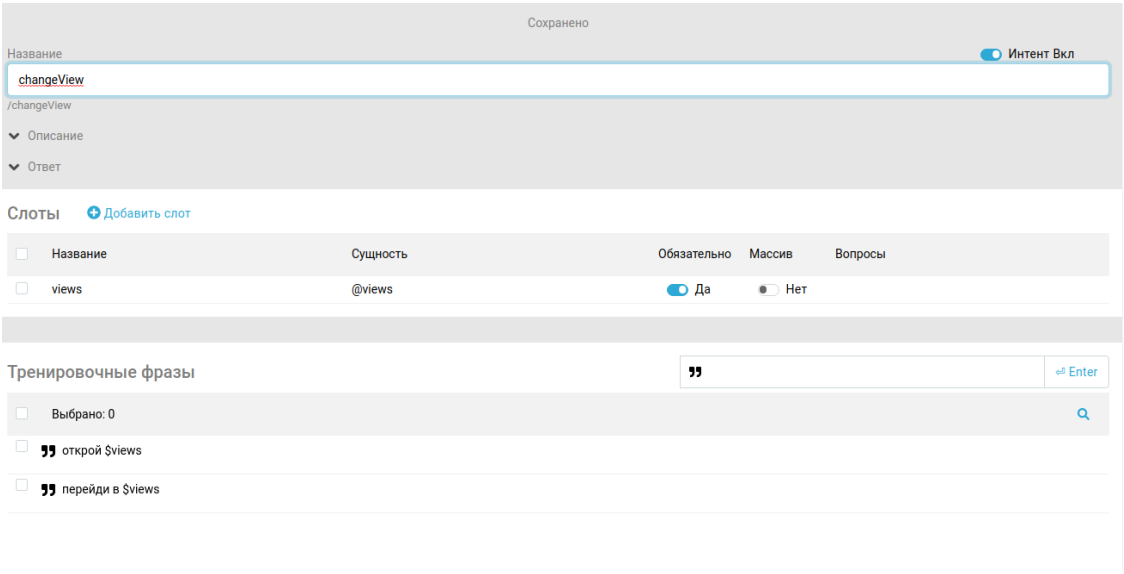
Для того, щоб користуватися NLU-функціоналом, підключаємо NLU-сервіс Caila - для цього реєструємося haapp.jaicp.com,в настройках знаходимо ключ API і прописуємо його в conf / jaicp.properties. Тепер ми можемо прямо в сценарії посилатися на ІНТЕНТ, які пропишемо haapp.jaicp.com.

Можна скористатися будь-яким іншим NLU-функціоналом або обійтися регулярними виразами - але для того, щоб зробити все красиво і просто для користувача, краще користуватися NLU.

Для початку заведемо ІНТЕНТ. Нам потрібно розпізнавати, що користувач хоче перейти в певний розділ програми. Для цього в сутності ми заводимо сутність під кожен з розділів, додаючи синоніми, і в DATA прописуємо то, як ми будемо розпізнавати це вже на рівні додатку (settings, characteristics, і т.д. з коду вище).

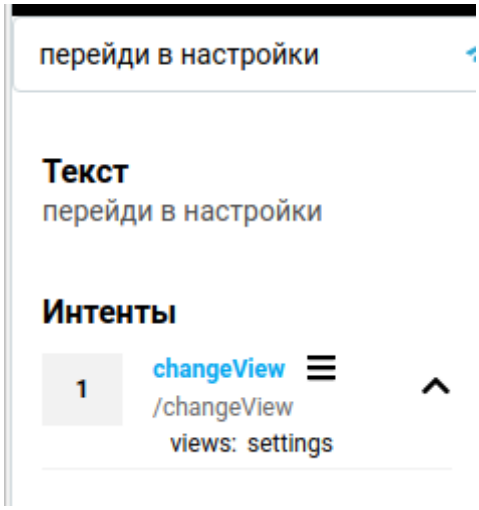


Далі прописуємо то, як саме ми очікуємо зустріти цю сутність у фразах користувача. Для цього створюємо Інтент і прописуємо там варіації фраз. Крім того, так як для переходу нам обов'язково потрібно знати, куди переходити, прописуємо, що зміст сутності views у фразі обов'язкове. У мене вийшло так.



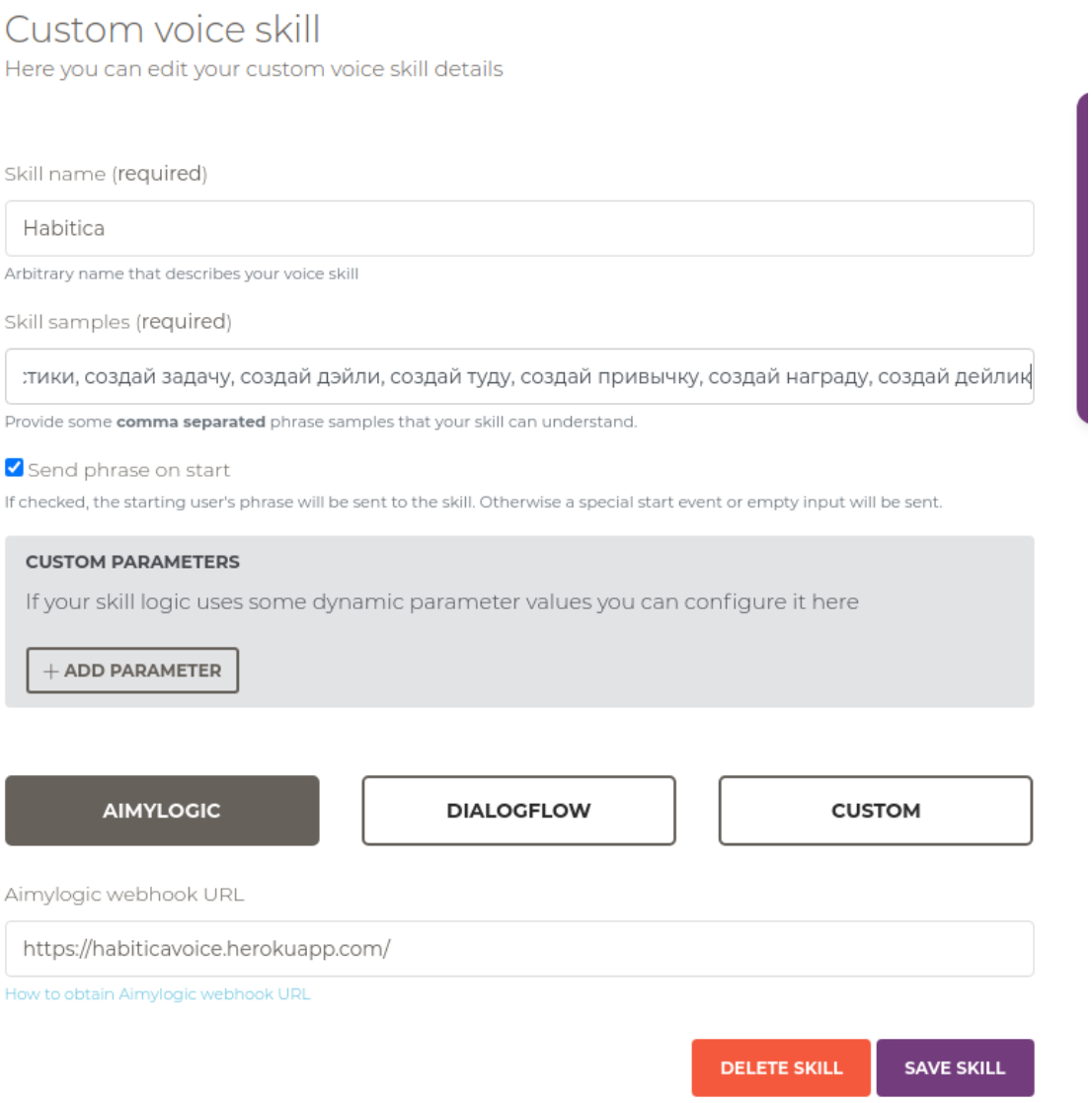
За назвою ми потім будемо відсилати до цього інтент в коді JAICF.

Щоб упевнитися, що інтент розпізнаються як треба, можна відразу ввести кілька тест-фраз по кнопці «Тестування».



Сценарій: викликаємо скилл

Скіли краще виносити в окремий пакет skills з Фаліка класу під кожен скілл.Дальше варіантів кілька. Можна підняти бота локально через ngrok, можна скористатися heroku. Отриману посилання прокидає в app.aimybox.com, через створення там кастомними навички, в поле Aimylogic webhook URL. В приклади пишемо пару прикладів виклику: відкрий настройки, відкрий інфо.



Після підключення каналу можна перевірити видачу прямо в консолі, щоб відловити баги, по кнопці Try in Action.

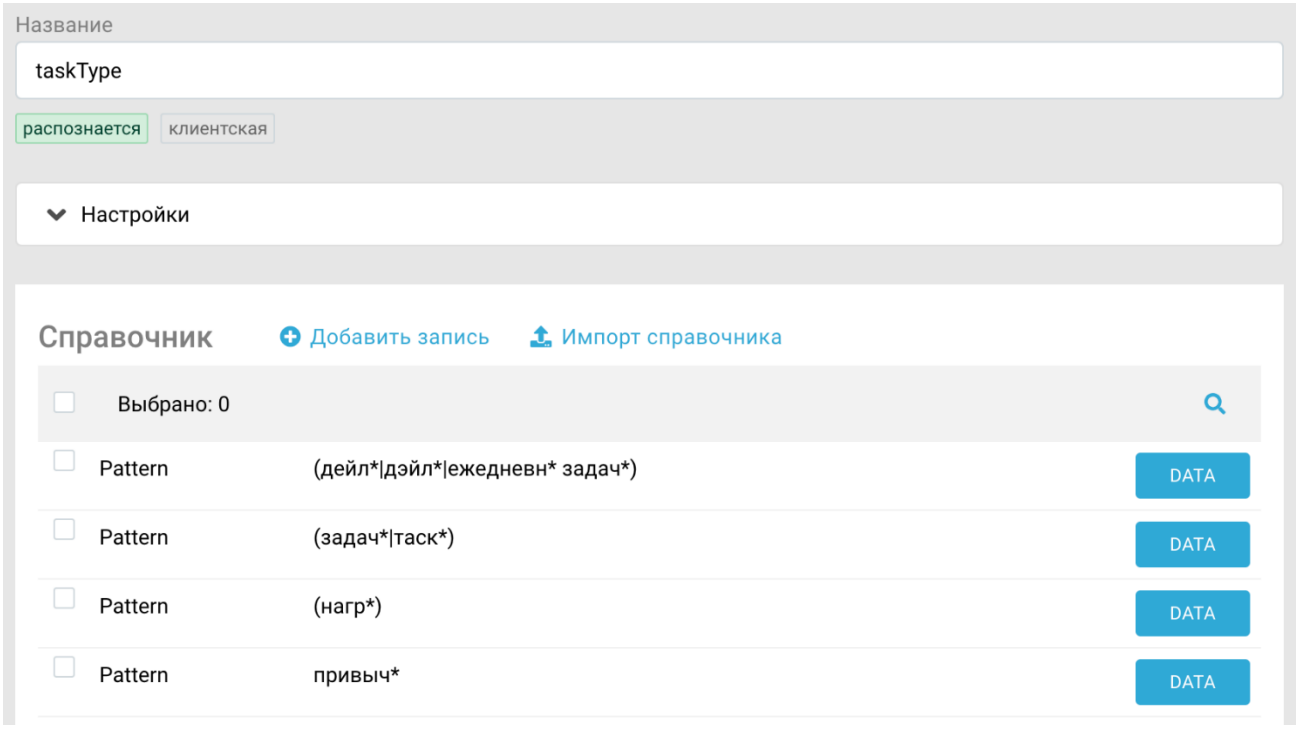
Можна підключити скилл безпосередньо, без консолі і додаткових навичок.

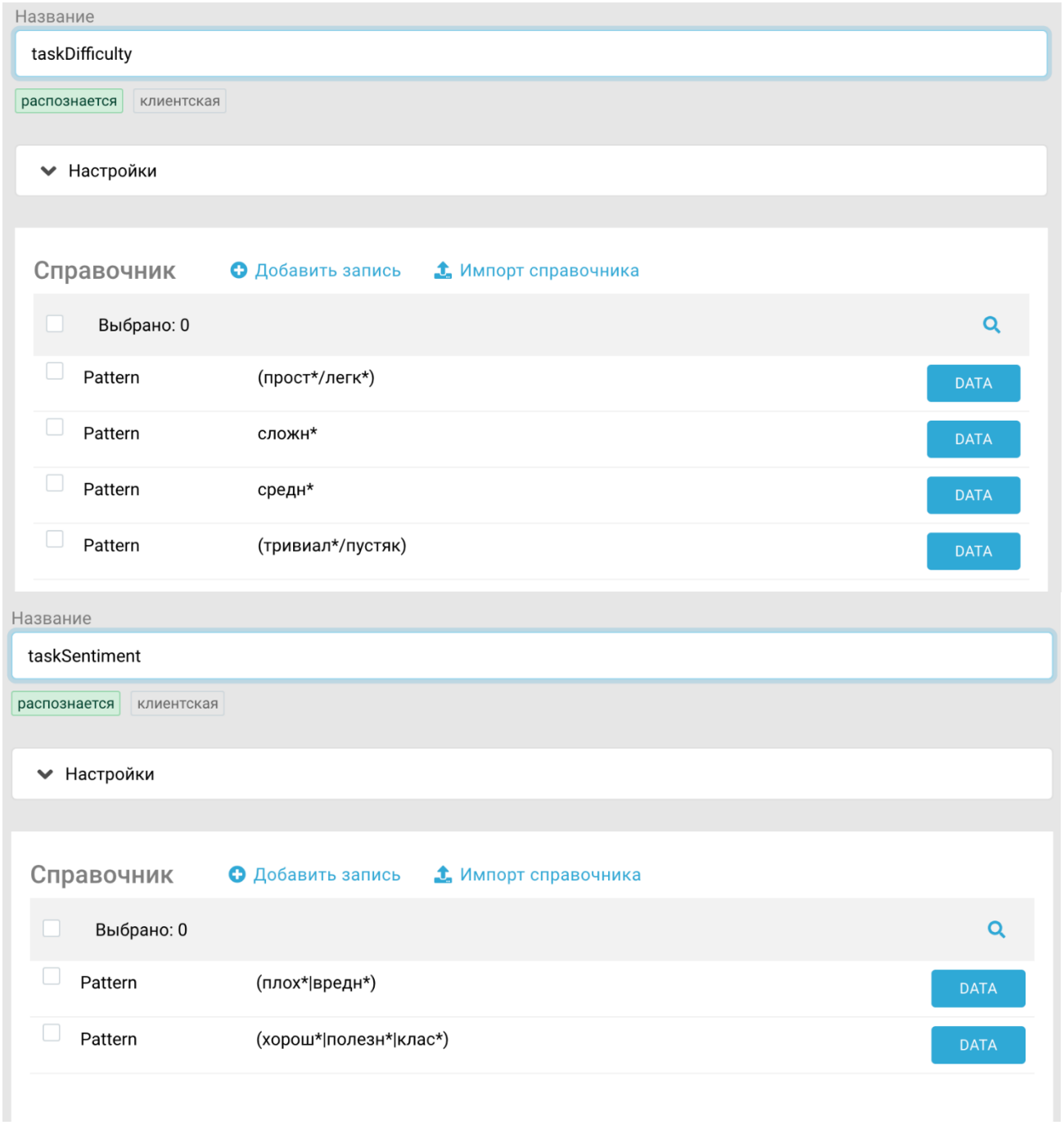
2.2.2. Заповнюємо завдання голосом

Хочеться однією командою заповнити завдання, перевірити, що все правильно, виправити якісь невеликі помилки (все-таки розпізнавання не завжди працює ідеально), і тільки після цього створити її остаточно.

Для цього зробимо другий скилл. Будемо відрізняти його від першого через response.action == "createTask", а те, який саме тип завдання створюється через response.intent.

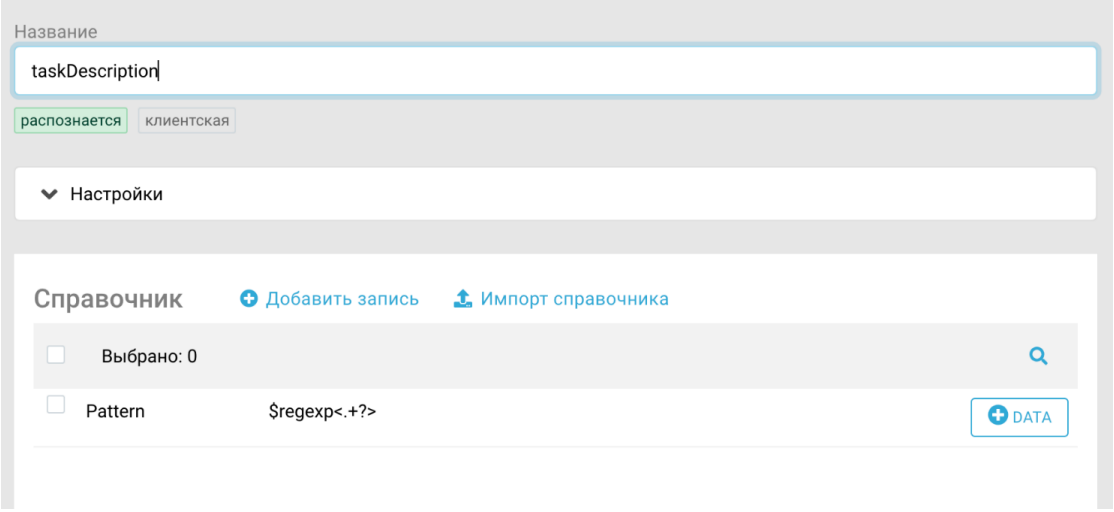
Вивчивши сорци додатки, розумієш, що і нагороди, і дейлікі, і звички, і завдання створюються через TaskFormActivity, просто з різними типами. Для початку пропишемо цю логіку.



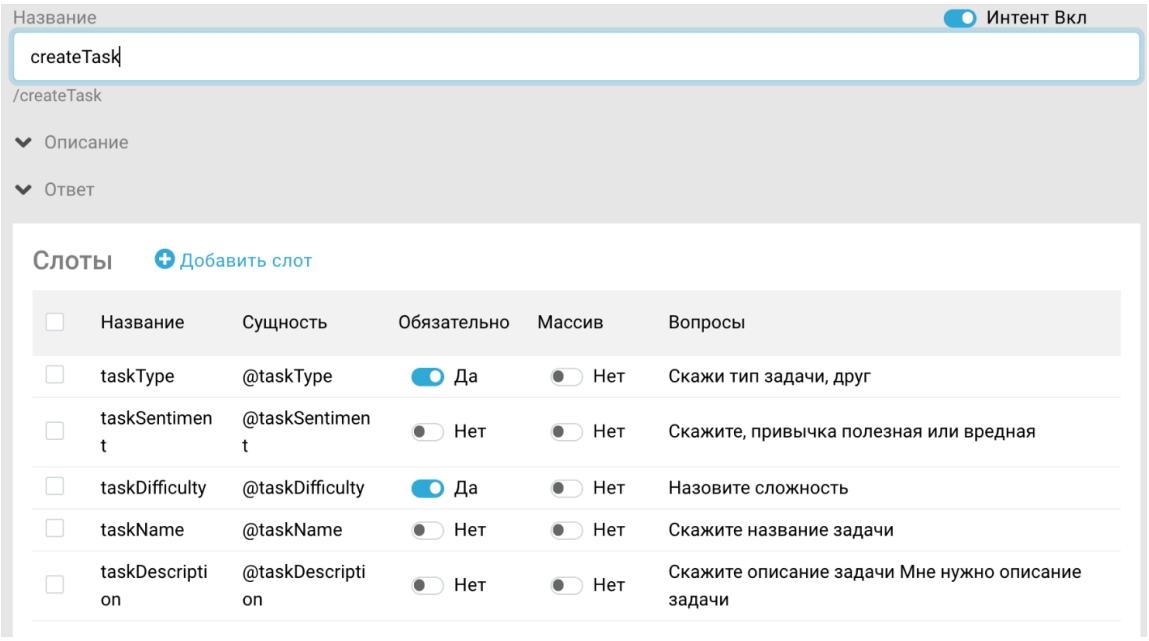


Не забуваємо в data прописати дані, які ми будемо обробляти на стороні клієнта - habit, pattern і так далі.

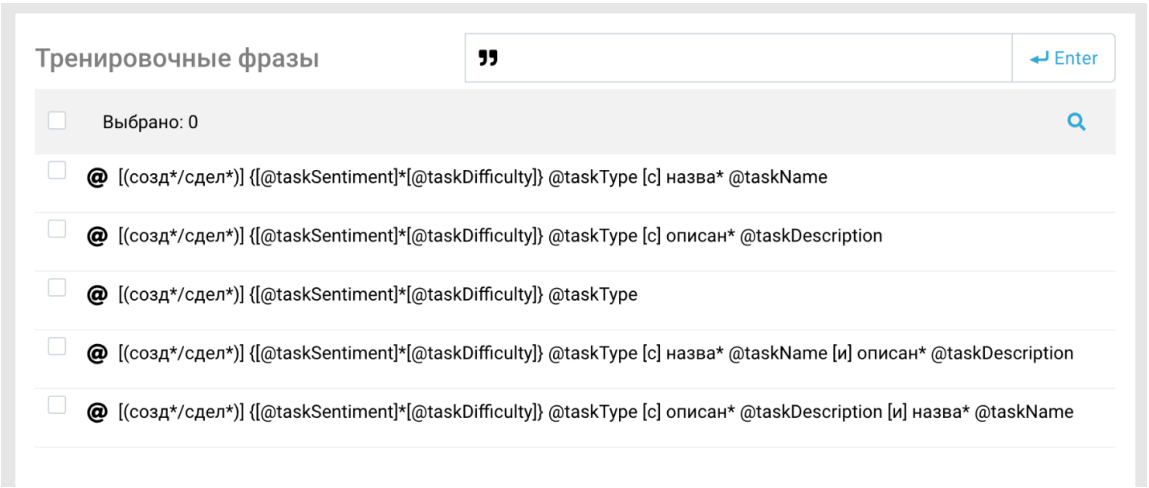
Так як назва і опис може бути будь-яким, створимо суті Name і Description, в якій пропишемо регулярний вираз, матчащее будь-яке слово. Поки що у нас в назві і описі буде по одному слову.



Робимо Інтент:



Зазначаємо, що нам обов'язково потрібен task_type і складність. Чи можемо додати в обов'язкові і назва, і опис - тоді, якщо користувач не скаже одне або інше, бот уточнить у нього за допомогою питання слот, який ще невідомий.



Прописуємо різні варіації того, як можна задати назву і опис разом з типом (порядок, відсутність одного або іншого). Тут немає межі досконалості, але для мінімуму необхідний шаблонів вище.

Підключаємо Інтент через активатор, записуємо з отриманих слотів тип в intent, назва і опис в data, і не забуваємо проставити action, щоб Aimybox з клієнтської сторони знав, який скилл вибрати.

2.3. Висновки до розділу

У розмовному додатку канал взаємодії з користувачем будується через розмову: Усний - з розумною колонкою, або через письмовий, наприклад, з Google Assistant. Крім колонки, пристроєм взаємодії може бути екран, тому розмовні програми ще й графічні.

У голосових додатків є важлива перевага перед мобільними: їх не треба завантажувати та встановлювати. Досить знати назву, і асистент сам все запустить.

Все тому, що нема чого завантажувати - і розпізнавання мови, і бізнес-логіка - все додаток живе в хмарі. Це величезна перевага перед мобільними додатками.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ ПРОГРАМНОГО МОДУЛЮ ГОЛОСОВОГО АСИСТЕНТА ТА ЙОГО ТЕСТУВАННЯ

3.1. Обґрунтування вибору інструментів розробки

Головна мета - спроектувати взаємодію між користувачем та програмою. У мобільному світі цей етап називається дизайном. Якщо дизайнер графічних додатків малює карти екранів, кнопки, форми і підбирає кольорову гаму, то VUI-дизайнер опрацьовує діалог між користувачем і додатком: прописує різні гілки діалогу, думає про розвилках і побічних сценаріях, вибирає варіанти фраз.

Проектування ведеться в три етапи.

– Приклади діалогів;

– побудова блок-схеми;

– складання prompt lists.

3.1.1. Приклади діалогів

Перше, що треба зробити - це зрозуміти, як буде працювати додаток. Розуміння і бачення потрібно транслювати на всіх інших, особливо, якщо ви аутсорс-компанія, і вам доведеться пояснювати замовнику, що він в результаті отримає.

Потужний інструмент в допомогу - приклади діалогів: розмова між користувачем і додатком за ролями, як в п'єсі.

Приклад діалогу для нашої гри.

Додаток вітається, розповідає користувачеві про правила, пропонує зіграти, і, якщо людина погоджується, показує картку з товаром, щоб користувач вгадав ціну.

Сценарій допомагає швидко зрозуміти, як буде працювати додаток, що воно зможе зробити, але, крім цього, приклади діалогів допомагають відсіяти головну помилку в світі голосових інтерфейсів - роботу над неправильними сценаріями.

Є просте правило: якщо не можеш уявити, як ти промовляти сценарій з іншою людиною, то не варто над ним працювати.

Голос і графіка істотно відрізняються, і не всі, що працює на графічних інтерфейсах, добре працює на голосі. Майже в кожному мобільному додатку є реєстрація, але я не можу уявити, як можна зареєструватися голосом? Як диктувати розумної колонці пароль: «Велика буква, маленька буква, ес як долар ...» - і все це вголос. А якщо я не один, а на роботі? Це приклад помилкового сценарію. Якщо ви почнете розробку сценарію з помилкою, то з ним виникнуть проблеми: ви не зрозумієте, як його виконати, користувачі не зрозуміють як ним користуватися.

Приклади діалогів допоможуть знайти подібні моменти. Щоб знайти помилки в сценаріях, запишіть діалог, виберете колегу, посадіть навпроти і

відіграв ролі: ви - користувач, колега - додаток. Після рольового зачитування діалогу стане ясно, звучить додаток чи ні, і чи буде користувачеві зручно.

Така проблема буде з'являтися постійно. Якщо у вас in-house розробка, то виникне спокуса: «У нас вже є сайт, давайте просто сконвертіруем його в голос і все буде добре!» Або прийде замовник і скаже: «Ось мобільний додаток. Зробіть те ж саме, тільки голосом! » Але так робити не можна. Ви, як фахівець, повинні швидко знаходити сценарії, над якими не варто працювати, і пояснювати замовнику чому. Приклади діалогів тут допоможуть.

Для прописування діалогів підійде абсолютно будь-який текстовий редактор, до якого ви звикли. Головне - запишіть текст і зачитайте його за ролями.

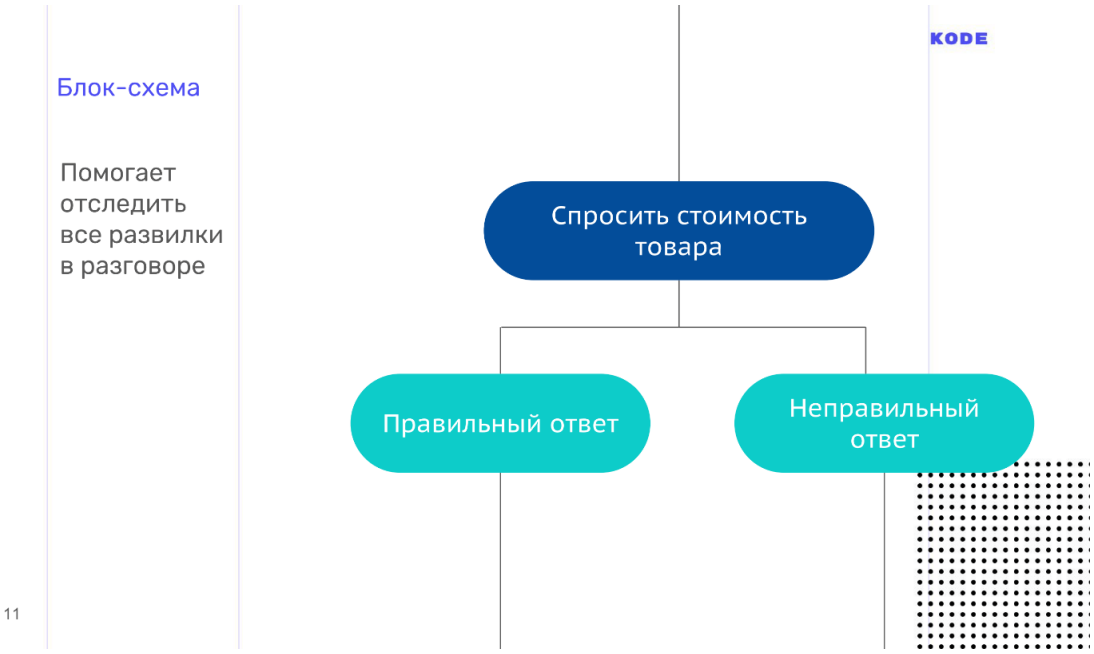
Блок-схема

Приклади діалогів - це потужний, швидкий і дешевий інструмент, але вони описують тільки лінійний розвиток подій, а розмови завжди нелінійні. Наприклад, в нашій грі «Вгадай ціну», користувач може відповісти на питання правильно чи неправильно - це перша розвилка з безлічі тих, що будуть зустрічатися далі.

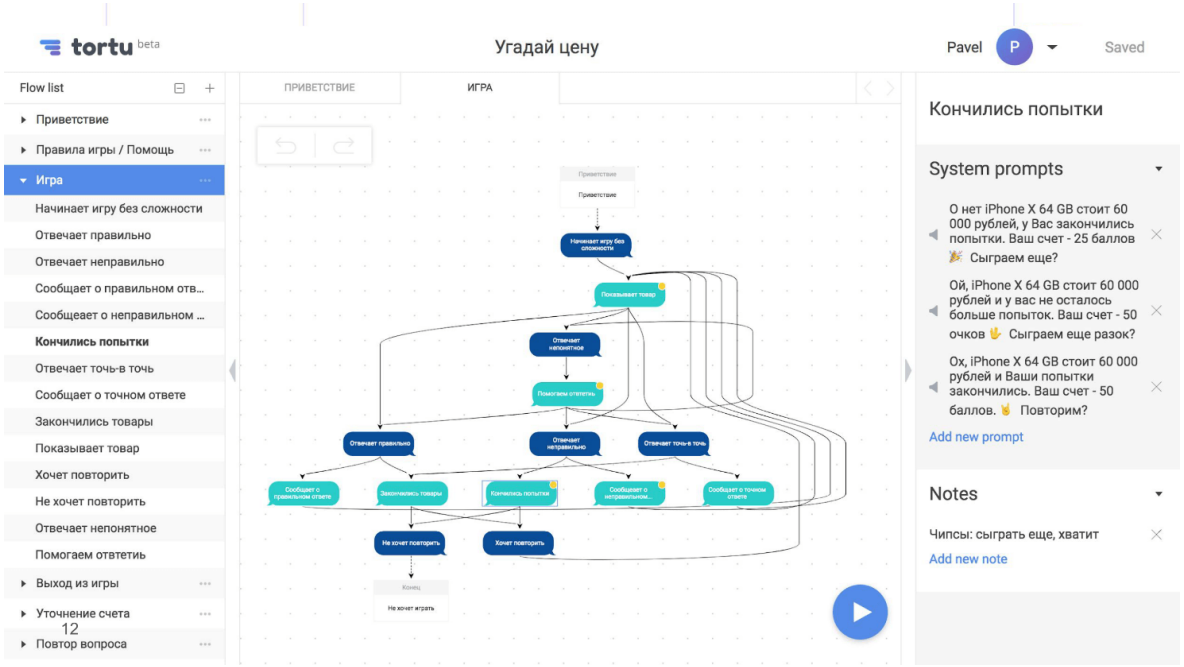
Щоб не заплутатися у всіх гілках діалогу вашої програми, складіть блок-схему - візуалізацію діалогу. Вона складається всього з двох елементів:

Крок діалогу від імені користувача.

Крок діалогу від імені програми.



Блок-схема - це карта нашого застосування, але з однією неприємною властивістю - вона сильно розростається, стає нечитаною і візуально незрозумілою. Ось, наприклад, скріншот з частиною блок-схеми з сценарію, де користувач вгадує ціну, з декількома розвилками.



Кілька розвилок - ще не межа, їх може бути десятки або сотні. Ми ставили перед собою запитання: «Що станеться, якщо людина відповість правильно? А якщо ні? Що буде, якщо закінчаться спроби? Що якщо закінчаться товари? А якщо він вгадає ціну точно? Що якщо інтернет пропаде на цьому кроці або на іншому? » У підсумку ми створили величезну нечитаемую схему.

В цьому ми не самотні. Я спілкувався з дизайнером з США, яка працювала над серйозним проектом. У проекті був і IVR, і банк, і skill одночасно, і все це роздуло блок-схему до 600 аркушів. До кінця схему ніхто не розумів, а коли дизайнер її побачила, то просто жахнулася.

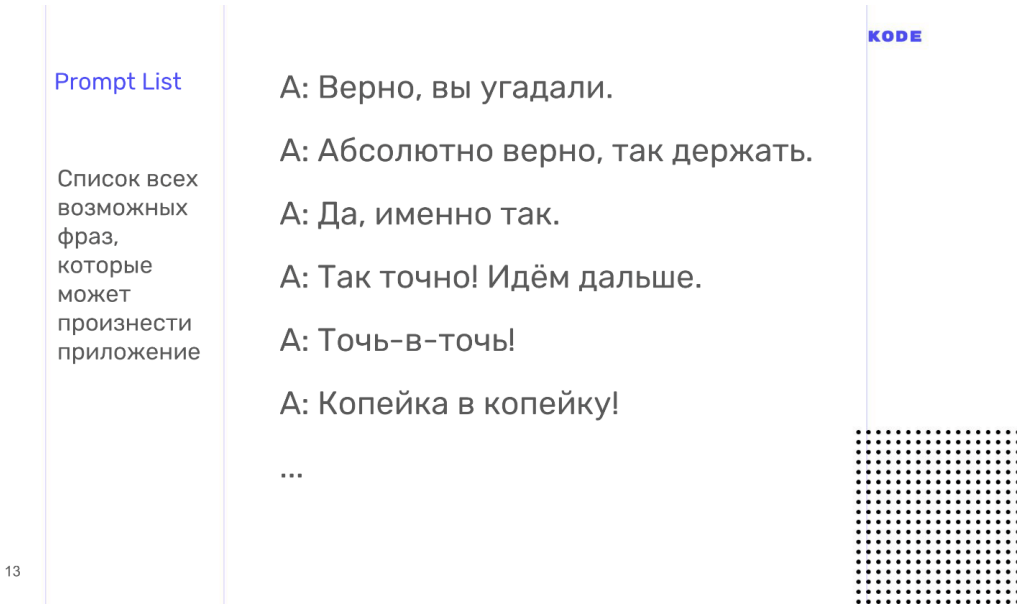
У мене є порада, як цього не допустити. Схема завжди буде розростатися, але ніколи не намагайтеся побудувати одну велику блок-схему на все додаток - вона буде громіздкою, і ніхто крім вас в ній не розбереться. Ідіть від зворотного і розбийте схему на логічні частини: окремо сценарій вгадування ціни, окремо сценарій допомоги. За необхідності розбийте і ці сценарії на подсценарії. В результаті вийде не одна велика карта з незрозумілими зв'язками, а багато маленьких, читабельних, добре пов'язаних між собою схем, в яких всім зручно орієнтуватися.

Для блок-схем підійде будь-який інструмент. Раніше я використовувавRealtimeBoard, а ще єDraw.ioі навіть XMind. В результаті розробив свій, тому що просто зручніше. На зображенні якраз він і представлений. Цей інструмент підтримує, в тому числі, розбивку на подсценарії.

prompt lists

Останній артефакт, який ми сформуємо на етапі проектування. prompt list - це список всіх можливих фраз, які може вимовити додаток.

Тут є одна тонкість. Розмова з додатком повинен бути гнучким і схожим на розмову з людиною. Це означає не тільки можливість пройти різними гілками, що ми робили на етапі блок-схеми, а звучання розмови в цілому. Людина ніколи не буде відповідати одній і тій же фразою, якщо ви будете задавати один і той же питання. Відповідь завжди буде перефразувавши і звучати якось інакше. Додаток має чинити так само, тому на кожен крок діалогу від імені додатки напишіть не один варіант відповіді, а як мінімум п'ять.



За prompt листам є ще важлива річ. Спілкування має бути не тільки живим і гнучким, а й консистентним в плані стилю мови і загального відчуття, якою ви спілкуєтесь з вашим додатком. Для цього дизайнери використовують відмінний прийом - створення персонажа. Коли я дзвоню своєму другові, то не бачу його, але підсвідомо уявляю співрозмовника. У користувача при спілкуванні з розумною колонкою те ж саме. Це називаєтьсяпарейдалія.

На етапі prompt листів ви створюєте персонажа, від імені якого додаток буде розмовляти. З персонажем ваші користувачі будуть асоціювати бренд і додаток - це може бути реальна людина або вигаданий. Опрацюйте для нього зовнішність, біографію, характер і гумор, але якщо часу немає, то просто приведіть всі ваші фрази в prompt листах до єдиного стилю. Якщо ви почали звертатися до користувача на «Ви», то не звертайтеся в інших місцях на «Ти». Якщо у вас неформальний стиль спілкування, то дотримуйтеся його всюди.

Зазвичай для створення prompt листів використовують Excel або Google-таблиці, але при з ними виникають величезні тимчасові втрати на рутинну роботу. Блок-схема і табличка з фразами ніяк не пов'язані один з одним, будь-які правки доводиться переносити вручну, що виливається в постійну і довгу рутину.

Я використовую НЕ Excel, а свій інструмент, тому що в ньому все фрази пишуться прямо в блок-схемі, вони закріплені за кроком діалогу. Це позбавляє від рутини.

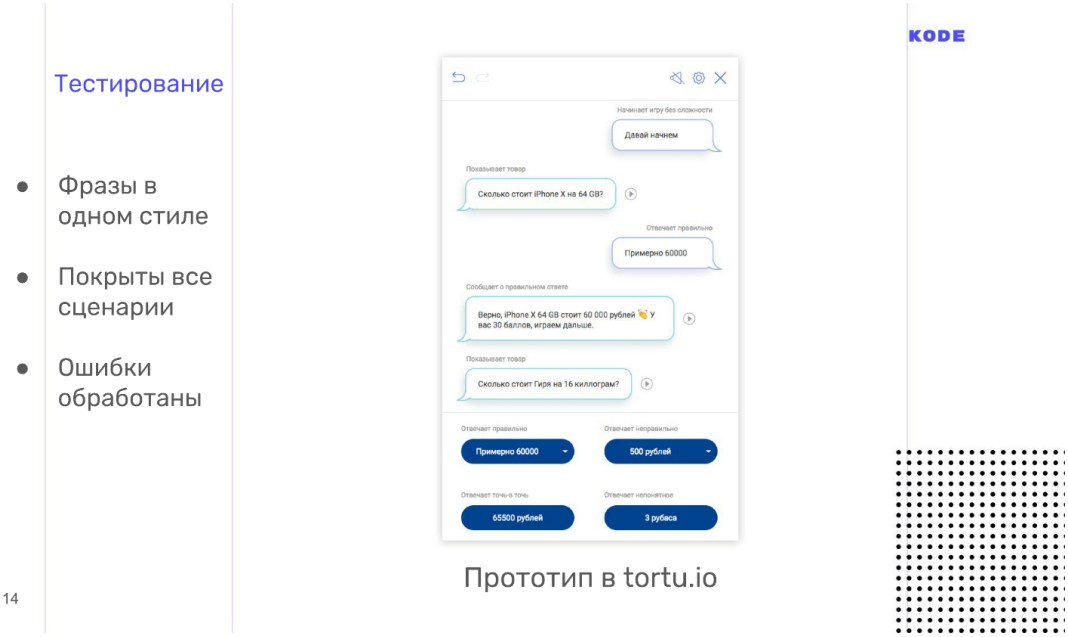
У проектуванні ми опрацьовуємо кожен сценарій: пишемо приклад діалогу, знаходимо побічні гілки, помилки, покриваємо це блок-схемою, а потім працюємо над стилем мови і фразами.

Здається, що тепер все готово і можна дати завдання розробникам і прийматися за код, але залишився ще один важливий етап - тестування. Ми повинні упевнитися, що як дизайнери все зробили правильно, що додаток буде працювати як ми хочемо, що всі фрази в одному стилі, що ми покрили всі побічні гілки і обробили всі помилки.

тестування

Тестування на такому ранньому етапі особливо важливо для голосових додатків. У світі графічних інтерфейсів користувач обмежений тим, що намалював дизайнер: він не зайде за межі екрану, не знайде кнопку, якої не існує, а буде натискати тільки на те, що є ...

У світі голосу все не так: користувач може говорити все, що завгодно, і ви не знаєте, як він почне працювати з вашим додатком, поки це не побачите. Краще це зробити на ранньому етапі проектування і підготуватися до несподіванок, поки не почалася дорога технологія.



Додатки тестуються за допомогою методології Wizard of Oz. Вона використовується і в графічних додатках, але рідше, а в голосі це must have. Це

метод, коли користувач взаємодіє з системою, припускаючи, що вона існує і працює сама по собі, але всім процесом керуєте ви.

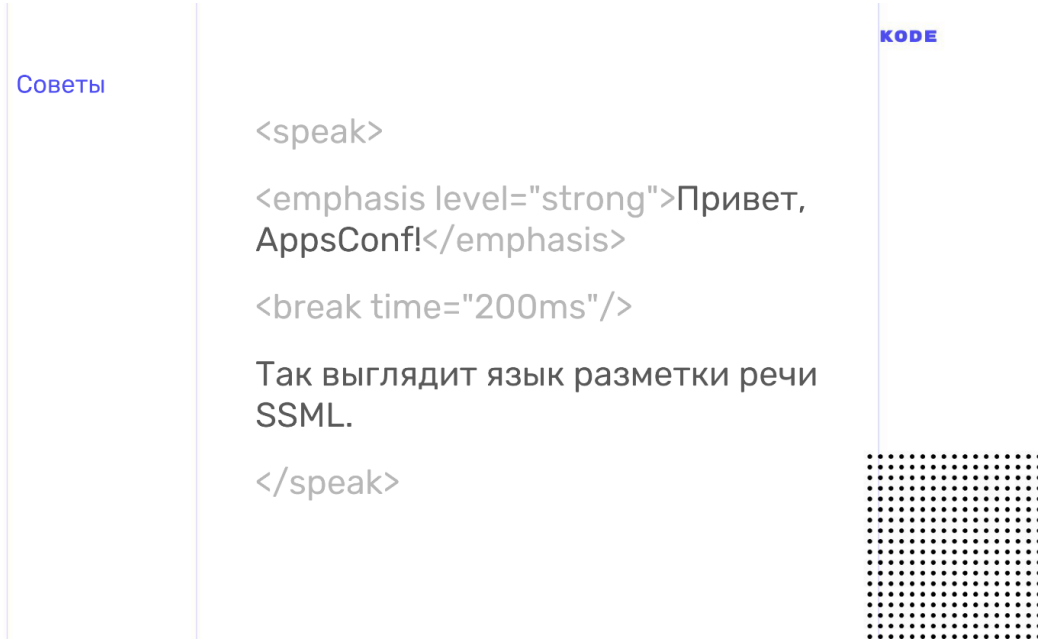
Тестування проводиться за допомогою інтерактивних прототипів. Зазвичай дизайнеру доводиться просити розробників створити прототип, але особисто я використовую свій інструмент, тому що в ньому все робиться в один клік і не треба нікого чекати. Ще нам потрібен користувач. Ми кличемо людини, який ніяк не залучений в розробку, нічого не знає про програму і, в ідеалі, входить в вашу ЦА. Надіслано запрошення, пояснюєте, що це за додаток, як ним користуватися, садите в кімнаті, включаєте інтерактивний прототип і користувач починає з ним розмовляти. Прототип не розпізнає мову, а це ви чуєте, що говорить людина, і вибираєте варіант відповіді, яким додаток відповідає на кожну фразу.

Якщо користувач не бачить екран, то йому здається, що додаток працює саме по собі, але процесом керуєте ви. Це і є тестування Wizard of Oz. З його допомогою ви не тільки почуєте, як звучить додаток, але і побачите, як люди його використовують. Гарантую, що ви знайдете багато непокритих сценаріїв.

Багато і довго тестируйте до розробки, поки не будете впевнені в тому, що додаток працює і користувачі взаємодіють з ним так, як ви очікуєте.

На цьому етап проектування закінчується і у нас на руках є приклади діалогів, блок схема - логічне опис роботи програми, і prompt-листи - то, що додаток говорить. Все це ми віддамо розробникам. Перед тим, як я розповім як розробники створюють додатки, поділюся порадами з проектування.

Використовуйте мову розмітки SSML- як HTML, тільки для мови. SSML дозволяє проставити паузи, виставити рівень емпатії, наголос, прописати, що прочитати по буквах і де зробити акцент.



Розмічена мова звучить набагато краще, ніж роботичних, а чим краще звучить додаток, тим приємніше їм користуватися. Тому використовуйте SSML - він не такий вже й складний.

Думайте про моментах, в яких користувачі звертаються до вашого додатком за допомогою.Для голосу це особливо важливо. Людина може розмовляти з колонкою на самоті в кімнаті, а може їхати в автобусі і спілкуватися зі смартфоном. Це два принципово різні сценарії поведінки для голосового додатки. Подібна ситуація у нас була з банківським додатком. У додатку був сценарій, коли користувач отримує інформацію про рахунок, а це приватна інформація. Я подумав - якщо людина розмовляє вдома, то все нормально, але, якщо він їде в автобусі, а додаток почне озвучувати баланс карти вголос - буде некрасиво.

Подумавши про таких моментах, ви можете визначити, що якщо користувач розмовляє зі смартфоном, нехай навіть голосом, то приватну інформацію краще не зачитувати вголос, а показати на екрані.

Використовуйте Мультимодальний дизайн.

Це дизайн під різні поверхні і платформи. Голосові пристрої дуже різні за своєю фактурою. У мобільному світі пристрої розрізняються лише платформою і розміром екрана - форм-фактором. З голосом все інакше. Наприклад, у колонки взагалі немає екрану - тільки голос. У смартфона є екран, і його можна тапа

пальцем. У телевізора екран величезний, але до нього марно торкатися. Подумайте, як ваш додаток буде працювати на кожній з таких поверхонь.

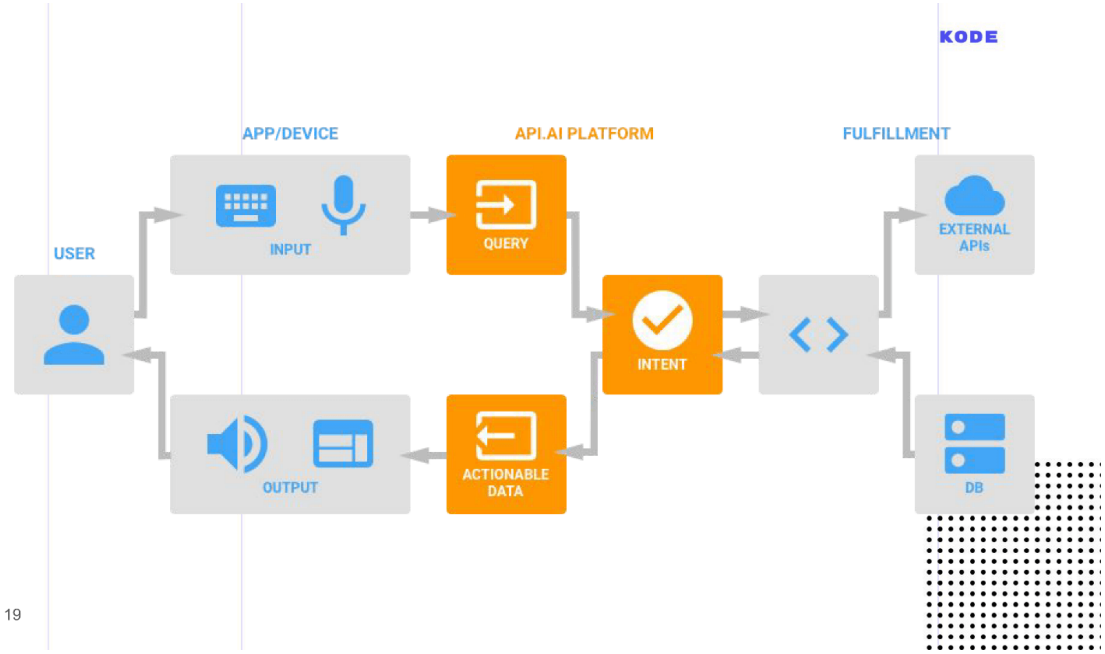
Наприклад, користувач здійснив покупку і ми хочемо показати чек. Зачитувати чек вголос - погана ідея, тому що інформації багато і ніхто її не запам'ятає, так як голосова інформація сприймається складно і важко.

Використовуючи принцип мультимодального дизайну, ми розуміємо, що якщо є екран, то чек краще показати, замість того, щоб читати. Якщо екрану немає, то ми змушені промовити вголос основні дані чека.

На цьому проектування закінчено. Те, що я розповів - це основи, верхівка айсберга. Для самостійного вивчення проектування, я зібрав багато матеріалу з проектування, в кінці статті будуть посилання.

Розробка

Розмова почнемо з універсальної схеми роботи програми під будь-якою платформою. Схема працює і з Алісою, і з Amazon Alexa, і з Google Assistant.



Коли користувач просить запустити наш додаток, асистент це робить і передає додатком контроль за розмовою. Користувач щось говорить і в додаток потрапляє сирий текст, який проходить обробку системою розпізнавання мови.

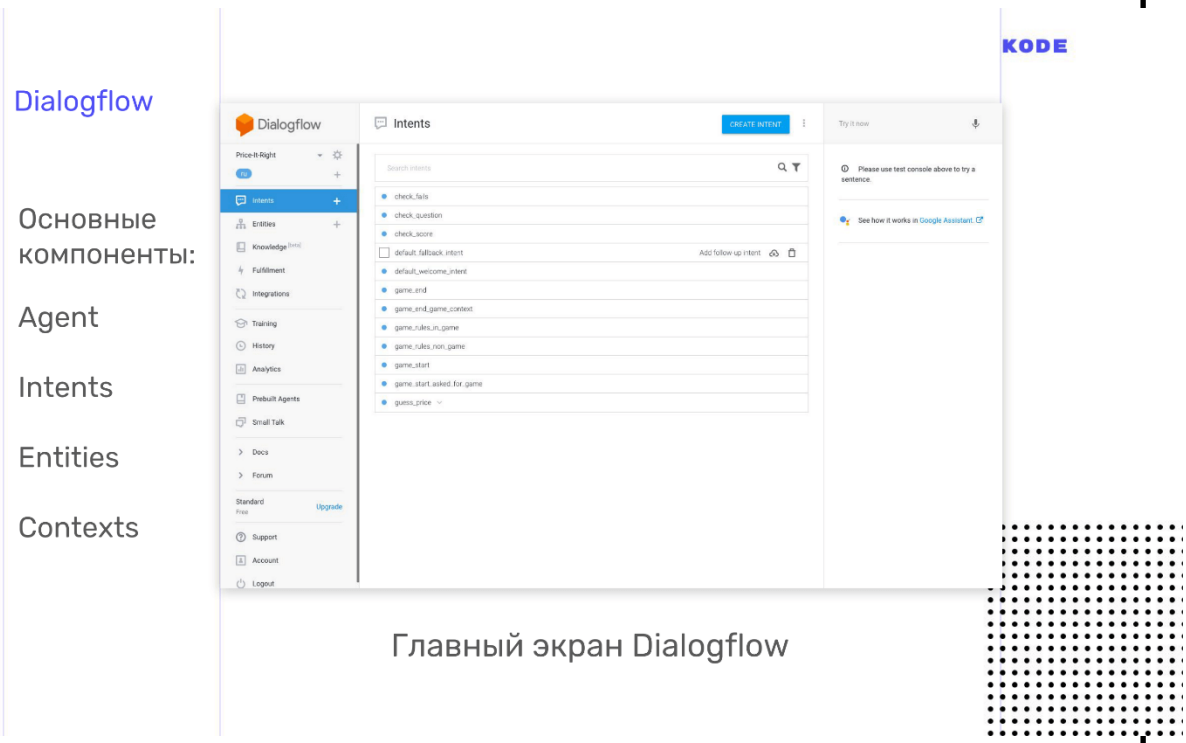
Система з сирого тексту визначає намір користувача - intent, його параметри - слоти, і формує відповідь: відразу, якщо не потрібно ніякої логіки і додаткової інформації, і використовує webhook, якщо логіка необхідна. Вся бізнес-логіка в голосових додатках лежить в webhook, звідти виробляються запити в бази даних, виклики API.

Відповідь формується і передається користувачеві голосом або показується на екрані.

Для обробки ми використовуємо Dialogflow, його структура абсолютно така ж, як і у інших систем розуміння мови, і розробку будемо розглядати на його прикладі.

Ми переходимо до першій ланці - до системи розуміння людської мови або Natural Language Understanding - NLU.

Dialogflow



Главный экран Dialogflow

Ми використовуємо Dialogflow, тому що у нього багаті можливості, хороша документація, жива підтримка і його просто і швидко освоїти. Dialogflow багатоплатформовий інструмент: основна кваліфікація - додатки для Google

Assistant, але для Яндекс-Аліси, Amazon Alexa і створення ботів в Telegram його теж можна використовувати. Окремий плюс - відкрите API. Ви можете використовувати систему, щоб розробити голосове управління для сайту або вже існуючого мобільного додатка.

Основні компоненти Dialogflow.

Agent- ваш проект, то, що у нього всередині.

Intents- наміри користувача.

Entities- об'єкти даних.

Contexts- контексти, сховища для інформації.

Пройдемося за всіма компонентами, але почнемо з головного - це Intents.

Intents

Це призначене для користувача намір, то, що користувач хоче зробити. Намір виражається фразами. Наприклад, в грі користувач хоче дізнатися правила гри і каже: «Розкажи правила гри», «Скажи, як грати?», «Допоможи мені - я заплутався» або що-небудь в такому дусі. Відповідно, ми створюємо окремий Intent для правил гри, і на вхід пишемо всі ці фрази, які очікуємо від користувача.

Я раджу писати 10 і більше варіантів фраз. В такому випадку розпізнавання мови буде працювати краще, тому що в Dialogflow використовується нейронна мережа, яка приймає ці 10 фраз на вхід, і генерує з них купу інших, схожих. Чим більше варіантів, тим краще, але не переборщіть.

У Intent повинен бути відповідь на будь-яке питання користувача. У Dialogflow відповідь можна сформувати без застосування логіки, а якщо логіка потрібна, то передаємо відповідь з webhook. Відповіді можуть бути різні, але стандартний - це текст: озвучується на колонках, показується чи проговорюється на смартфонах.

Залежно від платформи доступні додаткові «плюшки» - графічні елементи. Наприклад, для Google Assistant це кнопки, картки, списки, каруселі. Вони

показуються тільки якщо людина говорить з Google Assistant на смартфоні, телевізорі або іншому подібному пристрої.

3.2. Запуск Intent в голосовому помічнику

Intent запускається, коли користувач щось говорить. У цей момент передається інформація - параметри, які називаються слотами, а тип даних параметрів - Entities. Наприклад, для нашої гри, це приклади фраз, які говорить користувач, коли вгадує ціну. Тут є два параметри: сума і валюта.

Параметри можуть бути обов'язковими і необов'язковими. Якщо користувач відповість «дві тисячі», то фрази буде досить. За замовчуванням ми візьмемо рублі, тому валюта - необов'язковий параметр. Але без суми ми не зможемо зрозуміти відповідь, тому що користувач може відповісти на питання не конкретно:

- Скільки коштує? - Багато!

Для таких випадків в Dialogflow є поняття re-prompt - це фраза, яка буде виголошена, коли користувач не назвав обов'язковий параметр. Для кожного параметра фраза задається окремо. Для суми це може бути щось на зразок: «Назвіть точну цифру, скільки коштує ...»

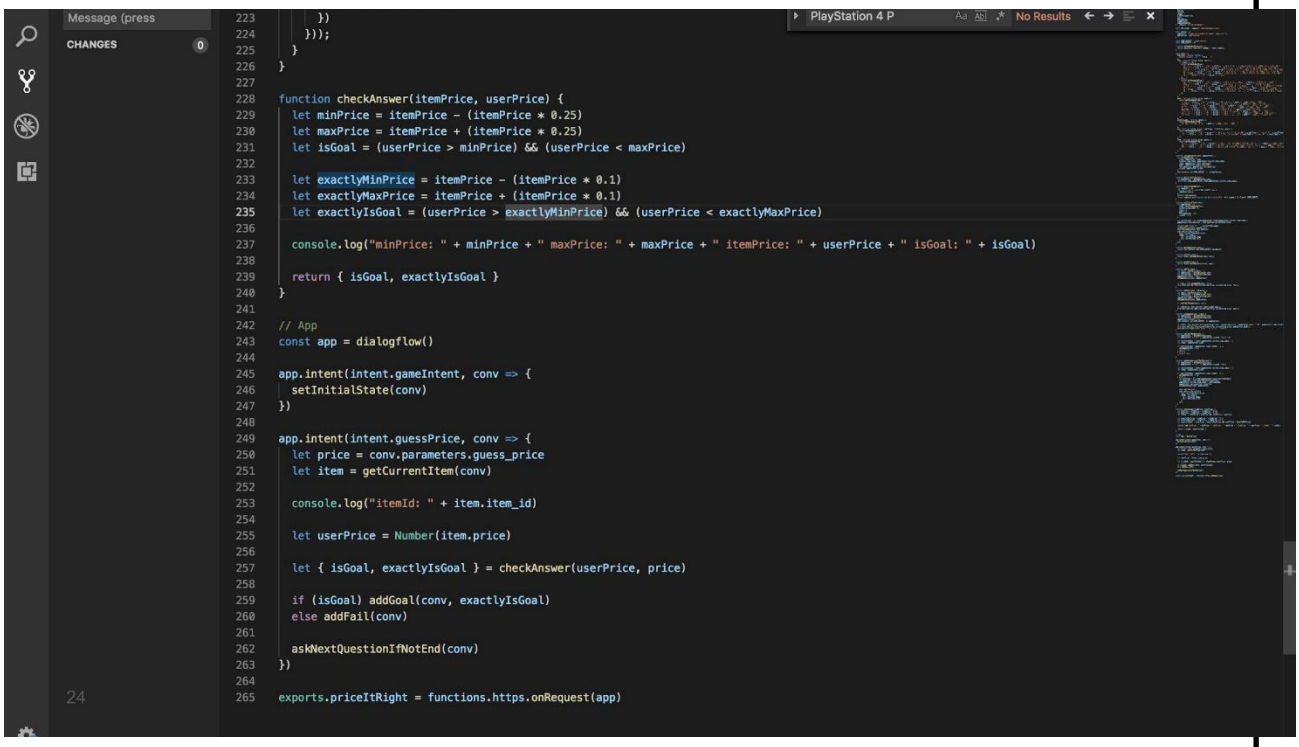
У кожного параметра обов'язково повинен бути тип даних - Entities. У Dialogflow багато стандартних типів даних - міста, імена, і це рятує. Система сама визначає, що є ім'я, що є число, а що місто, але можна ставити і свої кастомниє типи. Валюта в Dialogflow - кастомний тип. Ми створили її самі ^ описали технічне системне назву, яким будемо користуватися, і синоніми, які відповідають цим параметром. Для валюти це рубль, долар, євро. Коли користувач говорить: «Євро», то Dialogflow підсвічує, що це наш параметр «валюта»

3.3.Контекстні зв’язки у голосовому помічнику

Сприймайте це слово буквально: context - це контекст того, про що ви говорите з користувачем. Наприклад, у асистента можна запитати: «Хто написав Муму?» і він відповість, що це Тургенєв. Навздогін можна запитати, коли він народився. Звертаю увагу, що ми запитуємо: «Коли він народився», не уточнюючи хто. Google зрозуміє, тому що пам'ятає - в контексті розмови Тургенєв.

З технічної точки зору context - це сховище типу «ключ - значення», в яке складається інформація. Intent може або випускати context з себе, складаючи в нього щось, або приймати на вхід і діставати інформацію звідти. У context є час життя. Воно визначається кількістю кроків діалогу від останнього згадки: наприклад, через 5 кроків діалогу забути, що ми говорили про Тургенєва.

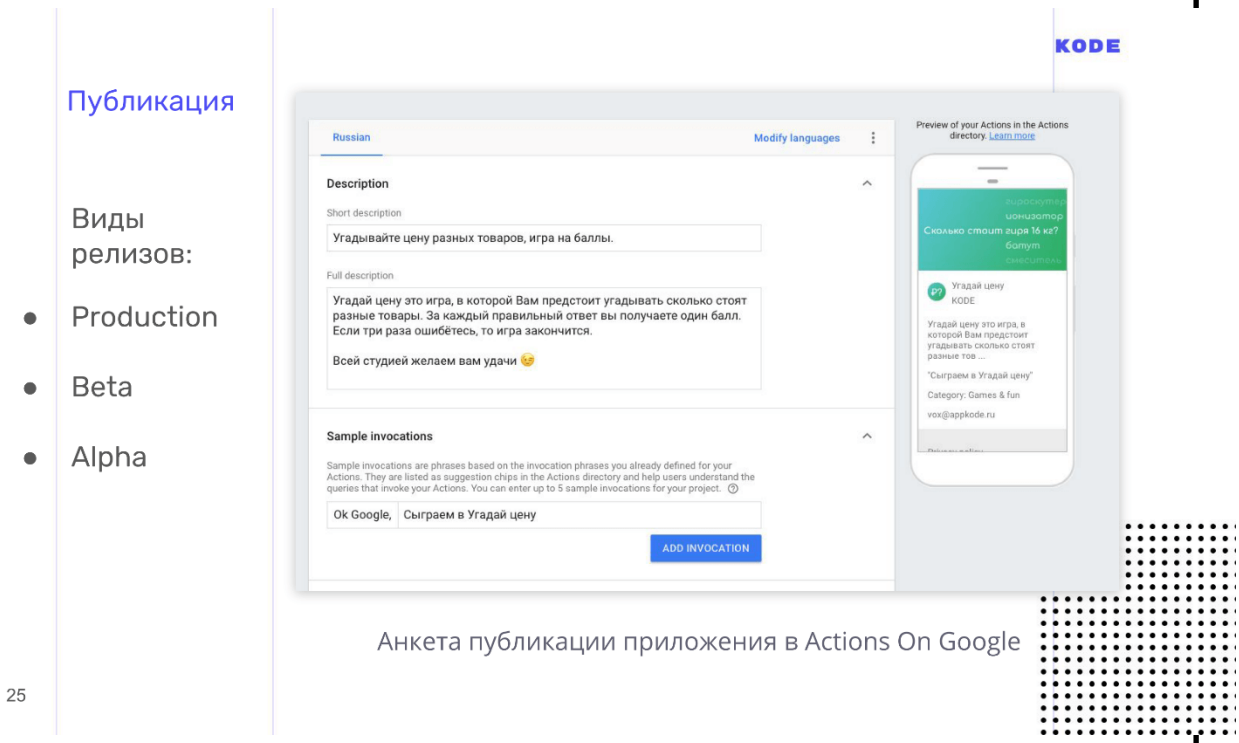
У context є ще одна важлива функція - він може допомогти нам розбити додаток на логічні зони: до уповноваженого і неавторизовану, на ігрову сесію чи ні. Розбивка будується так, що Intent, який приймає context на вхід, не може бути запущений без контексту і вимагає попереднього запуску іншого Intent. Так ми можемо логічно пов'язувати і будувати наше додаток.



Я згадував webhook.У Dialogflow є бібліотеки під абсолютно різні мови, ми використовували JS. У Google Assistant для webhook є обмеження - відповідь з нього повинен прийти не пізніше, ніж через 5 секунд, інакше випаде помилка і додаток спрацює в fallback. Для Аліси, час відповіді - 1,5 або 3 секунди.

Ми налаштували систему розуміння мови, написали webhook і у нас все працює, запустили QA, і тепер час для публікації.

3.4. Публікація голосового помічника



Анкета публикации приложения в Actions On Google

Публікація стандартна, майже як з мобільними додатками, але з парою нюансів.

Зверніть окрему, особливу і дуже пильну увагу на назву програми. Користувач буде його вимовляти вголос кожен раз при запуску. Тому назва повинна бути простим для вимови і простим для розпізнавання асистентом, тому що з цим іноді бувають проблеми.

У назви є два правила:

Не можна використовувати загальні фрази. У Яндекс.Алісе, не можна використовувати дієслова. Наприклад, ви не зможете взяти назву «Замовлення таксі», тому що це загальна фраза, яку багато хто хоче використовувати.

Якщо захочете використовувати назву своєї компанії, то будьте готові до того, що вас попросять підтвердити права на користування брендом.

В Google Assistant стандартна фраза, яка запускає будь-який додаток - «ОК, Google, поговори з ...». Ви можете скористатися цією фичей, наприклад, сказати: «ОК, Google, поговори з Uber» - і він запустить додаток на головному екрані, на стартовій точці. Але можна зробити так, щоб користувач сказав: «ОК, Google, скажи Uber забрати мене звідси і відвезти туди!» ми скорочуємо одну ітерацію, і користувач потрапить в потрібну дію.

Сценарій визначається фразами, якими запускається додаток. Вони встановлюються при публікації, але часто працюють некоректно - російською мовою точно. Наприклад, в нашій грі фраза «Давай зіграємо» працювала, а «Давай пограємо» не працювала. Я не знаю, в чому принципова різниця між «зіграємо» і «пограємо» для Google Assistant. Він розпізнавав обидві фрази коректно, але програма не працювало, хоча з англійською мовою у нас проблем не виникало.

В іншому публікація проходить гладко, без зайвих питань. Підтримка у Google Assistant дуже жива, відповідає швидко, і документація хороша.

Також хочу відзначити різні види релізів.

Alpha- для 20 осіб і без проходження рев'ю.

Beta- для 200 осіб.

Production-реліз- коли додаток потрапить в store. Якщо ми публікуємо в Production, то повинні обов'язково пройти рев'ю. Люди з Google вручну перевіряють, як працює додаток, і відсилають фідбек. Якщо все добре, то додаток публікується. Якщо немає, то вам приходить лист з правками, що в вашому додатку не працює і що виправити.

Здавалося б, на цьому все, але ми говоримо про голос, тому тут є ще один важливий момент - це аналітика.

3.5. Аналітика

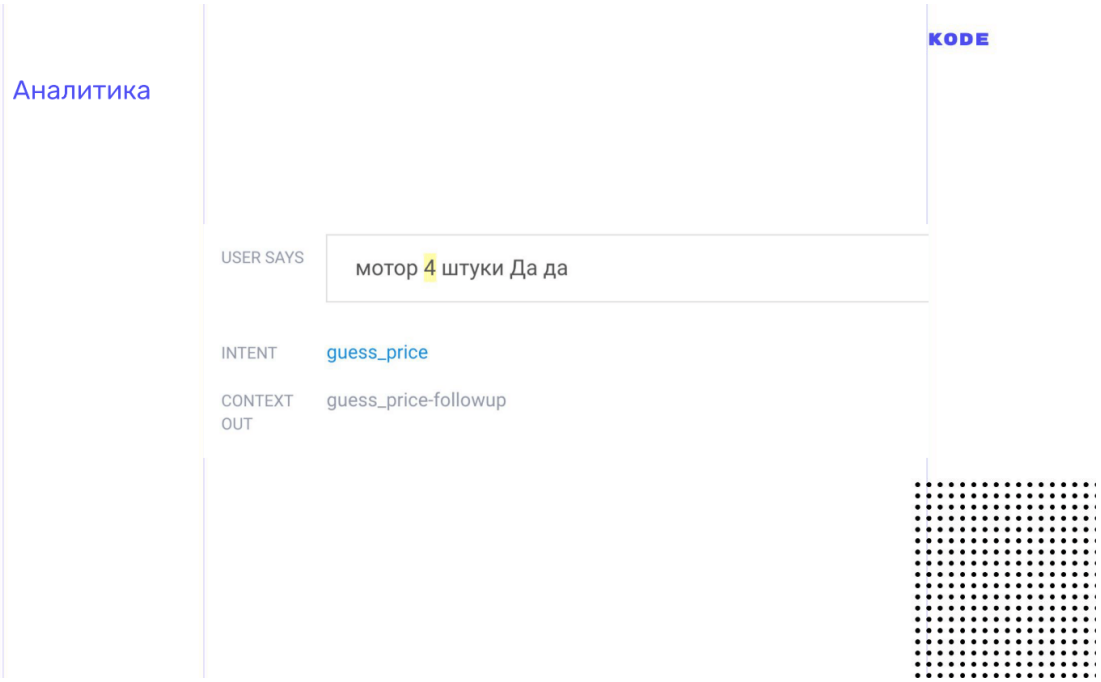
Особливо важлива для голосу. Якщо для мобільних додатків аналітика показує косяки, баги і системні помилки, то в світі голосу аналітика розкриває нам упущені можливості - то, як люди хотіли використовувати наш додаток, але не змогли.

Це обов'язковий етап. Саме тому, в Dialogflow є стандартні інструменти для аналізу додатків з наступними режимами:

Історія- знеособлена історія розмов з вашим додатком.

навчання- цікавий режим, який показує всі фрази, які додаток розпізнало, але не було в змозі обробити.

Вони показуються списком і виглядають, як на картинці. Користувач відгадував ціну якогось мотора і сказав: «4 штуки». Про «косарі» я згадав на тестуванні, а про «штуки» забув - тому доведеться лагодити.



Аналітика допомагає знайти недоробки, тому обов'язково дивіться в логи і перевіряйте, що відбувається з вашим додатком і що з ним роблять користувачі.

На цьому все про голосові програми. Сподіваюся, що у вас з'явилося хоча б мінімальне розуміння як вони розробляються. Доповідь була загальним, але всі додаткові матеріали про розробку, проектування і бізнес-частина зібрані по посиланнях.

3.6. Висновки до розділу

Проведено аналіз та описано основні інструменти розробки фреймворку тестування умінь голосового асистента. Визначені загальні принципи на яких ґрунтуватиметься розробка фреймворку.

Сьогодні особливо затребувані голосові боти в сегменті клієнтської підтримки: світові витрати на контакт-центри, за даними Everest Group, оцінюються в $ 330 млрд на рік. Саме економією на контакт-центрі пояснив запуск свого голосового помічника «Тінькофф».

Своїх помічників запускають туроператори, ритейлери, FMCG-бренди. Спеціалізовані асистенти консультують клієнтів, збирають замовлення і виконують HR-функції: віртуальний помічник «Мегафона» «Олена», голосовий навик «Папа Джонс» для замовлення піци через «Алісу» або HR-помічник в російському KFC, який звільняє рекрутерів від обробки «порожніх »заявок.

ВИСНОВКИ

Дипломний проект містить ґрунтовну інформацію щодо роботи голосових асистентів, їх майбутнього.

Обробка природної мови зараз не використовуються хіба що в зовсім консервативних галузях. У більшості технологічних рішень розпізнавання і обробка «людських» мов давно впроваджена: саме тому звичайний IVR з жорстко заданими опціями відповідей поступово відходить у минуле, чатботи починають все більш адекватно спілкуватися без участі живого оператора.

Голосові помічники були створені для того, щоб люди не витрачали зайвий час на прості щоденні завдання. Функціонал голосових помічників досить великий. Вони можуть:

Спілкуватися з користувачем.

Шукати інформацію в інтернеті і коротко відповідати на запити, що надходять від користувача.

Викликати таксі.

Дзвонити, писати повідомлення.

Включати музику.

Складати маршрут.

Заводити будильник.

Шукати автозаправки поблизу.

І багато багато іншого.

Голосові помічники завжди враховують місце розташування користувача, час доби і день тижня. Крім того вони відштовхуються від історії ваших запитів, попередніх покупок в інтернет-магазинах і т.д.

Голосові помічники істотно полегшують водіння. Іноді відволікатися на смартфон все-таки необхідно, але всі розуміють, наскільки це небезпечно. Крім простих завдань на зразок складання маршруту і підбору найбільш швидкого варіанту шляху без пробок, з голосовим помічником можна просто поговорити.

Різноманітні дослідження говорять про те, що на даний момент найперспективнішим голосовим помічником є «Аліса» від компанії «Яндекс». Програмісти активно додають нові функції і допрацьовують старі. З кожним днем цей голосовий асистент стає все розумнішими. А найбілш потужним на даний момент помічник *Alexa* від *Amazone.*

Проведено аналіз та розглянути основні типи та методи тестування, задокументовані основні відомості про них. Виявлено методи, які можна застосувати для тестування умінь голосового помічника *Alexa*. Проведено дослідження, щодо нових методів для тестування умінь.

СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ ВИКОРИСТАНИХ ДЖЕРЕЛ

ДОДАТОК А

Текст класів фреймворку для тестування умінь