Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение   
высшего образования

**Иркутский национальный исследовательский технический университет**

|  |
| --- |
| Институт информационных технологий и анализа данных |
| наименование института |

«Системы искусственного интеллекта»

|  |
| --- |
| **Отчёт** |
| по семестровой работе по теме «Разработка интеллектуального голосового помощника» |
| наименование темы |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Выполнил студент |  | АСУб-20-2 |  |  |  | А. В. Арбакова |
|  |  | шифр |  | подпись |  | И.О. Фамилия |
| Проверил |  |  |  |  |  | А.Б. Столбов |
|  |  |  |  | подпись |  | И.О. Фамилия |
| Работа защищена с оценкой | | | |  | | |

Иркутск 2023 г.

**Содержание**

[Введение 3](#_Toc156293584)

[1 Руководство по Google Calendar 4](#_Toc156293585)

[2 Нюанс с Google API 10](#_Toc156293586)

[3 Реализация 11](#_Toc156293587)

[3.1. Описание онтологий «События» и «Время» 11](#_Toc156293588)

[3.2. Создание базы знаний на языке CLIPS 13](#_Toc156293589)

[3.3. Спецификация голосового помощника 15](#_Toc156293590)

[3.4. Архитектура приложения 20](#_Toc156293591)

[4 Описание сервиса и компонентов 23](#_Toc156293592)

[5 Тестирование 29](#_Toc156293593)

[5.1. Тест 1 29](#_Toc156293594)

[5.2. Тест 2 30](#_Toc156293595)

[Заключение 33](#_Toc156293596)

# Введение

Виртуальные голосовые помощники — это высокотехнологичные системы, которые уже давно не просто включают будильник на смартфоне или запускают нужный музыкальный трек. Они цитируют классику, рассказывают детям сказки и даже умеют обижаться. В рамках данной семестровой работы предлагается команде разработать виртуального помощника, который будет решать задачу проверки возможности планирования событий по календарю. Команда состоит из таких студентов как: Ермоленко Антон из группы АСУб-20-1, Манданов Вадим из группы АСУб-20-1 и Арбакова Анастасия из группы АСУб-20-2.

Необходимо на основе сервиса Google Calendar составить онтологию событий: место, дата, время и компаньоны. Онтология должна строиться на основе онтологии «События» (events) и «Дата и Время» (datetime).

Далее необходимо написать алгоритм выбора решения, используя основанный на правилах язык программирования CLIPS. В последующем правила будут использоваться для определения ответа пользователю на основании указанных им данных.

Следующим шагом нужно перенести онтологию событий в спецификацию Dialog Flow. В качестве сущностей (entities) будут выступать классы онтологии. Структура намерений (intents) должна логично выстраивать диалог с пользователем, при этом позволяя начать диалог с любой точки.

Последним этапом работы является связывание. Необходимо написать PHP скрипт, который будет обрабатывать приходящие из Dialog Flow данные, преобразовывать их в сущности CLIPS, запускать правила и обрабатывать возвращаемый из CLIPS результат. Обработанный результат передаётся обратно в Dialog Flow либо в виде готового ответа пользователю, либо в виде специфичной информации, создающей контексты или запускающей другие намерения.

## Руководство по Google Calendar

В Google Calendar есть следующие основные компоненты:

* Пользователь
* Календарь
* Дата и время
* Событие

Календари разделяются по пользователям. Они становятся доступны при предоставлении пользователем доступа.

В календаре есть сущность даты, которая в свою очередь может иметь события, которые связаны с временем.

Для получения доступа к Google Calendar Api нужно:

1. В консоли Google Cloud (https://console.cloud.google.com/) выберите  Select Project > New Project;

Изображение выглядит как текст, снимок экрана, программное обеспечение, веб-страница

Автоматически созданное описание

Рисунок 1.1 – Кнопка «Select project»

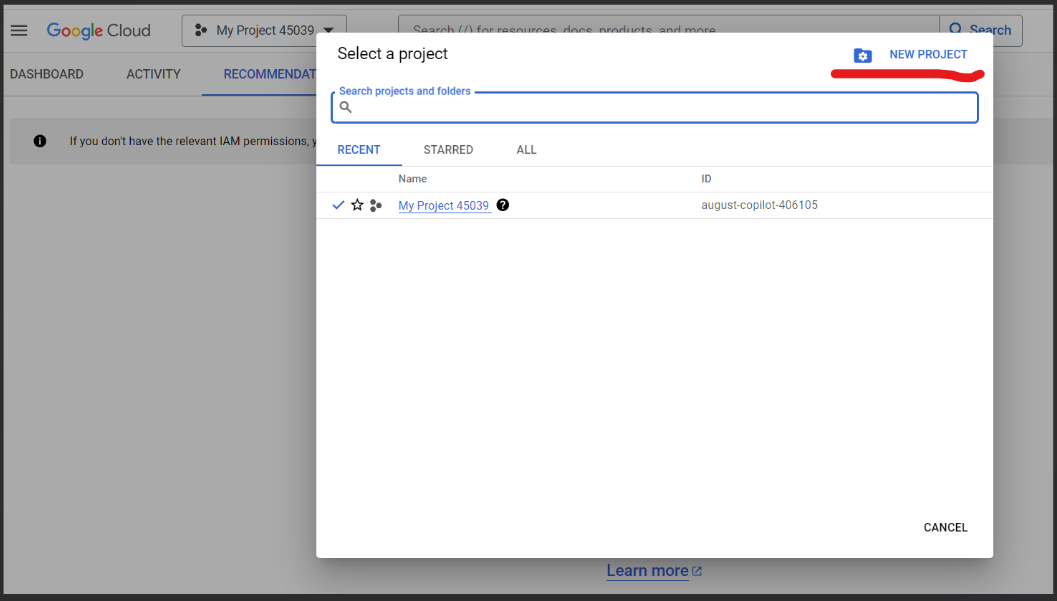


Рисунок 1.2 – Кнопка «New project»

1. Введите данные для создания проекта (имя проекта) и нажмите кнопку «Create»;

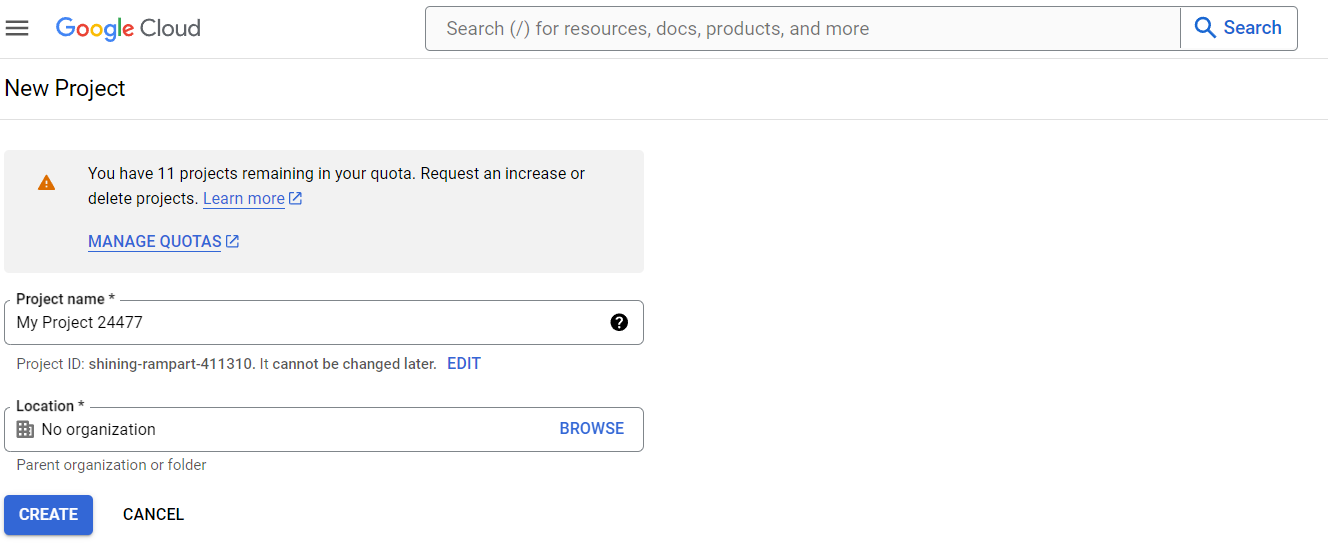


Рисунок 1.3 – Кнопка «Create»

1. В консоли Google Cloud выберите menu > More products > Google Workspace > Product Library;

Изображение выглядит как текст, снимок экрана, число, диаграмма

Автоматически созданное описание

Рисунок 1.4 – Кнопка «Library»

Изображение выглядит как текст, снимок экрана, программное обеспечение, число

Автоматически созданное описание

Рисунок 1.5 – Выбор Google Calendar API

В списке API выберите Google Calendar API и нажмите кнопку «Enable»;Изображение выглядит как текст, снимок экрана, программное обеспечение, веб-страница

Автоматически созданное описание

Рисунок 1.6 – Кнопка Enable (в данном примере ее нет, т.к. API включен, но кнопка «Enable» будет вместо кнопки «manage»)

1. В консоли Google Cloud выберите menu > APIs&Services > Credentials;

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 1.7 – Раздел «Credentials»

1. Нажмите на кнопку «create credentials»;

Изображение выглядит как текст, программное обеспечение, Шрифт, веб-страница

Автоматически созданное описание

Рисунок 1.8 – Кнопка «create credentials»

1. Выберите OAuth ClientID, заполняете все данные (тип веб-приложения, название, redirect uri (в нашем случае для Postman: https://oauth.pstmn.io/v1/callback)) и создаете данные для доступа к API, после чего в пункте Credentials появятся данные для авторизации по технологии Auth 2.0 (ClientID, ClientSecret, redirect\_uri);

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 1.9 – Раздел «create credentials» с полями для ввода данных для создания Auth 2.0 credentials

Изображение выглядит как текст, программное обеспечение, Значок на компьютере, веб-страница

Автоматически созданное описание

Рисунок 1.10 – Раздел «Credentials» с данными для Auth 2.0 после создания Auth 2.0 credentials

1. После всех действий в google console, для доступа к google API необходим токен доступа, который можно получить после авторизации по Auth 2.0. Для этого необходимо приложение имеющее свой домен с протоколом https (данный домен будет использоваться как redirect\_uri). В нашем случае этим приложением является Postman.

## Нюанс с Google API

В ходе разработки ВА для google calendar была выявлена проблема: для доступа к API необходим токен, который живет 1 час. Для получения токена нужно авторизироваться по Auth 2.0. В свою очередь, данная технология требует наличие приложения способного отправлять запрос для авторизации по Auth 2.0, а также приложение должно иметь свой домен, на который будет происходить перенаправление после авторизации в google. В нашем случае в роли приложения выступает Postman (во вкладке авторизации вводятся данные ClientID, SecretID, Redirect\_URI по умолчанию равен https://oauth.pstmn.io/v1/callback). Также домен приложения должен иметь протокол https для связи с сервисами google.

В итоге для полноценной работы (т.е. автоматического получения и обновления токена без ведома пользователя) ВА необходимо наличие приложения на сервере способного авторизироваться по auth 2.0 в Google API.

## Реализация

## Описание онтологий «События» и «Время»

На основе онтологий «Events» и «datetime» создадим новую. Эти онтологии имеют следующие важные для нас сущности:

1. Group of Person – выделяет сущности возможных групп людей;
2. Day of week –выделяет сущности дней недели;
3. Event – выделяет сущности событий;
4. Person – выделяет сущности людей;
5. SpatialThing – выделение сущности места;
6. Temporal – сущности реализующие хранение времени.

Но для реализации функций нашего голосового помощника необходимо добавить «Group of Person», что будет выделять сущности возможных групп людей; Также необходимо дополнить сущности

«Event», а именно, создать в ней «Горы», «Кино», «Матч», «Сон» и «Работа».

SpatialThing, а именно, создать в ней «ConstantEvents», «DialogFlowEvents» и «EventsBase».

Созданные сущности представлены на рисунке 3.1.

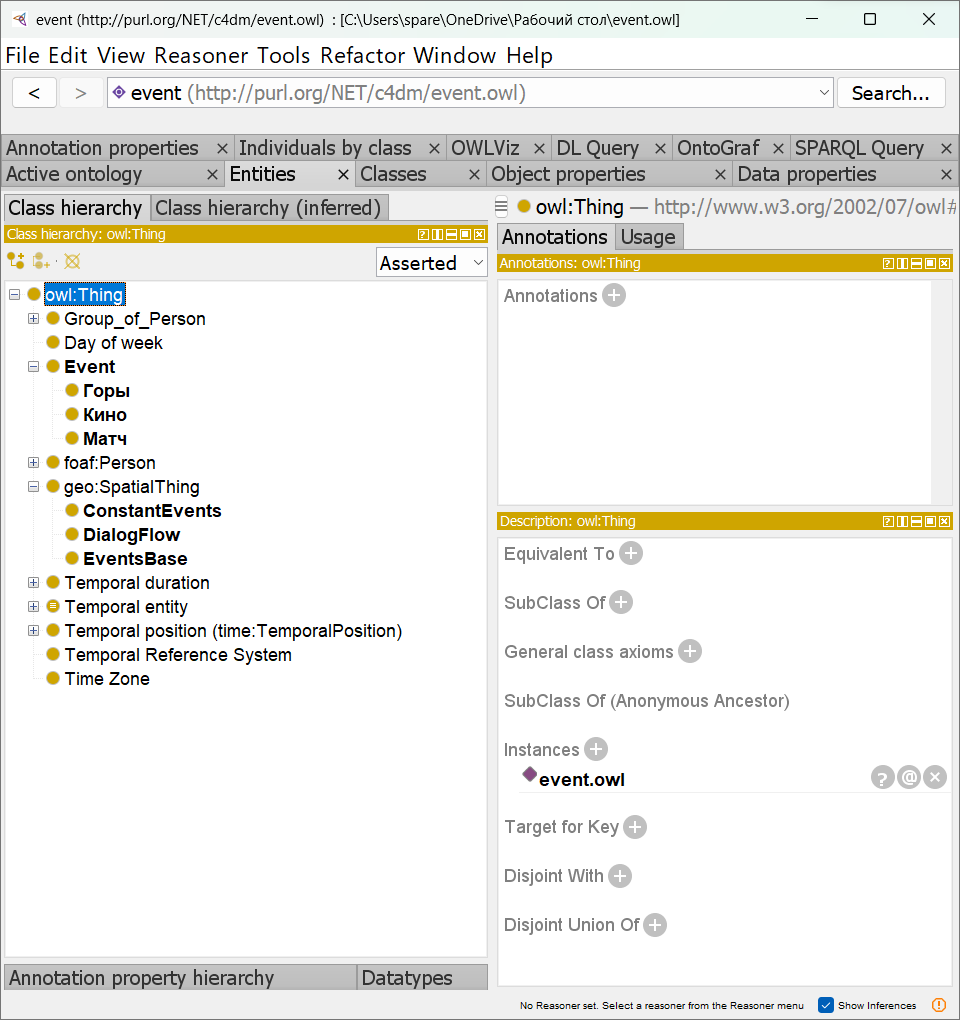


Рисунок 3.1 – Созданные сущности

Свойства объектов в онтологии представлены на рисунке 3.2.

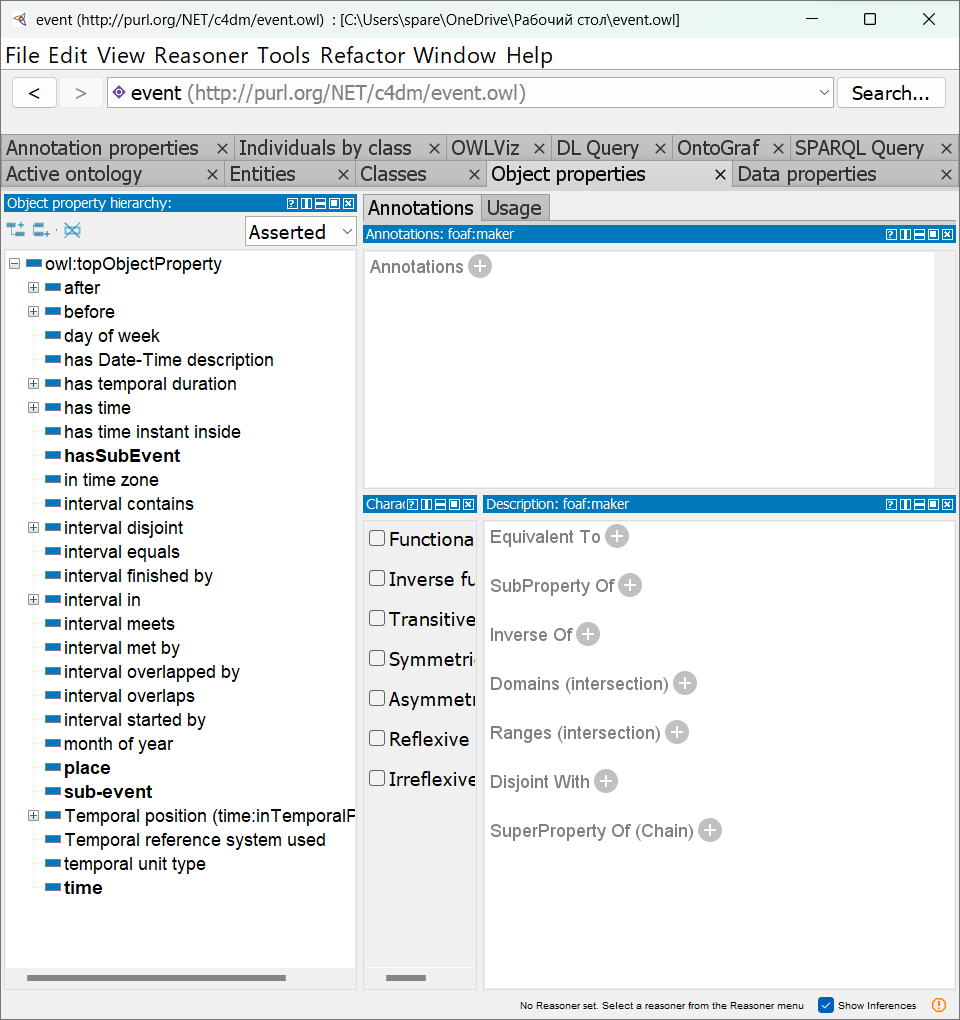


Рисунок 3.2 – Свойства объектов

Свойства данных показаны на рисунке 3.3.

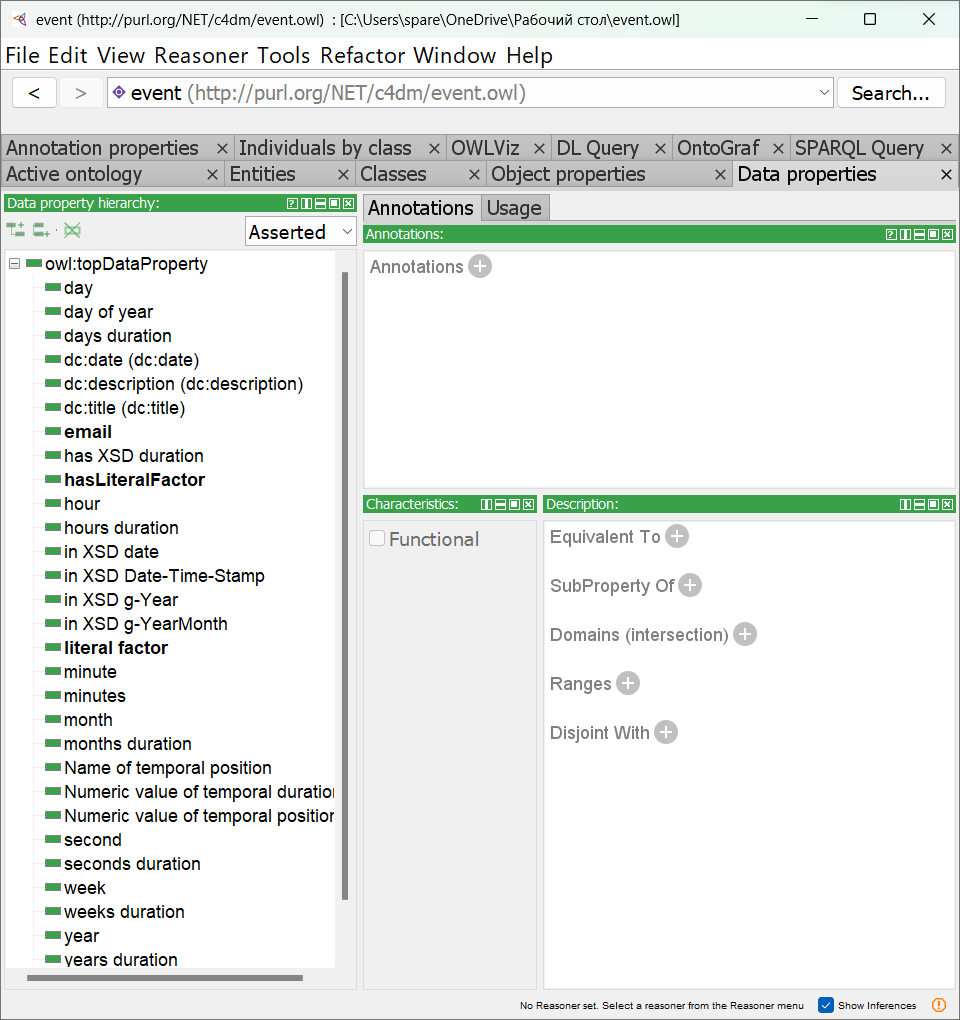


Рисунок 3.3 –Свойство данных

## Создание базы знаний на языке CLIPS

Общая структура CLIPS будет состоять из двух частей:

* описание шаблонов (классов), отражающих структуру онтологии;
* описание правил, определяющих логику диагностики.

Файл KnowledgeBase.clp содержит правила и описание классов CLIPS, соотносящихся с онтологией. Пример шаблонов:

(deftemplate weekday

(slot id)

(slot name)

)

(deffacts weekdays

(weekday (id 1) (name "понедельник"))

(weekday (id 2) (name "вторник"))

(weekday (id 3) (name "среда"))

(weekday (id 4) (name "четверг"))

(weekday (id 5) (name "пятница"))

(weekday (id 6) (name "суббота"))

(weekday (id 7) (name "воскресенье"))

)

(deftemplate personGroup

(slot id)

(slot group\_name)

(slot person\_name)

)

(deftemplate person

(slot id)

(slot name)

(slot group\_name)

)

(deftemplate freeTime

(slot id\_person)

(slot minDate)

(slot maxDate)

(slot minHour)

(slot maxHour)

(slot minMinute)

(slot maxMinute)

(slot minSecond)

(slot maxSecond)

(slot id\_weekday)

)

Пример правил:

(defrule person-count-three

(initial-fact)

(eventRequest (group\_name ?eventRequest\_group\_name))

(test (eq ?eventRequest\_group\_name "коллеги"))

=>

(assert (three-person))

)

(deffunction timeString (?time)

(if (< ?time 10)

then (str-cat "0" ?time)

else

?time

)

)

(defrule create-used-freeTime-1

(initial-fact)

(event (name ?event\_name) (hour ?event\_hour) (minute ?event\_minute) (second ?event\_second))

(eventRequest (group\_name ?eventRequest\_group\_name) (date ?eventRequest\_date) (name ?eventRequest\_name)

(minHour ?minHourRequest) (minMinute ?minMinuteRequest) (minSecond ?minSecondRequest)

(maxHour ?maxHourRequest) (maxMinute ?maxMinuteRequest) (maxSecond ?maxSecondRequest))

(personGroup (group\_name ?personGroup\_group\_name) (person\_name ?personGroup\_person\_name))

(person (id ?id) (name ?person\_name))

(freeTime (id\_person ?person\_id) (minDate ?minDate) (maxDate ?maxDate)

(minHour ?minHour) (minMinute ?minMinute) (minSecond ?minSecond)

(maxHour ?maxHour) (maxMinute ?maxMinute) (maxSecond ?maxSecond) (id\_weekday ?id\_weekday))

(test (and (eq ?eventRequest\_group\_name ?personGroup\_group\_name)

(eq ?personGroup\_person\_name ?person\_name)

(= ?person\_id ?id)

(eq ?minDate ?eventRequest\_date)

(eq ?event\_name ?eventRequest\_name)

(<= (+ (\* (+ (\* ?event\_hour 60) ?event\_minute) 60) ?event\_second)

(- (+ (\* (+ (\* ?maxHour 60) ?maxMinute) 60) ?maxSecond) (+ (\* (+ (\* ?minHour 60) ?minMinute) 60) ?minSecond))

)

(<= (+ (\* (+ (\* ?minHourRequest 60) ?minMinuteRequest) 60) ?minSecondRequest)

(+ (\* (+ (\* ?minHour 60) ?minMinute) 60) ?minSecond)

)

(>= (+ (\* (+ (\* ?maxHourRequest 60) ?maxMinuteRequest) 60) ?maxSecondRequest)

(+ (\* (+ (\* ?maxHour 60) ?maxMinute) 60) ?maxSecond)

)

)

)

=>

(assert (useFreeTime (id\_person ?person\_id) (minDate ?minDate) (maxDate ?maxDate)

(minHour ?minHour) (minMinute ?minMinute) (minSecond ?minSecond)

(maxHour ?maxHour) (maxMinute ?maxMinute) (maxSecond ?maxSecond) (id\_weekday ?id\_weekday)))

)

## Спецификация голосового помощника

Первым шагом перенесём классы из онтологии в сущности Dialog Flow (рисунок 3.4).

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

Рисунок 3.4 – Сущности Dialog Flow

В качестве значения сущности используем английское наименование из онтологии, в качестве синонимов можно использовать любые соотносящиеся по смыслу значения (рисунок 3.5).

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 3.5 – Значения и синонимы сущности Dialog Flow

Вторым шагом необходимо создать намерения. Намерения могут вызываться пользовательским вводом, когда введённый текст соотносится с заранее заданными шаблонами, либо запускаться с помощью специальных событий, которые, например, могут приходить в качестве ответа от сервера.

В таблице 3.1 отображён список всех намерений, необходимых для реализации диалога с пользователем, последовательность их срабатывания, а также описание.

Таблица 3.1 – Список намерений

|  |  |  |  |
| --- | --- | --- | --- |
| **Название** | **Порядок срабатывания** | **Как вызывается** | **Выполняемая функция** |
| Welcome\_intent | 1 | Пользовательским вводом | Приветствие пользователя |
| Request\_for\_event | 2 | Пользовательским вводом | Запрос на сервер с параметрами для анализа занятости пользователя. |
| Ask\_event\_parameter\_context | 3 (если все параметры введены пользователем) | Событием, приходящим с сервера | Вывести сформированный на сервере ответ пользователю (запрос с вводом недостающего параметра) |
| Ask\_time\_context | Событием, приходящим с сервера | Вывести сформированный на сервере ответ пользователю (запрос с вводом недостающего параметра) |
| Ask\_day\_of\_week\_context | Событием, приходящим с сервера | Вывести сформированный на сервере ответ пользователю (запрос с вводом недостающего параметра) |
| Ask\_person\_context | Событием, приходящим с сервера | Вывести сформированный на сервере ответ пользователю (запрос с вводом недостающего параметра) |
| Trigger\_by\_Request\_for\_event (ответ от сервера) | 3 (если все параметры введены пользователем), иначе 4 | Событием, приходящим с сервера | Вывести сформированный на сервере ответ пользователю (конечный ответ на возможность проведения события) |
| End\_of\_dialog | 4 (если все параметры введены пользователем), иначе 5 | Пользовательским вводом | Конец диалога |

В таблице 3.1 видно, что работа с пользователем происходит последовательно, однако на 7 шаге возможен возврат к предыдущим этапам диалога, если неисправность не устранена. Также подробная схема вызовов интентов приведена на рисунке 3.6.

Изображение выглядит как текст, диаграмма, линия, Параллельный

Автоматически созданное описание

Рисунок 3.6 – Последовательность вызовов интентов

Шаблоны фраз в намерениях необходимо соотносить с сущностями, чтобы потом их можно было использовать как переменные контекста или как основу создания начальных состояний в CLIPS. Пример таких фраз отображён на рисунке 3.7.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 3.7 – Размеченные фразы пользовательского ввода в намерении

Далее, если намерение не подразумевает какую-то сложную логику, можно просто составить варианты ответов голосового помощника. Желательно указывать несколько вариантов, чтобы создавалось ощущение диалога с реальным пользователем. Если же намерение подразумевает логику, необходимо включить вызов вебхука в разделе Fulfillment (рисунок 3.8).

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 3.8 – Включение вебхука для намерения

Чтобы отправлять данные из Dialog Flow на сервер, необходимо настроить вебхук. В общем случае необходимо указать ссылку. В зависимости от настроек сервера можно также добавить авторизацию и прочие заголовки. Обращаю внимание, что Dialog Flow будет работать только по HTTPS. Пример конфигурации отображён на рисунке 3.9.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 3.9 – Пример конфигурации вебхука в Dialog Flow

Также был подключен Telegram в качестве канала связи (рисунок 3.10 и рисунок 3.11).

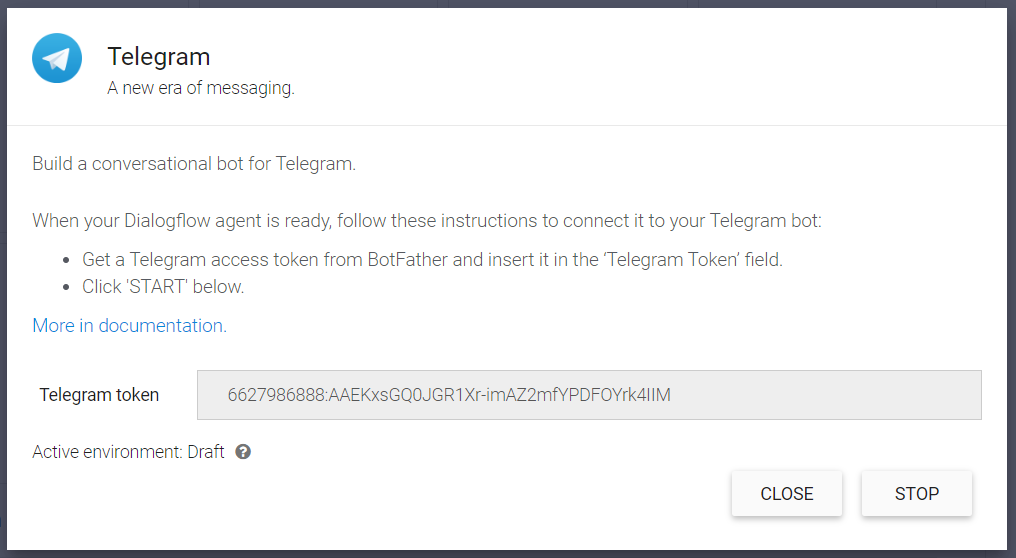


Рисунок 3.10 – Настройки для интеграции с Telegram

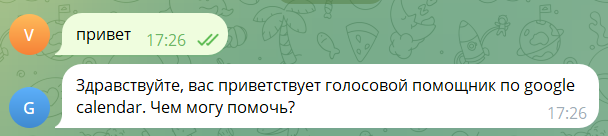


Рисунок 3.11 – Пример взаимодействия с ботом Telegram

## Архитектура приложения

Общая архитектура взаимодействия изображена на рисунке 3.12.

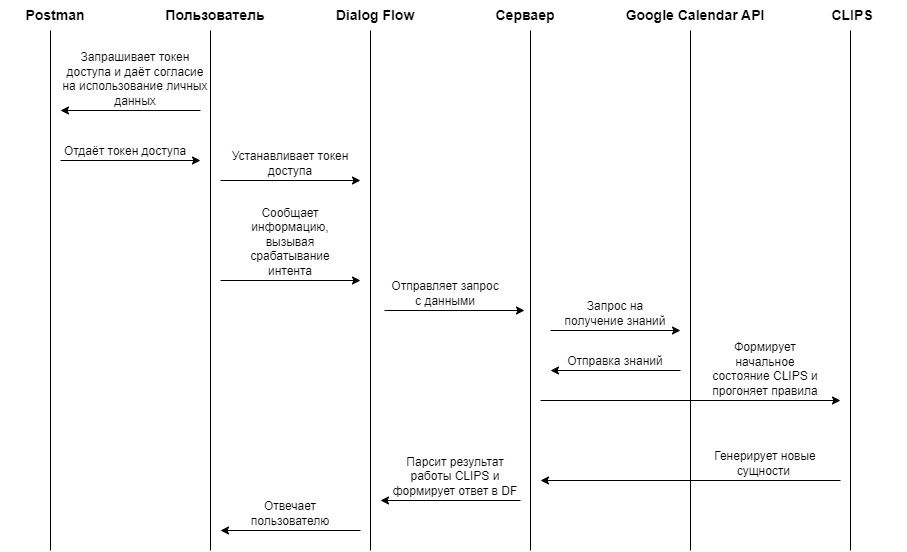


Рисунок 3.12 – Общая архитектура взаимодействия всех частей

Из этой схемы можно видеть, в чём заключается роль каждого компонента в голосовом помощнике.

* 1. **Работа с сервером**

Назначение сервера заключается в преобразовании пришедших из Dialog Flow данных в сущности CLIPS, загрузка начальных значений, запуск самого CLIPS, преобразовании ответа от CLIPS в ответ Dialog Flow. Ниже отображены примеры кода.

Преобразование данных в сущности CLIPS:

foreach ($persons as $person) {

    $clipsString .= "(person (id {$person['id']}) (name \"{$person['name']}\"))\n";

}

foreach ($personGroups as $personGroup) {

    $clipsString .= "(personGroup (id {$personGroup['id']}) (group\_name  \"{$personGroup['group\_name']}\") (person\_name  \"{$personGroup['person\_name']}\"))\n";

}

foreach ($events as $event) {

    $clipsString .= "(event (id {$event['id']}) (name  \"{$event['name']}\") (hour  {$event['hour']}) (minute  {$event['minute']}) (second  {$event['second']}))\n";

}

foreach ($freeTimes as $freeTime) {

    $clipsString .= "(freeTime (id\_person {$freeTime['id\_person']}) (minDate  \"{$freeTime['minDate']}\") (maxDate  \"{$freeTime['maxDate']}\") (minHour  {$freeTime['minHour']}) (minMinute  {$freeTime['minMinute']}) (minSecond  {$freeTime['minSecond']}) (maxHour  {$freeTime['maxHour']}) (maxMinute  {$freeTime['maxMinute']}) (maxSecond  {$freeTime['maxSecond']}) (id\_weekday  {$freeTime['id\_weekday']}))\n";

}

$clipsString .= "(eventRequest (name \"$eventClips\") (date \"$date\_of\_event\") (group\_name \"$personClips\") (minHour $minHourTime)  (minMinute $minMinuteTime)  (minSecond $minSecondTime) (maxHour $maxHourTime)  (maxMinute $maxMinuteTime)  (maxSecond $maxSecondTime))";

// print\_r($clipsString);

$deffacts="(deffacts dialogflow\_data ".$clipsString." )";

Запуск CLIPS:

 $result = RunClips($user, $deffacts);

Преобразование ответа:

$response =

[

    "fulfillmentText"=> $parse,

    "fulfilmentMessage" =>

        [

            [

                "text" => [$parse, $result, $freeTimes]

            ]

        ],

    'statusCode'=> 200,

    'outputContexts'=> [$Context]

];

echo json\_encode($response);

die();

## Описание сервиса и компонентов

* 1. **Сервис**

Работа с сервисом производится через URL – <https://www.googleapis.com/calendar/v3/>.

Точки доступа используемые нами представлены на рисунке 4.1.

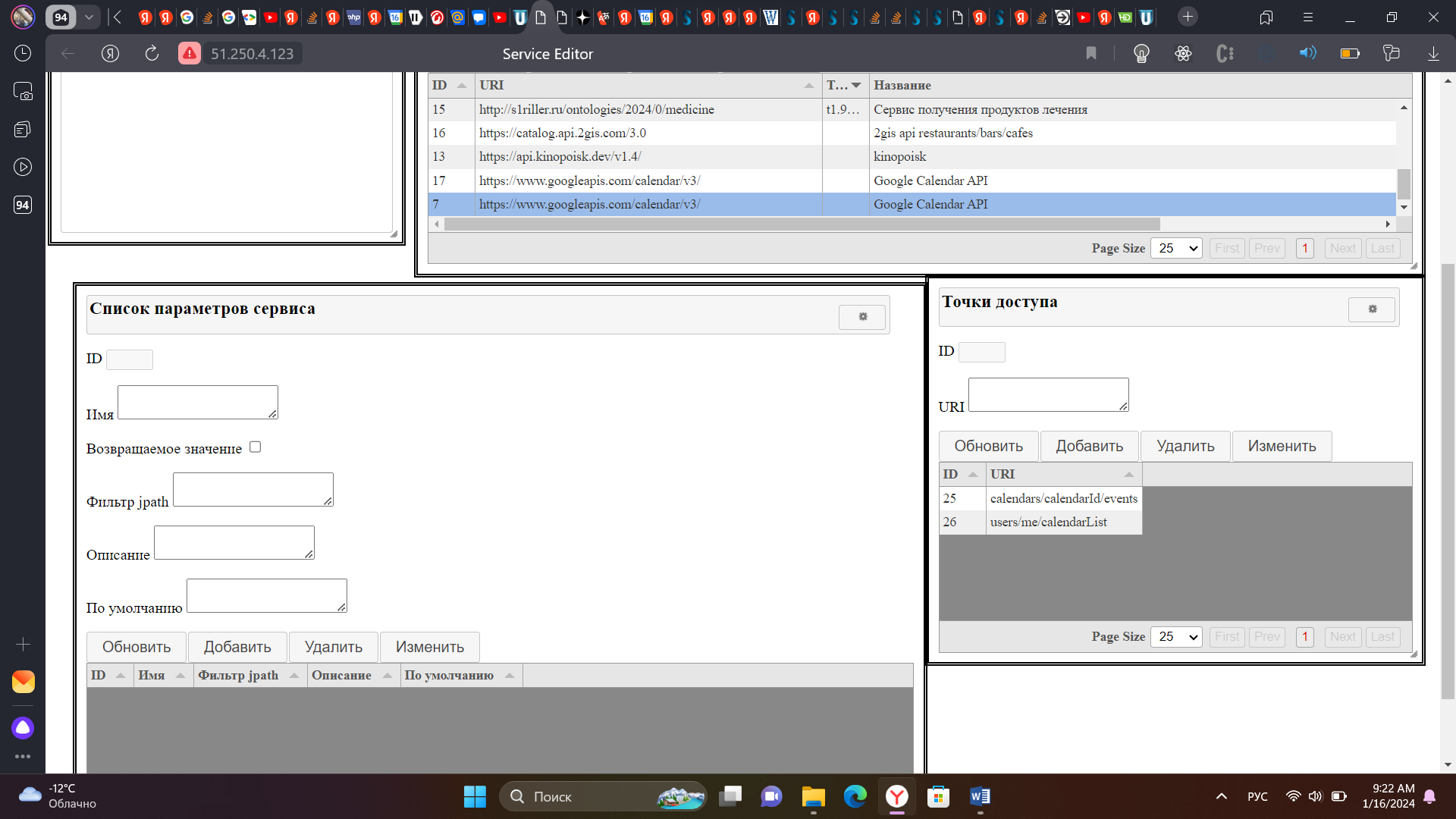


Рисунок 4.1 – Точки доступа к Google Calendar API

Параметры сервиса по точке доступа «calendars/calendarId/events» представлены на рисунке 4.2. Она предоставляет доступ к событиям пользователя.

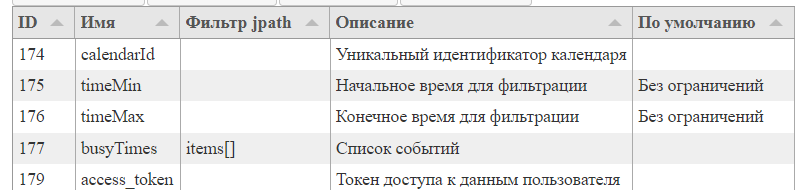


Рисунок 4.2 – Параметры для получения событий

Параметры сервиса по точке доступа «users/me/calendarList» представлены на рисунке 4.3. Она предоставляет доступ к идентификаторам доступных календарей.

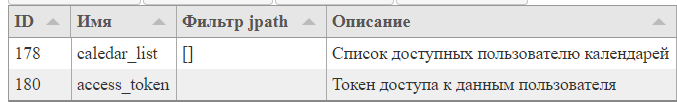


Рисунок 4.3 – Параметры для получения событий

Связь сервиса с компонентом представлена на рисунке 4.4.

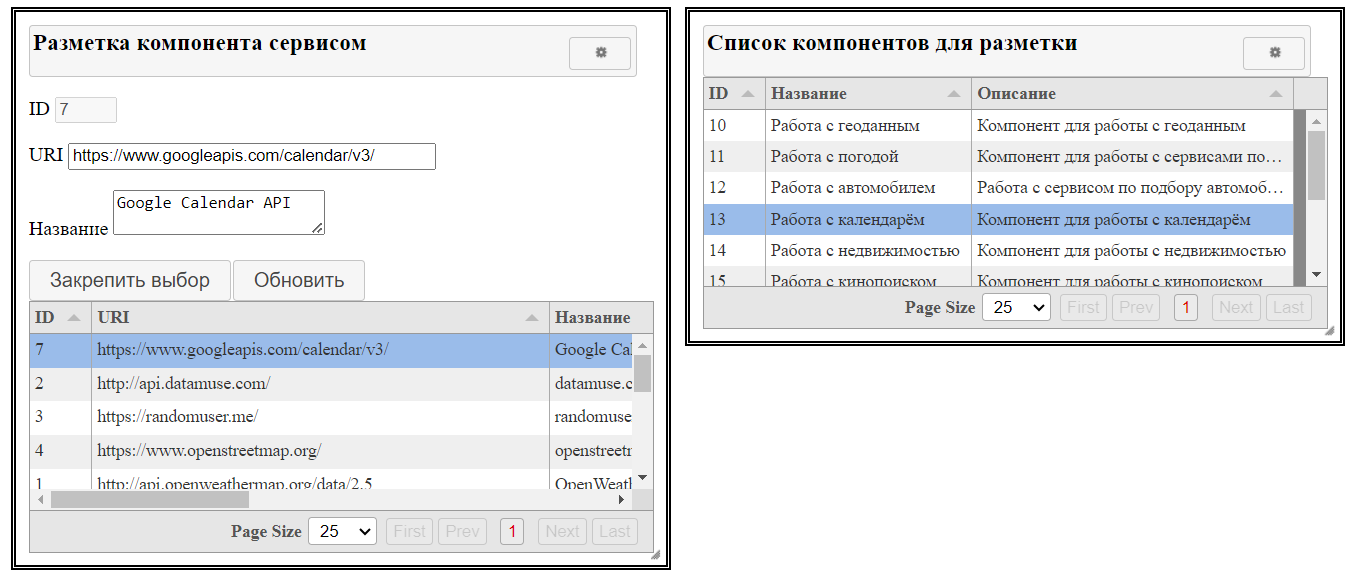


Рисунок 4.4 – Связь сервиса и компонента

* 1. **Компонент**

Функции компонента, используемые для связи с сервисом представлены на рисунках 4.5 и 4.6.

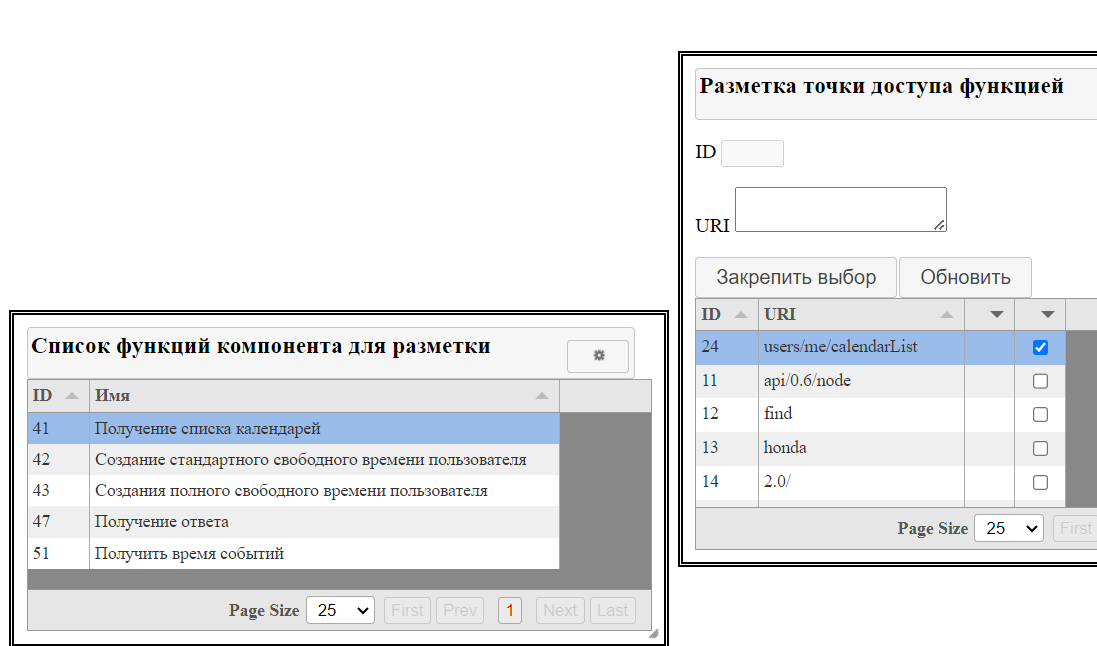


Рисунок 4.5 – Связь функции и точки доступа

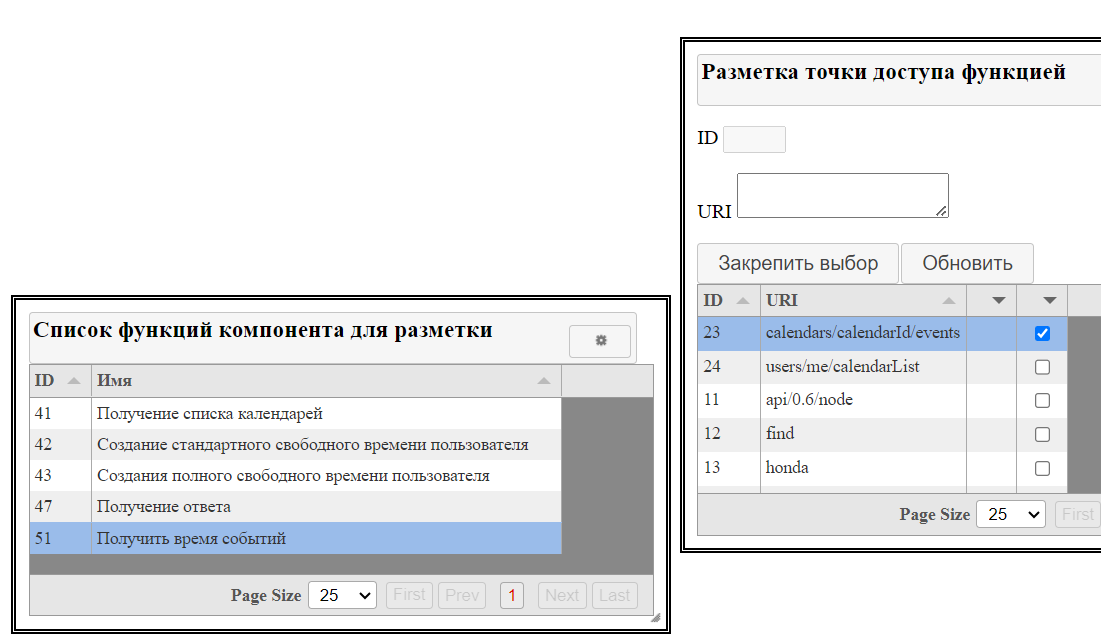


Рисунок 4.6 – Связь функции и точки доступа

Параметры этих функций представлены на рисунках 4.7 и 4.8.

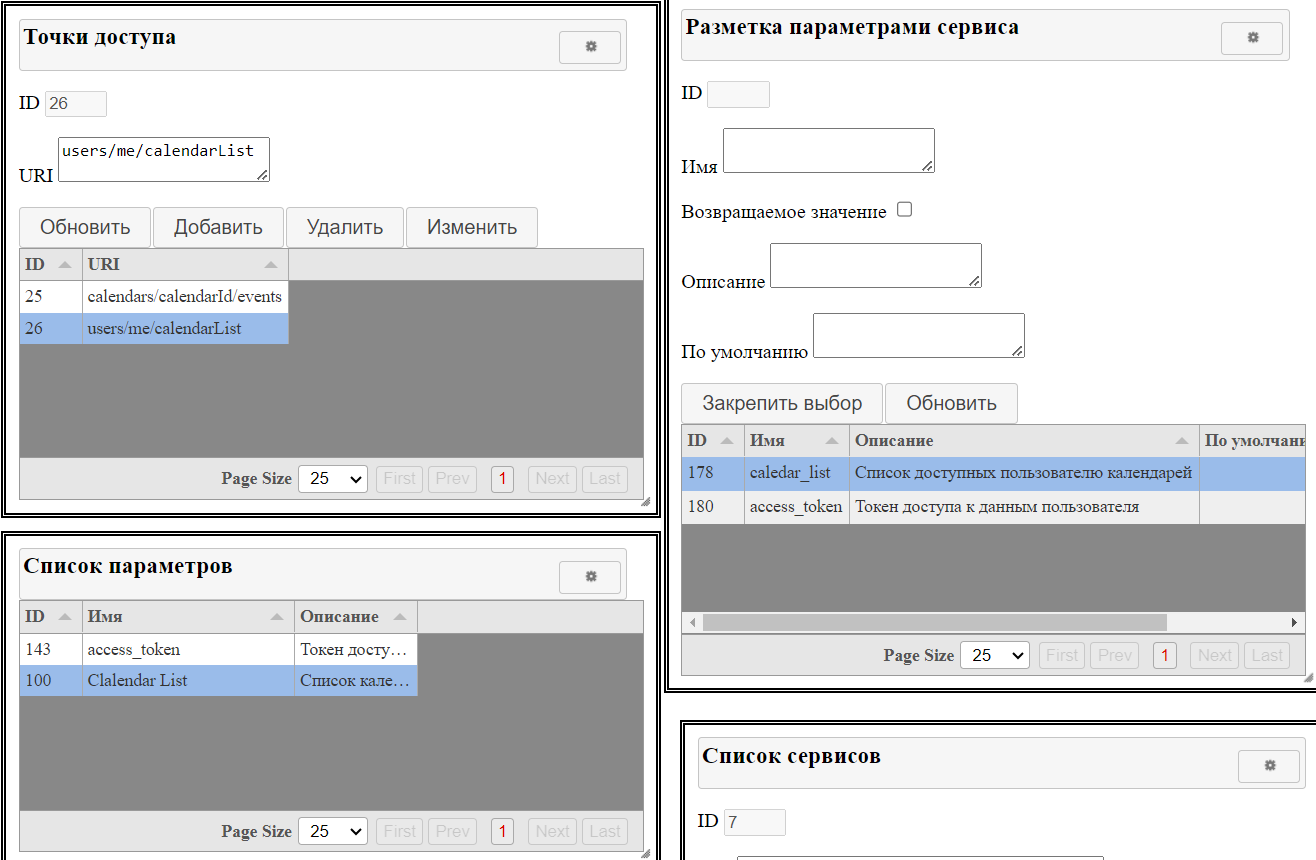


Рисунок 4.7 –Параметры функции компонента для получения списка календарей

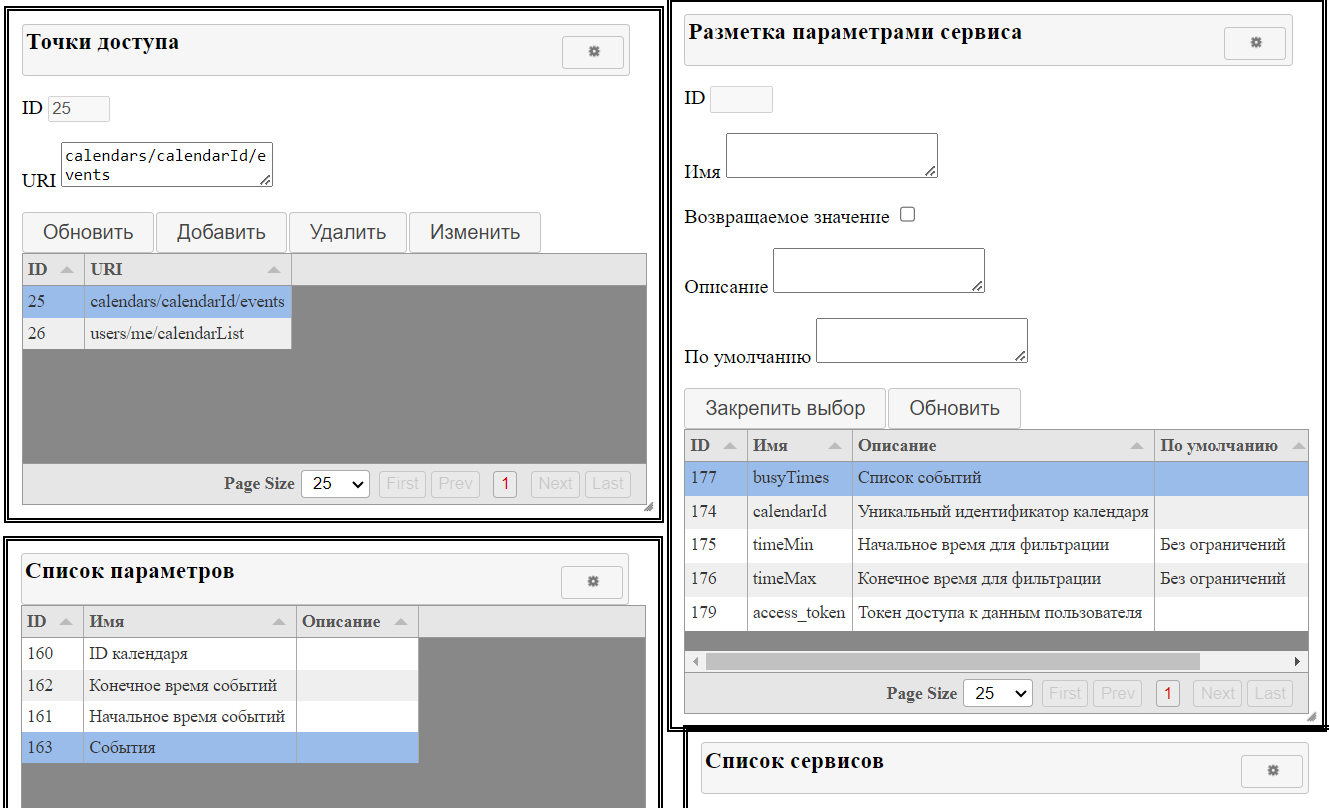


Рисунок 4.8 –Параметры функции компонента для получения списка событий

Остальные функции компонента используются для обработки результата работы с сервером и представлены на рисунках 4.9 и 4.10.

Таким образом функция «Создание полного свободного времени пользователя» получает данные из функции «Получить время событий». Также эта же функция использует результаты выполнения функции «Создание стандартного свободного времени пользователя». В итоге она объединяет данные двух этих функция, сортирует, фильтрует и преобразует уже в свободное время пользователя.

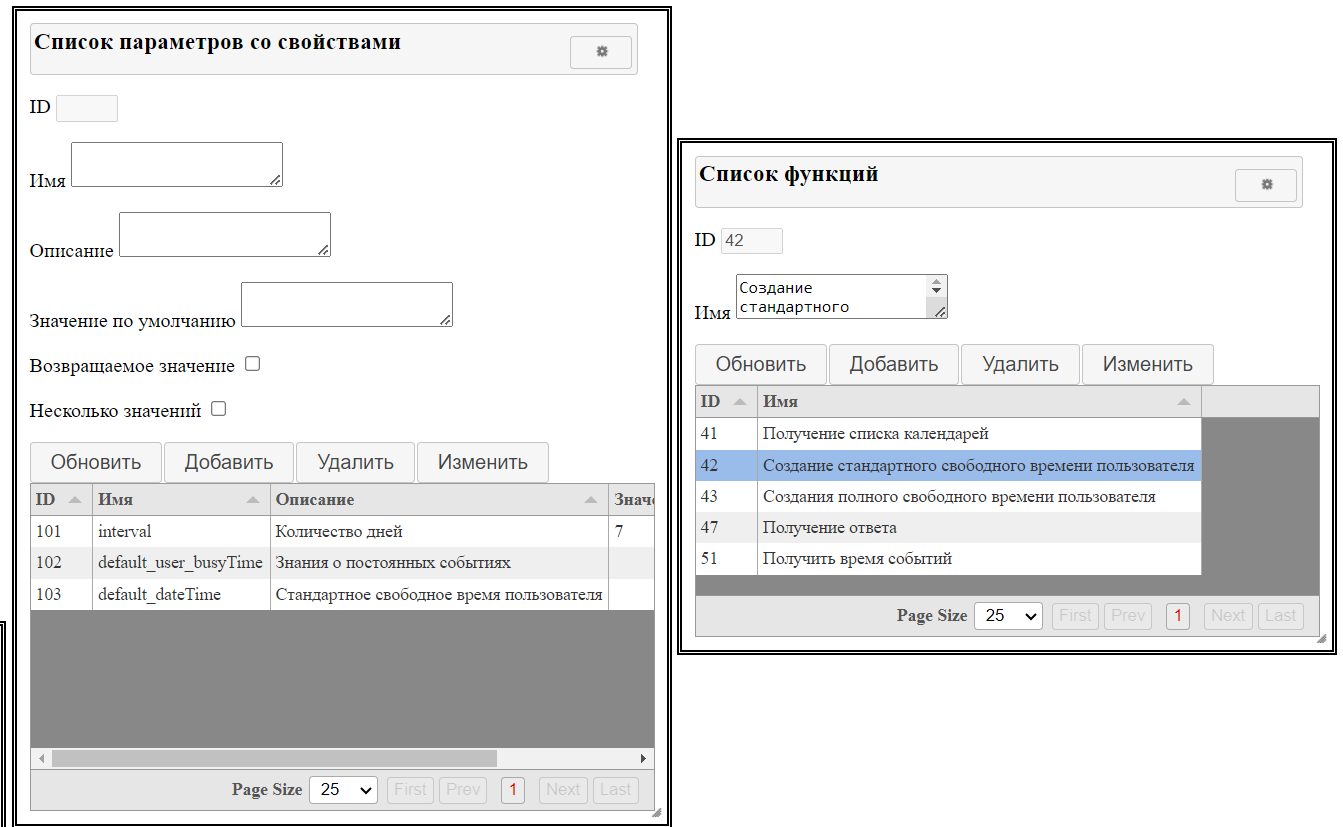


Рисунок 4.9 –Параметры функции компонента

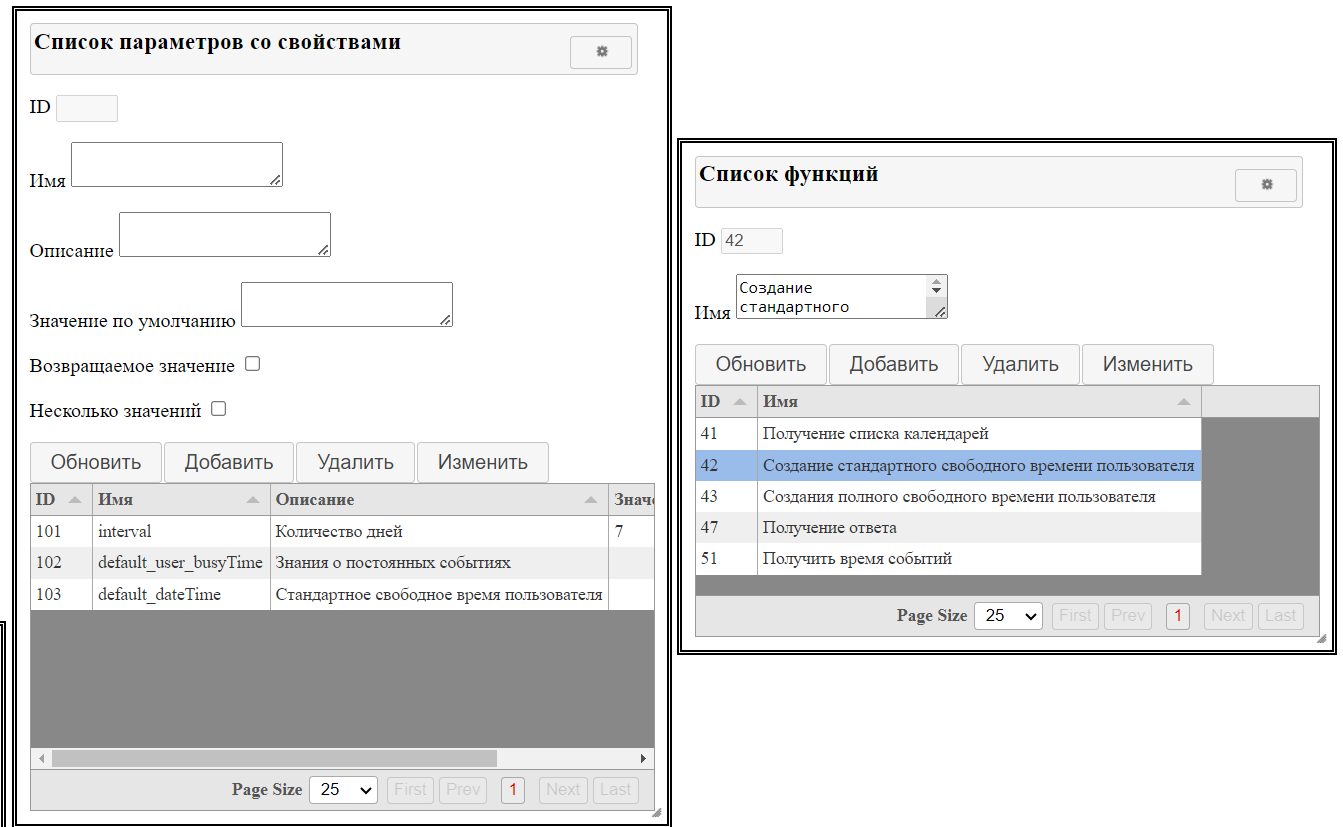


Рисунок 4.10 –Параметры функции компонента

## Тестирование

## Тест 1

Тест 1. Пользователю нужно узнать сможет ли он принять участие в событии в определенной группой людей. При этом все параметры имеются в запросе пользователя.

Начало диалога (рисунок 5.1). Фраза пользователя – «Привет». После приветствия запускается контекст диалога с пользователем.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 5.1 – Начало диалога (тест 1). Фраза «Привет»

Бот предложит описать намерение пользователя, чтобы определить возможность проведения события. Когда пользователь опишет намерение, Dialog Flow отправит запрос на сервер. На сервере запустятся правила Clips, и по результатам их работы будет сформирован ответ пользователю (рисунок 5.2). Фраза пользователя – «Сможем ли мы с друзьями собраться в кино в это воскресенье утром».

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 5.2 – Запрос о возможности проведения события (тест 1). Фраза «Сможем ли мы с друзьями собраться в кино в это воскресенье утром»

Как видно из рисунка выше, когда пользователь описывает намерение, бот отвечает, что у пользователя есть свободное время для проведения мероприятия.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 5.3 – Запрос о возможности проведения события (тест 1). Фраза «Сможем ли мы с друзьями собраться в кино в пятницу утром»

Как видно из рисунка выше, когда пользователь описывает намерение, бот отвечает, что у пользователя нет свободного времени для проведения мероприятия.

Далее идет конец диалога.

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

Рисунок 5.4 – Конец диалога (тест 1). Фраза «Рад помочь вам»

## Тест 2

Тест 2. Пользователю нужно узнать сможет ли он принять участие в событии в определенной группой людей. При этом не все параметры имеются в запросе пользователя.

Начало диалога (рисунок 5.5). Фраза пользователя – «Привет». После приветствия запускается контекст диалога с пользователем.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 5.5 – Начало диалога (тест 2). Фраза «Привет»

Бот предложит описать намерение пользователя, чтобы определить возможность проведения события. Когда пользователь опишет намерение, Dialog Flow отправит запрос на сервер. На сервере запустятся правила Clips, и по результатам их работы будет сформирован ответ пользователю (рисунок 5.6). Фраза пользователя – «Сможем ли мы с друзьями собраться в это воскресенье».

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 5.6 – Запрос о возможности проведения события с отсутствующим параметром времени (тест 2). Фраза «Сможем ли мы с друзьями собраться в это воскресенье»

Как видно из рисунка выше, когда пользователь описывает намерение, сервер выясняет, что не хватает параметра времени и возвращает контекст, после которого запускается намерение с уточнением параметра (рисунок 5.7).

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

Рисунок 5.7 – Уточнение недостающего параметра (тест 2). Фраза «Утром»

Далее пользователь передает недостающий параметр и этот параметр передается на сервер, где возвращается ответ с возможностью проведения события (рисунок 5.8).

Далее идет конец диалога.

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

Рисунок 5.8 – Конец диалога (тест 2). Фраза «Рад помочь вам»

# Заключение

В ходе выполнения работы были изучены и применены различные технологии, такие как CLIPS, PHP, DialogFlow и Protege. Работу с онтологией выполняла Арбакова Анастасия. Разработку правил и базы знаний в CLIPS и функций для php скрипта выполнял Ермоленко Антон. Разработку Dialog Flow, контекста, связей с сервером и скрипта для отправления данных из Dialog Flow осуществил Манданов Вадим.

Была создана онтология событий на основе Google Calendar, а также разработан алгоритм выбора решения на основе языка программирования CLIPS. Далее онтология была перенесена в спецификацию Dialog Flow, где были созданы сущности и структура намерений для логического диалога с пользователем. Наконец, был написан PHP скрипт, который обрабатывает данные из Dialog Flow, преобразует их в сущности CLIPS, запускает правила и обрабатывает возвращаемый результат.

В результате работы был создан функциональный голосовой помощник, способный проверять возможность планирования событий по календарю.