Архитектура API *Ассистента на русском*

Приложение *"Ассистент на русском"* - это стандартное приложение под ОС Android, взаимодействующее с пользователем как с помощью речи, так и посредством графического интерфейса (GUI). Ассистент содержит набор собственных сервисов, которые так же реализованы с помощью данного API - мы называем их билтинами (built-in). APi предоставляет реализовать свои собственные (стороние) сервисы, которые можно называть плагинами (plugins).

Что такое сервис

Каждый сервис - это набор из агентов, модулей и некоторой бизнес-логики. Агенты представляют собой надстройку над Android-сервисами, а взаимодействие между ассистентом и агентом происходит за счёт межпроцессной коммуникации (IPC).

Функции ассистента

Само приложение *"Ассистент на русском"* состоит из GUI и ядра, в функции которого входит диспетчеризация запросов от пользователя к нужному агенту и получение от агента результата (ов) для отображения пользователю и переключение диалогового контекста. Компонент, выполняющий эти операции (не зависящие от GUI) называется `Dispatcher`.

> Таким образом, ассистент - это полностью модульная структура, разделяющая уровень представления (GUI), логику управления диалогом (Dispatcher) и сервисы (Агенты).

Распознавание речи и NLP

Ассистент взаимодействует с системой распознавания речи (ASR), позволяющей получить из пользовательской речи строку, которая затем обрабатывается специальным компонентом ядра - *Матчером*. Матчер реализует функцию NLP (процессинг натуральной речи), используя данные модулей, описывающих паттерны пользовательских речевых запросов. Результатом работы матчера является решение о том, какую команду какого агента из текущего скоупа и с какими параметрами (дерево разбора текста) необходимо выполнить.

Скоупами называются наборы модулей, доступных в данный момент времени пользователю. Такие наборы формируют диалоговый

контекст.

> *Ассистент на русском* использует сторонние решения ASR и полностью независим от конкретной реализации данной технологии.

Гибридный NLP

Матчер в "Ассистенте на русском" реализует гибридное NLP, при котором для матчинга текста по паттернам используются данные из разных источников. Другими словами, матчер перед процессингом текста может получить данные из локальных БД или отправить строку на удалённые сервера, где хранятся большие объёмы данных. Эти сервера дополняют информацию о тексте (производят *аннотирование*) и возвращают эту информацию для процессинга локальному матчеру. Таким образом можно хранить некоторые данные на серверах (например, списки городов мира), а некоторые локально на устройстве (список контактов или названия приложений).

Таким образом, вы можете сформировать паттерны различными способами, что позволяет реализовать сколь угодно сложный вид речевой коммуникации с пользователем:

- Статически описать их в модуле
- Использовать контент-провайдеры для динамической генерации сущностей для паттерна (в том числе и для получения данных с удаленных устройств или из локальных БД)
- Хранить большие списки сущностей на серверах, где работает специальный аннотатор, а в паттерне указать источник (атрибут `uri`)

> О том, как использовать аннотирование текста на удалённых серверах, читайте в разделе контент-провайдеры.

Это позволяет масштабировать разрабатываемую систему так, что становится возможным, с одной стороны, не зависеть от способов хранения необходимых данных, а с другой - не описывать все возможные сущности, а ограничиться только теми, которые действительно необходимы для реализации логики приложения.

Взаимодействие с агентом

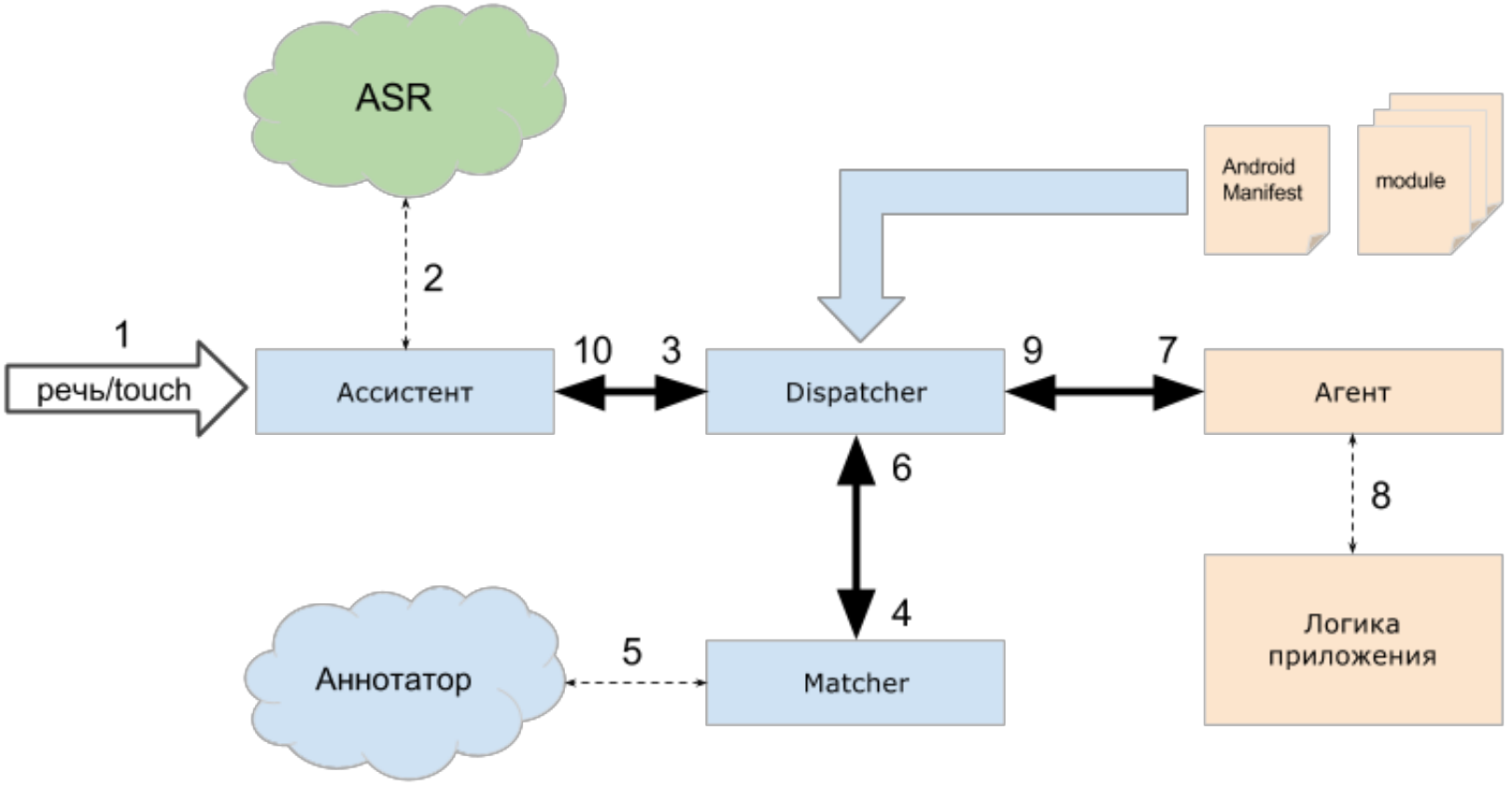
Как было сказано выше, ядро ассистента взаимодействует с агентом с помощью IPC. Агент, по сути, выполняет роль программного интерфейса между ассистентом и бизнес-логикой приложения. Каждый агент описывается в манифесте приложения `AndroidManifest.xml`, и ассистент загружает его основной модуль каждый раз при "холодном" старте или динамически при установке приложения.

Далее при каждом запросе пользователя к агенту ассистент вызывает методы абстрактного класса `AssistantAgent`, которые должны быть реализованы конкретным агентом. Ассистент ведёт историю запросов пользователя и привязывает ответ(ы) от агента на запрос к конкретному элементу в истории.

Каждый ответ от агента - это некоторый контент, отображаемый в GUI ассистента и, опционально, инструкция по управлению диалогом (подробнее в разделе Scopes).

> Подробнее о видах контента, генерируемого агентами, читайте в разделе баблы.

Схема взаимодействия пользователя с агентами



Синим в этой схеме обозначаются компоненты, реализованные в Ассистенте. Розовым - компоненты, которые необходимо реализовать разработчику в своем приложении. Зеленым - сторонние компоненты

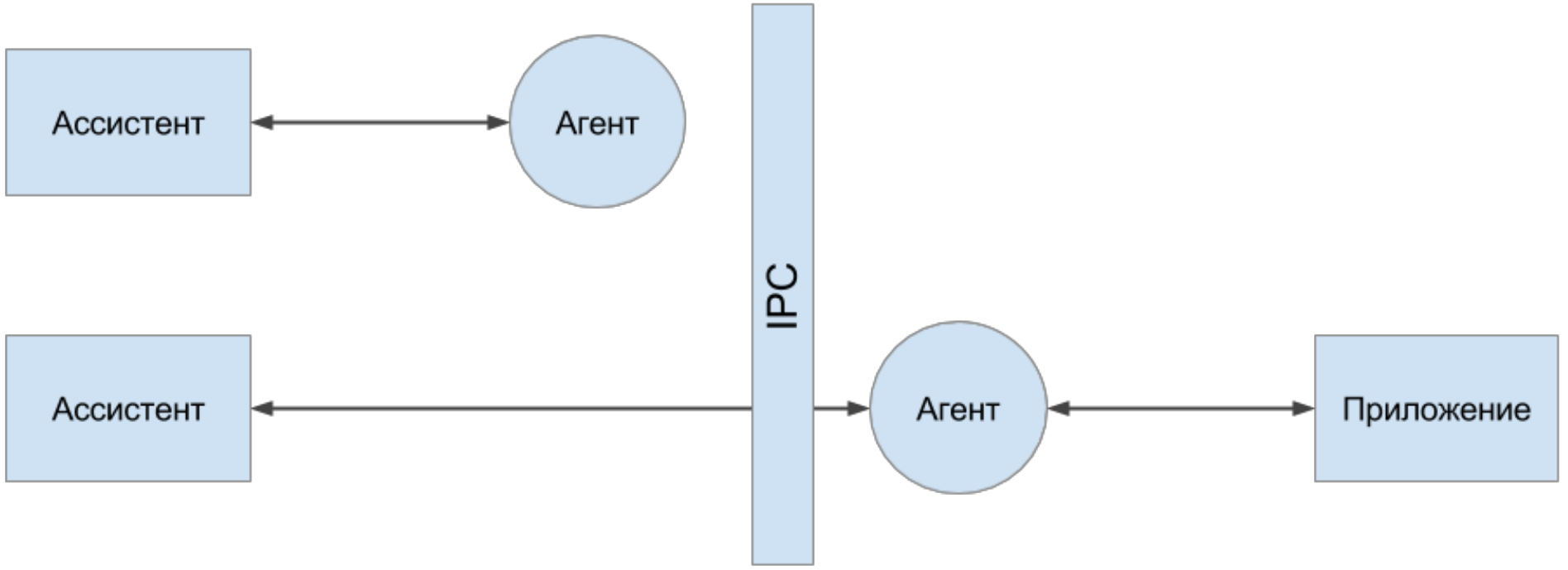
В общем случае, после успешной загрузки модулей вашего приложения ассистентом, схема взаимодействия пользователя с агентом выглядит так:

1. Пользователь взаимодействует с приложением "Ассистент на русском" либо с помощью речи (активирует микрофон и говорит фразу), либо с помощью touch-интерфейса (нажимает на кнопки, выбирает элементы из списка и т.п.).
2. Ассистент обрабатывает ввод пользователя и, при необходимости, взаимодействует с системой распознавания речи (ASR) для получения строки текста из речи.

3. Затем Ассистент формирует запрос (Request) к диспетчеру (Dispatcher), который управляет всем процессом взаимодействия с матчером и агентами.
4. Диспетчер в каждый момент времени хранит информацию о состоянии контекста. Получив от Ассистента запрос, он вызывает Матчер на том скоупе, который сейчас активен, передавая Матчеру строку текста. *Если запрос пользователя представляет собой взаимодействие с GUI (PendingRequest), а не речь, то процесс взаимодействия с Матчером пропускается и Диспетчер напрямую диспетчеризует уже готовый запрос к Агенту (п. 7).*
5. При наличии сетевого подключения Матчер сперва отправляет строку на удаленные сервера для добавления в текст необходимой информации, а затем производит матчинг по паттернам из указанного скоупа модулей.
6. Результатом работы Матчера является информация о том, какую команду какого агента необходимо вызывать. Также Матчер возвращает Диспетчеру токен - семантическое дерево разбора фразы по паттерну команды.
7. Диспетчер формирует запрос к конкретному Агенту приложения, при необходимочти разворачивая автоматические скоупы. Вызов методов Агента происходит по IPC.
8. Агент обрабатывает вызов одного из методов `AssistantAgent`, взаимодействуя с логикой приложения. Реализация этого взаимодействия ничем не ограничена и зависит только от разработчика приложения.
9. На каждый запрос от Диспетчера Агент обязан вернуть хотя бы один ответ (Response), который содержит контент ответа и, по необходимости, информацию об изменении скоупа. *Если выполнение логики приложения занимает продолжительное время (сетевое взаимодействие и т.п. операции), то Агент может вернуть Диспетчеру ответ, уйти в бэкграунд (`goBackground`), а затем обновить уже отправленные ранее ответ.*
10. После получения от Агента ответа(ов), Диспетчер обновляет состояние скоупа и передает Ассистенту необходимые данные для отображения в пользовательском интерфейсе и озвучки результата через TTS.

Виды взаимодействия агента с ассистентом

С точки зрения способа реализации взаимодействия агента с ядром ассистента, существует несколько подходов.



Built-in агент

Этот вид реализации агентов используется для сервисов самого "Ассистента на русском" или любого другого приложения, реализованного на его ядре (не описывается в данном документе).

Сторонний агент

Этот вид используется для реализации взаимодействия вашего (стороннего) приложения с ассистентом. Агент реализуется на стороне вашего приложения и предоставляет интерфейс к нему. Такой агент может отображать необходимый GUI внутри ассистента посредством

баблов.

> Ваше приложение при этом не обязано иметь *собственный* touch GUI, иконку для лаунчера и т.д. В таком случае, оно будет выглядеть как plug-in к ассистенту.