

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
ЮГОРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Институт цифровой экономики
Направление подготовки: 09.03.01 «Информатика и вычислительная
техника»
Профиль подготовки: «Информатика и вычислительная техника»

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
(БАКАЛАВРСКАЯ РАБОТА)**

На тему: «Разработка серверной части интеллектуального информационного
сервиса для образовательных организаций»

Студент группы 11626
Панчишин И.Р.

Научный руководитель,
доцент ИЦЭ, к.ф.-м.н.
Сафонов Е.И.

Нормоконтролер,
преподаватель
Гончаренко О.В.

Допустить к защите
Руководитель образовательной программы,
доцент ИЦЭ, к.т.н.
Самарин В.А.

«____» _____ 2020 г.

г. Ханты-Мансийск
2020 год

АННОТАЦИЯ

Бакалаврская работа 61 с., 27 рис., 19 табл., 10 источн., 10 прил.

ML, БОТ, NLP, NER, ОБРАЗОВАНИЕ, ИНФОРМАЦИОННЫЙ СЕРВИС, ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ.

Объектом исследования являются информационные системы (ИС) образовательных организаций (ОО).

Цель работы — реализовать сервис, осуществляющий передачу информации от ОО пользователю через множество коммуникационных приложений, использующий при этом методы искусственного интеллекта.

В процессе работы проводился анализ проблемы информирования, оценка требуемых ресурсов для разработки, моделирование разрабатываемого сервиса, формализация языка пользователя, сравнение методов обработки текста, программирование и тестирование сервиса.

В результате работы реализован сервис, позволяющий учащимся и работникам ОО в интерактивном режиме получать быстрый и удобный доступ к новостям, справочной информации и расписанию занятий. Многие ОО не имеют подобных решений.

Результаты работы могут применяться в школах, колледжах, университетах и других организациях с соответствующим набором данных.

Сервис готов к использованию в Югорском государственном университете (ЮГУ). Для подключения от организации требуется предоставление доступа к необходимым данным по сети. Формат данных и способ передачи не важны.

Сервис позволяет придерживаться современных требований к поиску информации. Это может оказать положительное влияние на репутацию и узнаваемость организации.

В обозримом будущем большинство ОО будут использовать искусственный интеллект в своих ИС, но на данный момент это не так.

СОДЕРЖАНИЕ

| | |
|---|----|
| ВВЕДЕНИЕ | 4 |
| 1 Анализ предметной области | 5 |
| 1.1 Анализ проблемы | 5 |
| 1.2 Исходные данные | 6 |
| 1.3 Обзор аналогов | 8 |
| 1.4 Требования к сервису | 9 |
| 1.5 Оценка ресурсов | 10 |
| 2 Проектирование системы | 15 |
| 2.1 Концептуальное проектирование | 15 |
| 2.2 Структурирование сервиса | 17 |
| 2.3 База данных | 20 |
| 2.4 Моделирование процессов | 24 |
| 2.5 Размещение системы | 27 |
| 3 Разработка системы | 30 |
| 3.1 Выбор средств разработки | 30 |
| 3.2 Организация файлов | 30 |
| 3.3 Извлечение именованных сущностей | 32 |
| 3.4 Приведение сущности к начальной форме | 36 |
| 3.5 Классификация фраз пользователей | 36 |
| 4 Тестирование | 42 |
| ЗАКЛЮЧЕНИЕ | 47 |
| СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ | 48 |
| ПРИЛОЖЕНИЕ А Требования к сервису | 49 |
| ПРИЛОЖЕНИЕ Б Конфигурация Vim | 52 |
| ПРИЛОЖЕНИЕ В Распознающая грамматика для названия занятия . . | 54 |
| ПРИЛОЖЕНИЕ Г Бот в «Яндекс Алиса» | 55 |
| ПРИЛОЖЕНИЕ Д Бот во «ВКонтакте» | 56 |
| ПРИЛОЖЕНИЕ Е Бот в «Телеграм» | 57 |
| ПРИЛОЖЕНИЕ Ж Диплом победителя конкурса докладов КМУ 2020 . | 58 |
| ПРИЛОЖЕНИЕ З Диплом участника «Славим человека труда!» 2019 . | 59 |
| ПРИЛОЖЕНИЕ И Диплом призера «IT4U» 2019 | 60 |
| ПРИЛОЖЕНИЕ К Сертификат участия «Рост UP» 2019 | 61 |

ВВЕДЕНИЕ

Не все ОО идут в ногу со временем и активно развиваются свою информационную инфраструктуру. Ресурсов на улучшение и самостоятельную разработку новых подсистем может не хватать. В таком случае выходом является использование готового решения.

В данной работе предлагается использовать разработанный сервис для решения проблем информирования, описанных в подразделе 1.1.

Сервис для пользователя является чат-ботом. Преимущество использования ботов заключается в том, что они позволяют человеку получать информацию в процессе общения — самым близким и естественным способом. Наличие голосового интерфейса позволяет пользователю меньше концентрировать свое внимание на устройстве. Актуальность чат-ботов подтверждает исследование медиаагентства «Mindshare», результаты которого были оглашены на конференции «This Year Next Year 2019». Согласно этим результатам, 37% россиян используют голосовых помощников хотя бы раз в месяц.

Для реализации сервиса определены следующие задачи:

- выполнить анализ предметной области,
- выполнить проектирование сервиса,
- реализовать основную бизнес-логику бота и адаптировать его под использование через сторонние сервисы,
- научить бота лучше понимать пользователя,
- разместить бота на выделенном сервере и внедрить его в ИС ОО,
- заняться продвижением бота, оптимизацией работы, расширением функциональности.

В работе будет рассматриваться ЮГУ (далее Университет). Однако система проектировалась таким образом, чтобы любая другая ОО со схожей структурой могла использовать ее функциональность.

Разработка сервиса была основана на гибкой методологии Канбан. Она не требует наличия команды разработчиков, позволяет визуализировать процесс работы, не вести разработку линейно и формировать требования по мере развития проекта.

1 Анализ предметной области

1.1 Анализ проблемы

Чтобы программный продукт был востребован, важно определиться, какие проблемы он будет решать или какой спрос удовлетворять.

Основным публичным информационным ресурсом Университета является веб-сайт <https://www.ugrasu.ru/>. В частности, он предоставляет пользователю новости, справочную информацию и расписание занятий. Новости также публикуются в сообществе во «ВКонтакте».

Первая потенциальная проблема — это низкий охват новостей. Причиной может быть отсутствие таргетивности. Для своих новостей Университет не использует теги и разбиение на достаточное количество категорий, не учитывает, какое отношение пользователь имеет к университету. В результате того, что пользователь получает новости, которые ему не интересны, у него исчезает желание следить за ними.

Другая причина может быть связана с плохой актуальностью новостей. Использовать одну популярную социальную сеть недостаточно. Многим пользователям удобней узнавать новости и получать уведомления в других местах, например, мессенджерах, которые по уровню популярности почти сравнялись с соц. сетями [1].

Вторая проблема может заключаться в плохой доступности справочной информации и расписания занятий. Эти данные используются ежедневно и достаточно часто, поэтому у пользователя должно быть как можно меньше преград на пути к ним. Преградами являются неадаптивный для разных устройств веб-сайт, необходимость текстового ввода, отсутствие мобильных приложений и избыточность информации. У человека, который знаком с интерфейсом веб-сайта Университета, в среднем уходит 40-60 секунд на поиск, что довольно много.

1.2 Исходные данные

Университет включает 5 институтов: гуманитарный североведения, юридический, нефти и газа, цифровой экономики и дополнительного образования.

Студенты определяются в группы с уникальными именами. Уникальность сохраняется между институтами. Имя группы может включать буквы русского алфавита в верхнем и нижнем регистре, цифры, а также дефис. На рисунке 1.1 приведены три типа имен.

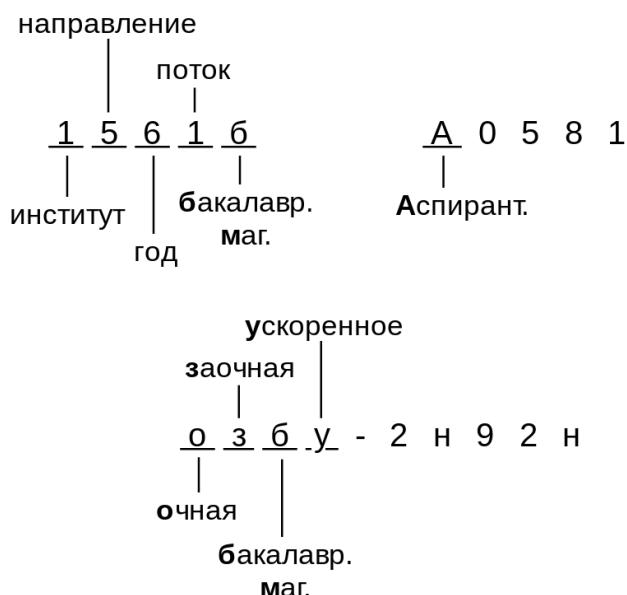


Рисунок 1.1 — Формирование имен групп

В группе могут быть дисциплины по выбору. Такие дисциплины разбиваются на блоки, и из каждого блока студент выбирает ту, которую желает посещать. Дисциплины из одного блока могут начинаться в одно время.

Группа может делиться на подгруппы. Имя подгруппы определяется ее порядковым номером. Занятия в разных подгруппах могут начинаться в одно время.

У преподавателя на занятии может присутствовать несколько групп.

Университет имеет RESTful веб-сервис, который предоставляет данные в формате JSON при запросе методом GET. Буквы русского алфавита передаются кодовыми позициями (code point) стандарта Unicode.

URL api.ugrasu.ru/api.php позволяет получить общие сведения о

сотрудниках и данные занятий в течение нескольких недель с момента запроса. Для этого используются строки запроса `view=contacts` и `view=timetable` соответственно. Структуру объекта в ответе демонстрирует таблица 1.1.

Таблица 1.1 — JSON-объект в ответе Университета

| Занятие | Сотрудник |
|---|---|
| <pre>[{ "date": "2019.03.15", "time_start": "10:15", "time_end": "11:50", "department": "ИНСТИТУТ (НОЦ) ↳ ТЕХНИЧЕСКИХ СИСТЕМ И ↳ ИНФОРМАЦИОННЫХ ↳ ТЕХНОЛОГИЙ", "group": "14516", "subgroup": "1", "discipline": "Расчёт и ↳ проектирование систем ↳ обеспечения безопасности ↳ труда", "lecturer": "Иванов И.И.", "classroom": "3/245 (230)", "type": "Л6" }, ...]</pre> | <pre>[{ "PATH": "РЕКТОРАТ", "FIO": "ИВАНОВ ИВАН ИВАНОВИЧ", "PHONE": "123-456 (приемная)", "KORP": "Адм. корп./418 ↳ (415)", "PODR_REC": "800100000001AC2", "PODR": "РЕКТОРАТ", "EMAIL": "foo@ugrasu.ru", "DOL": "Проректор по учебной ↳ работе", "RECNO": "80010000000F238", "PRIZN": "0", "RUK": "СЕМЕНОВ СЕМЕН ↳ СЕМЕНОВИЧ", "KORPPHONE": "к.Адм. корп./418 ↳ (415) (тел.123-456 ↳ (приемная))", "RATERUK": "1", "FNAIKAT": "P", "FCPERSON": "80010000000F238" }, ...]</pre> |

URL `timetable.ugrasu.ru/index.php/json` позволяет получить данные занятий в течение дня. Страна запроса должна включать параметр `group` с именем группы и может включать параметр `date` со значением в формате `dd_mm_гггг`.

1.3 Обзор аналогов

Обзор аналогов проводится для того, чтобы понять, действительно ли нужно создавать новое решение, или имеющихся решений уже достаточно. Также обзор позволяет выделить сильные стороны систем, которые решают похожие задачи, чтобы отразить их в списке требований.

«VEKучись» — мобильное приложение для Android или iOS, предоставляющее расписание занятий для студентов и преподавателей Университета. Занятия выводятся в виде списка для каждого дня. Приложение не имеет виджета для главного экрана и требует установки на устройство. В случае потери интернет-соединения доступ к расписанию выбранной группы или преподавателя сохраняется.

Иностранный компания «Virtual Spirits» оказывает услуги по подключению своего чат-бота на веб-сайты. В основном компания специализируется на интернет-магазинах, но университеты также являются ее клиентами. Имеется возможность настройки внешнего вида чата. Бота во время диалога можно прервать и писать сообщения вместо него. Обучение бота осуществляется следующим образом. Сначала определяется вопрос, несколько похожих вопросов (по желанию) и ответ. Затем для вопроса выбираются подходящие теги из списка. Каждый тег связан с набором ключевых слов, поиск которых осуществляется во вразе пользователя.

Компания «Signal Vine» критикует использование чат-ботов в университетах и считает, что они не смогут заменить общение с человеком [2]. Для общения со студентами компания предлагает свое решение со смешанным подходом, который заключается в использовании как искусственного интеллекта, так и реальных людей.

Некоторые университеты сами создают себе ботов. Такие боты, как правило, не используются за пределами одного университета, привязаны к одной соц. сети или мессенджеру и выполняют небольшое количество функций.

1.4 Требования к сервису

Требования определяются на ранней стадии создания системы. Они позволяют получить полное представление о конечном продукте, а также служат исходными данными для выполнения проектирования и разработки тестов. Согласно выбранной методологии, требования дополнялись в процессе разработки.

Для классификации требований использовалась модель FURPS+. В таблице 1.2 приведены функциональные требования. Они определяют основное поведение сервиса. Нефункциональные требования перенесены в приложение А.

Таблица 1.2 — Функциональные требования

| ИД | Описание |
|--------|---|
| FUN_1 | Бот должен сообщить о своих возможностях, когда его просит пользователь. |
| FUN_2 | Пользователь может представиться боту. В таком случае бот будет учитывать данные пользователя при последующих обращениях. |
| FUN_3 | Бот должен сообщить время до ближайшего занятия, когда его просит пользователь. |
| FUN_4 | По требованию пользователя бот должен предоставить список занятий на указанный день. |
| FUN_5 | По требованию пользователя бот должен сообщить, кто посещает занятие. |
| FUN_6 | По требованию пользователя бот должен сообщить приблизительное местоположение преподавателя. |
| FUN_7 | По требованию пользователя бот должен предоставить информацию о работнике. |
| FUN_8 | По требованию пользователя бот должен предоставить актуальную и соответствующую пользователю новость ОО. |
| FUN_9 | Бот должен уточнять необходимые ему данные у пользователя путем отправки ответного запроса. Бот позволяет пользователю отказаться от уточнения. |
| FUN_10 | Бот учитывает контекст разговора. |
| FUN_11 | Бот способен правильно ответить на вопрос, поставленный в рамках предыдущего. |
| FUN_12 | Бот считает, что его пользователем является либо преподаватель, либо учащийся. |
| FUN_13 | Бот должен поприветствовать пользователя перед началом диалога. |

Ранее в проекте требования делились на C (Customer) и D (Developer),

но от этой идеи было решено отказаться, чтобы снизить объем документации.

1.5 Оценка ресурсов

Для оценки трудоемкости и длительности разработки сервиса использовалась конструктивная модель стоимости СОСМО II. Модель основывается на статистике, собранной на множестве реальных проектов.

Прежде чем использовать модель, необходимо оценить размер проекта. Сделать это можно путем анализа его функциональности (Function Point Analysis). Функциональный поинт (ФП) — единица измерения, выражающая объем функциональности, которую система предоставляет пользователю.

Чтобы называть функции внутренними или внешними, необходимо определить границу системы. На рисунке 1.2 граница показана красной рамкой, также на нем отображены функции данных и транзакций.

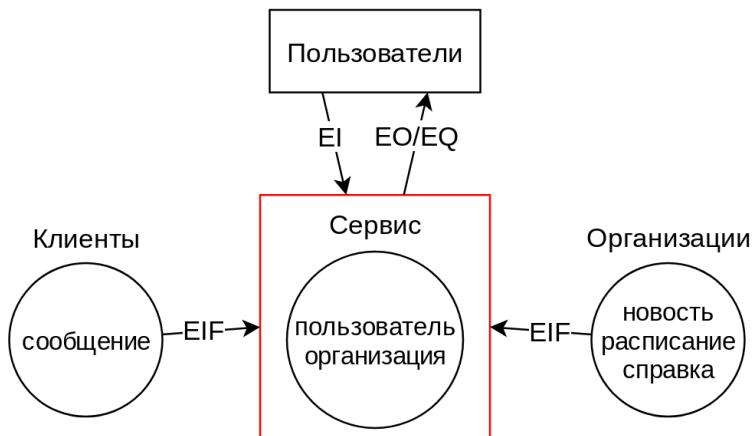


Рисунок 1.2 — Граница системы

Внешний компонент *Пользователи* осознанно был отделен от интерфейса *Клиенты*. Дело в том, что некоторые клиентские системы не отправляют сообщения пользователей самостоятельно. Вместо этого они ожидают сервер, который их заберет. Если не отделять *Пользователи* и *Клиенты*, то можно неправильно интерпретировать диаграмму и увидеть в ней ошибочное выделение параметров транзакции в виде файла.

На основе рисунка составлена таблица 1.3, содержащая найденные функции и их объем. Объем зависит от сложности данных, с которыми рабо-

тает функция. Соответствие объема каждого типа функции и весовых коэффициентов можно найти в описании методики анализа [3]. Всего получено 60 поинтов.

Таблица 1.3 — Стоимость функций

| № | Function | DET's | RET's / FTR's | Points |
|---|-----------------------------------|--|--|--------|
| 1 | сообщение (EIF) | текст польз., имя, фамилия, отчество, начало диалога (5) | именован. сущн., фраза польз., сессия (3) | 5 |
| 2 | пользователь (ILF) | назв. группы, подгруппа, имя, фамилия, отчество (5) | ФИО, группа (2) | 7 |
| 3 | организация (ILF) | назв. орган., тип орган., длитель. занятия (3) | организация (1) | 7 |
| 4 | новость (EIF) | заголовок, ссылка, тег, отдел (4) | новость (1) | 5 |
| 5 | расписание (EIF) | назв. занятия, тип занятия, назв. группы, подгруппа, имя, фамилия, отчество, кабинет, корпус, время начала, время конца (11) | время, место, преподаватель, группа, занятие (5) | 5 |
| 6 | справка (EIF) | имя, фамилия, отчество, должность, телефон, почта, отдел, корпус, кабинет (9) | ФИО, место, контакты, достижения (4) | 5 |
| 7 | Проверка доступности сервиса (EQ) | текст сообщ., текст ответа (2) | (0) | 3 |
| 8 | FUN_1 (EO) | текст ответа, текст польз. (2) | сообщение (1) | 4 |
| 9 | FUN_2 (EI) | текст польз., назв. группы, имя, фамилия, отчество, назв. орган., подгруппа, текст ответа, уточнение данных (9) | сообщение, пользователь, образ. орган. (3) | 6 |

Продолжение таблицы 1.3

| № | Function | DET's | RET's / FTR's | Points |
|----|------------|---|--|--------|
| 10 | FUN_3 (EO) | текст ответа, текст польз., назв. занятия, тип занятия, подгруппа, назв. группы, фамилия, имя, отчество, кабинет, корпус, назв. орган., время осталось, уточнение данных (14) | сообщение, пользователь, организация, расписание (4) | 7 |
| 11 | FUN_4 (EO) | текст ответа, текст польз., подгруппа, день, назв. группы, имя, фамилия, отчество, назв. орган., назв. занятия, уточнение данных (11) | сообщение, пользователь, организация, расписание (4) | 7 |
| 12 | FUN_5 (EO) | текст ответа, текст польз., подгруппа, назв. занятия, тип занятия, назв. группы, имя, фамилия, отчество, назв. орган., уточнение данных (11) | сообщение, пользователь, организация, расписание (4) | 7 |
| 13 | FUN_6 (EO) | текст ответа, текст польз., имя, фамилия, отчество, назв. орган., кабинет, корпус, уточнение данных (9) | сообщение, организация, расписание, справка (5) | 7 |
| 14 | FUN_7 (EO) | текст ответа, текст польз., имя, фамилия, отчество, назв. орган., должность, отдел, корпус, кабинет, почта, телефон, уточнение данных (13) | сообщение, организация, справка (4) | 7 |

Продолжение таблицы 1.3

| № | Function | DET's | RET's / FTR's | Points |
|----|------------|---|---|--------|
| 15 | FUN_8 (EO) | текст ответа, текст польз., назв. орган., отдел, назв. группы, ссылка, тег, заголовок, уточнение данных (9) | сообщение, организация, пользователь, новость (5) | 7 |

Один ФП соответствует 53,33 строкам кода на языке программирования Python [4]. Поэтому размер проекта, выраженный в количестве строк исходного кода, равен $53,33 \cdot 60 = 3199,8$.

Рас считать оценку трудоемкости (Person-Months) проекта до проработки архитектуры можно с помощью уравнения 1.1:

$$PM = 2,94 \cdot SIZE^E \cdot \prod_{i=1}^7 EM_i, \quad (1.1)$$

где $SIZE$ — размер, который выражается в тысячах строк исходного кода, EM — множитель трудоемкости (Effort Multiplier).

Показатель степени E определяется уравнением 1.2.

$$E = 0,91 + 0,01 \cdot \sum_{j=1}^5 SF_j, \quad (1.2)$$

где SF — коэффициент масштаба (Scale Factor).

Значения SF для проекта представлены в таблице 1.4.

Таблица 1.4 — Коэффициенты масштаба

| SF | Значение |
|-------------------------|------------------|
| PREC (Precedentedness) | 4,96 (Low) |
| FLEX (Flexibility) | 1,01 (Very High) |
| RESL (Risk Resolution) | 5,65 (Low) |
| TEAM | 4,38 (Low) |
| PMAT (Process Maturity) | 7,80 (Very Low) |

Значения EM для проекта представлены в таблице 1.5.

Таблица 1.5 — Множители трудоемкости

| <i>EM</i> | Значение |
|---|-----------------|
| RCPX (Product Reliability and Complexity) | 1,00 (Nominal) |
| RUSE (Developed for Reusability) | 1,07 (High) |
| PDIF (Platform Difficulty) | – (Very Low) |
| PERS (Personnel Capability) | 1,00 (Nominal) |
| PREX (Personnel Experience) | 0,87 (High) |
| FCIL (Facilities) | 1,30 (Very Low) |
| SCED (Required Development Schedule) | 1,00 (Nominal) |

Критерии для определения значений *EM* и *SF* можно найти в описании модели COCOMO II [5].

Расчет значения *E*:

$$E = 0,91 + 0,01 \cdot 23,8 = 1,148$$

Расчет значения *PM*:

$$PM = 2,94 \cdot 3,2^{1,148} \cdot 1,210 = 13,522$$

В описании модели сказано, что одному человеко-месяцу соответствуют 152 часа рабочего времени. Если разработчик будет работать 5 дней в неделю по 5 часов, то количество чел.-мес. будет равно $152 \cdot 13,522 / 100 = 20,553$.

Примерная стоимость разработки при зарплате 35 000 рублей: $21 \cdot 35\,000 = 735\,000$ рублей. В стоимость также можно включить аренду сервера (1000 руб/мес), покупку домена (35 руб/мес) и оплату других услуг (2000 руб/мес). Тогда примерная стоимость составит 798 735 рублей.

Длительность разработки (Time to Develop) оценивается с помощью уравнения 1.3.

$$TDEV = 3,67 \cdot PM_{NS}^{0,28+0,2 \cdot (E-0,91)} \cdot SCED, \quad (1.3)$$

где *PM_{NS}* — трудоемкость проекта без учета *SCED*.

Расчет значения *TDEV*:

$$TDEV = 3,67 \cdot 20,553^{0,28+0,2 \cdot (1,148-0,91)} \cdot 1 = 9,880$$

Примерное количество человек для работы над проектом:
 $PM/TDEV = 2,080 \approx 2$.

2 Проектирование системы

2.1 Концептуальное проектирование

Отобразить внешнюю функциональность системы позволяет диаграмма вариантов использования, которая представлена на рисунке 2.1. На диаграмме прослеживается часть требований, определенных ранее.

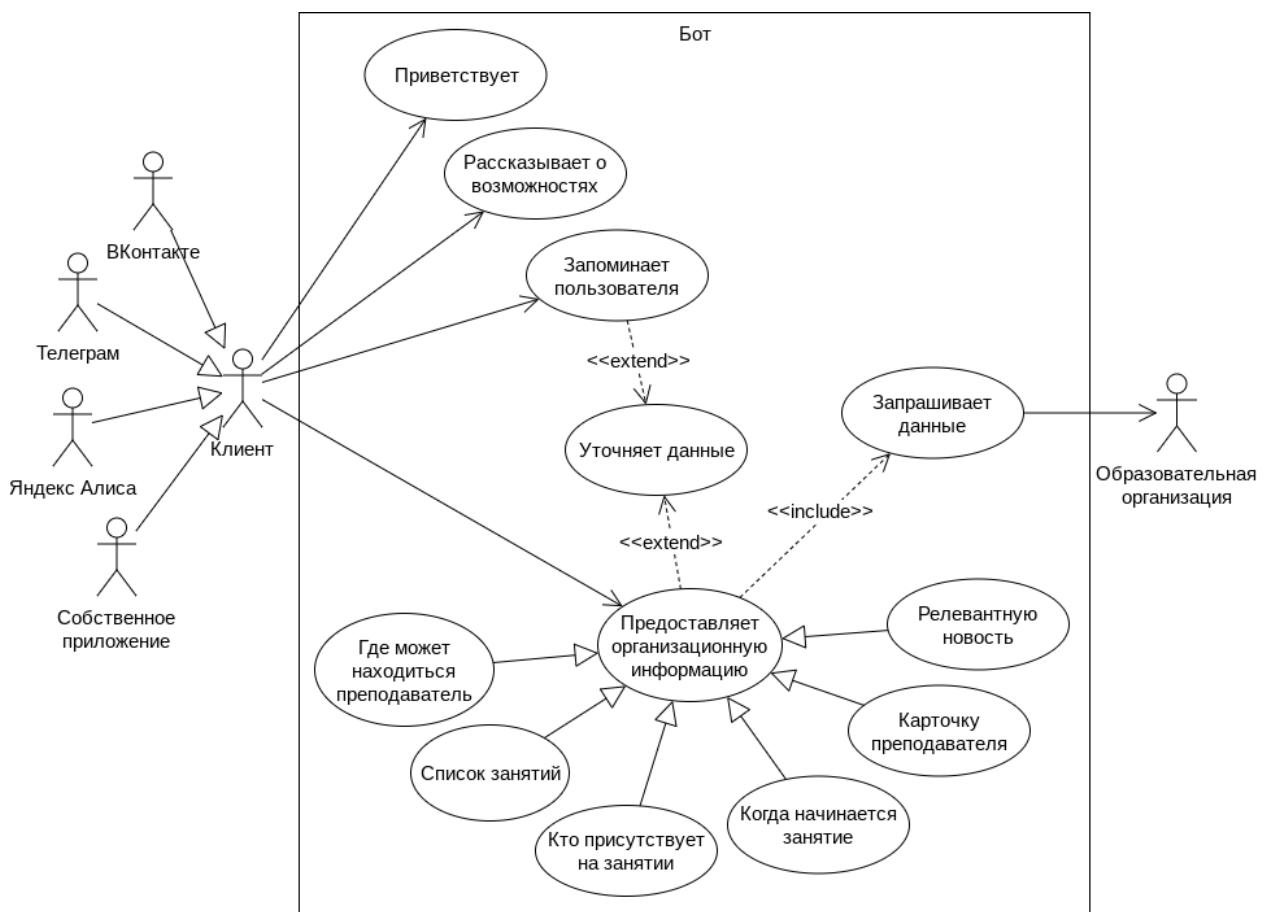


Рисунок 2.1 — Диаграмма вариантов использования бота

Актор *Клиент* является обобщением клиентских систем, которые использует пользователь, чтобы общаться с ботом. Множество специализирующихся акторов отражают требование SUP_1. Направленная ассоциация используется, чтобы явно показать инициатора и исполнителя последовательности действий.

Прецедент *Приветствует* отражает требование FUN_13. Этот и другие прецеденты описывают субъект *Бот*, который на диаграмме выделен рамкой.

Прецедент *Рассказывает о возможностях* отражает требование FUN_1.

Прецедент *Запоминает пользователя* отражает требование FUN_2. Отношение расширения показывает, что поведение precedента может быть расширено precedентом *Уточняет данные*.

Прецедент *Уточняет данные* отражает требование FUN_9. Он используется, когда субъекту для предоставления результата недостаточно тех данных, которые содержатся во фразе пользователя. Также данный precedент может использоваться, когда распознавание данных затруднительно.

Прецедент *Запрашивает данные* отражает требование INT_2.

Актор *Образовательная организация* является источником данных, которые необходимы субъекту.

Прецедент *Предоставляет организационную информацию* используется для логического объединения других precedентов, чтобы уменьшить количество связей на диаграмме и облегчить ее восприятие. Отношение включения показывает, что precedент включает поведение precedента *Запрашивает данные*. Включение и расширение создают связь, которая выражает зависимость. Данний precedент отражает требования FUN_6, FUN_4, FUN_5, FUN_3, FUN_7, FUN_8.

Концептуальная модель сервиса изображена на рисунке 2.2.

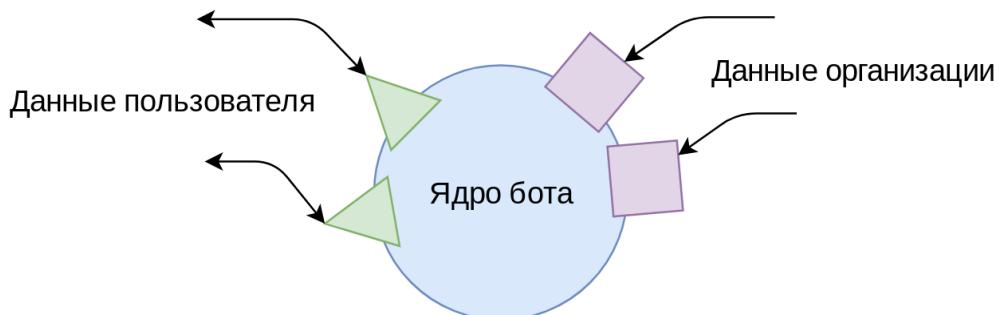


Рисунок 2.2 — Концептуальная модель сервиса

Синим цветом на рисунке показано ядро сервиса. Оно выполняет обработку данных независимо от того, к какой ОО относится пользователь и какое клиентское приложение он использует, в соответствии с DES_1.

Зеленым цветом обозначены интерфейсы, необходимые для получения данных от пользователя, в чем прослеживается требование INT_3.

Фиолетовым цветом обозначены интерфейсы, необходимые для получения данных от ОО, в чем прослеживается требование SUP_2, INT_2.

Использованные на рисунке цвета также будут использоваться на других диаграммах, которые являются декомпозицией данной. Это позволит наглядно показать соответствие элементов на нескольких диаграммах.

2.2 Структурирование сервиса

Диаграмма компонентов представлена на рисунке 2.3. Используется клиент-серверный архитектурный шаблон, бот может выступать в роли веб-хука.

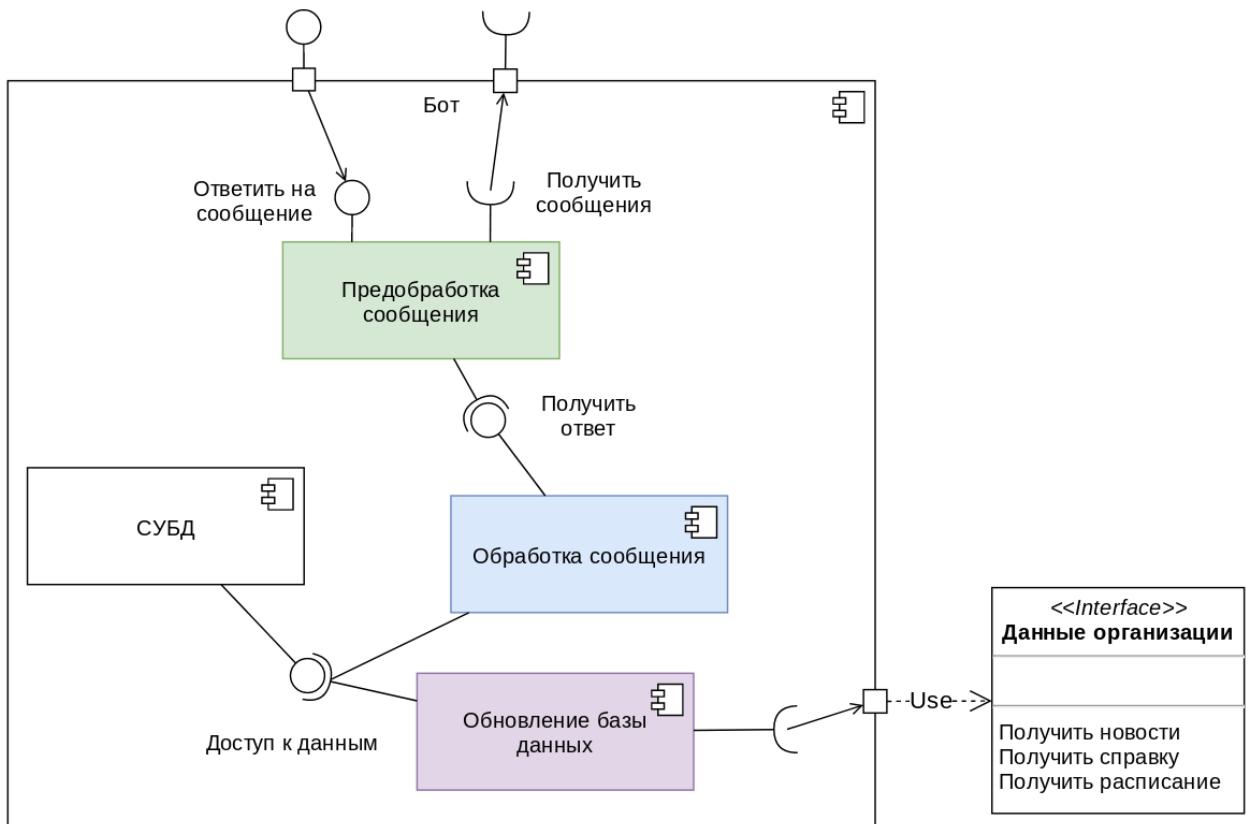


Рисунок 2.3 — Диаграмма компонентов системы

Зеленый компонент является точкой входа сообщения пользователя в систему. Сообщение либо приходит само, когда клиент использует интерфейс компонента, либо компонент запрашивает сообщения, используя интерфейс клиента. Формат полученного сообщения зависит от конкретного клиента, поэтому компонент выполняет преобразование во внутренний формат.

Синий компонент используется зеленым для получения ответа на сооб-

щение пользователя. Он является самым большим и сложным в системе, так как внутри осуществляется анализ текста, который отправил пользователь, и генерация текста, который он получит. Компоненту требуется доступ к базе данных (БД), в которой он сможет хранить данные пользователя и получать данные ОО.

Фиолетовый компонент занимается обслуживанием БД и актуализацией информации в ней. Компоненту требуется доступ к БД и интерфейс для получения данных от ОО. Здесь прослеживается требование INT_2.

Локальная БД выполняет функцию кэширования. Это необходимо, так как частое обращение к ОО может существенно снизить скорость ответа пользователю. Кроме задержек передачи данных, ОО может оказаться недоступной на какое-то время.

Более детальное и подходящее для реализации представление системы позволяет получить диаграмма модулей, представленная на рисунке 2.4.

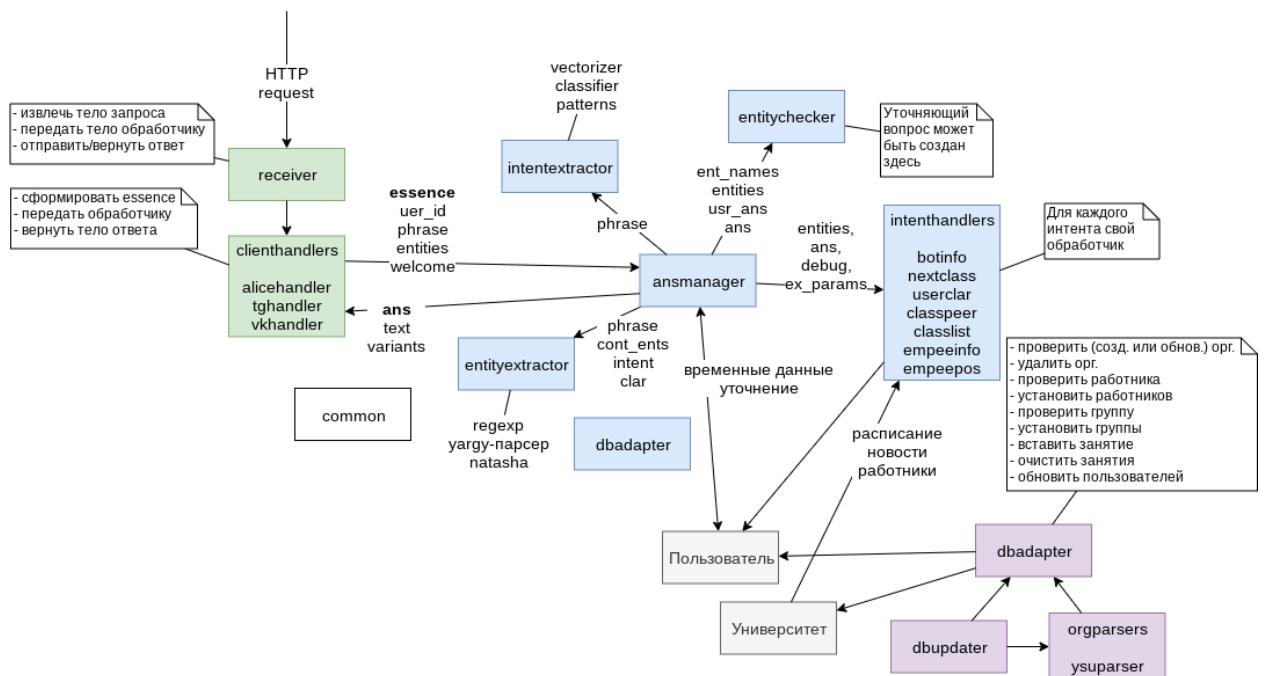


Рисунок 2.4 — Диаграмма модулей системы

Стрелки показывают направление передачи данных. Данная диаграмма не означает последовательность выполнения модулей. На диаграмме имеются заметки, отражающие функциональность некоторых модулей, которую необходимо реализовать.

Модуль *receiver* принимает сообщение от клиента по протоколу HTTP

в соответствии с INT_1. Он обеспечивает единую точку входа для всех клиентов. Модуль имеет функциональную связанность (СС = 10).

Пакет модулей *clienthandlers* используется для преобразования формата сообщения. Для каждого клиента имеется свой модуль. Все модули в пакете имеют СС = 10 и сцепление с модулем *receiver* по данным (СЦ = 1). Остальные модули имеют сцепление по образу (СЦ = 3).

Модуль *ansmanager* отвечает за формирование данных, которые пользователь получит в виде ответа. СС = 10.

Модуль *intentextractor* используется для извлечения намерения из фразы пользователя. Для этого внутри модуля используется сравнение с шаблоном, классификатор и веторизатор. СС = 10.

Модуль *entityextractor* используется для извлечения именованных сущностей из фразы пользователя. СС = 7.

Модуль *entitychecker* используется для проверки именованных сущностей. Под проверкой имеется ввиду приведение к начальной форме, если сущность имеется, или запрос конкретной сущности у пользователя. СС = 10.

Модуль *dbadapter* предоставляет интерфейс для доступа к БД на подходящем уровне абстракции. СС = 1.

Пакет *intenthandlers* содержит модули, которые выполняют обработку намерений. СС = 10.

Модуль *dbupdater* используется для обслуживания БД. СС = 10.

Пакет модулей *orgparsers* используется для кэширования данных. Для каждой ОО имеется свой модуль, который запрашивает у нее данные и помещает их в локальную БД. СС = 10.

Модуль *cotton* содержит элементы, которые могут использоваться как при обслуживании базы данных, так и веб-приложением, а также элементы, которые нельзя отнести к одному модулю. Данный модуль можно считать ошибкой проектирования. СС = 0.

Именованные сущности, выделяемые из фразы пользователя, определены в таблице 2.1. * означает, что поле может отсутствовать.

Таблица 2.1 — Именованные сущности

| Имя | Поле | Пример |
|-------------|---|--|
| Работник | фамилия* имя* отчество* внешний номер* | Меня зовут Иванов Иван Иванович. |
| Группа | имя | Я учусь в группе 1162б. |
| День | смещение* день недели* | Какие занятия были вчера? |
| Занятие | название* вид* | Кто ведет лекции по системному анализу? |
| Подгруппа | имя | Подгруппа 2. |
| Организация | имя | Я учусь в Югорском государственном университете. |
| Место | кабинет корпус* | Кабинет 312 в корпусе 4 свободен? |

2.3 База данных

Изначально для реализации базы данных был использован Redis. Выбор был сделан в пользу этой СУБД, так как она позволяет обеспечить высокую производительность за счет хранения базы данных в оперативной памяти. Такой подход к хранению данных имеет низкую надежность, так как отказ сервера приведет к потере изменений, выполненных с момента последней синхронизации данных на диск. Но это не является проблемой так как сервис не имеет требований для обеспечения надежного хранения данных.

В качестве ключа выступал идентификатор пользователя: имя преподавателя или группы. Значением являлись ассоциированные с ключом данные в формате JSON. Например, ключ *Иванов Иван Иванович* мог содержать значение, приведенное в листинге 2.1

Листинг 2.1 — Объект в базе данных

```
1  {
2      "timetable": [
3          [
4              {
5                  "campus": "3",
6                  "room": "449",
7                  "start": "10:15",
8                  "end": "11:50",
9                  "discipline": "Разработка и сопровождение ИС"
10             }
11         ],
12         []
13     ],
14     "info": {
15         "name": "Иванов Иван Иванович",
16         "email": "foo.bar@gmail.com",
17         "phone": null,
18         "post": "Преподаватель",
19         "unit": "ИНСТИТУТ ЦИФРОВОЙ ЭКОНОМИКИ",
20         "campus": null,
21         "room": null
22     }
23 }
```

Со временем структура данных становилась сложнее и работа с ними по модели «ключ-значение» требовала больших усилий. Поэтому было решено сменить СУБД на классическую — реляционную.

Логическая модель базы данных представлена на рисунке 2.5.

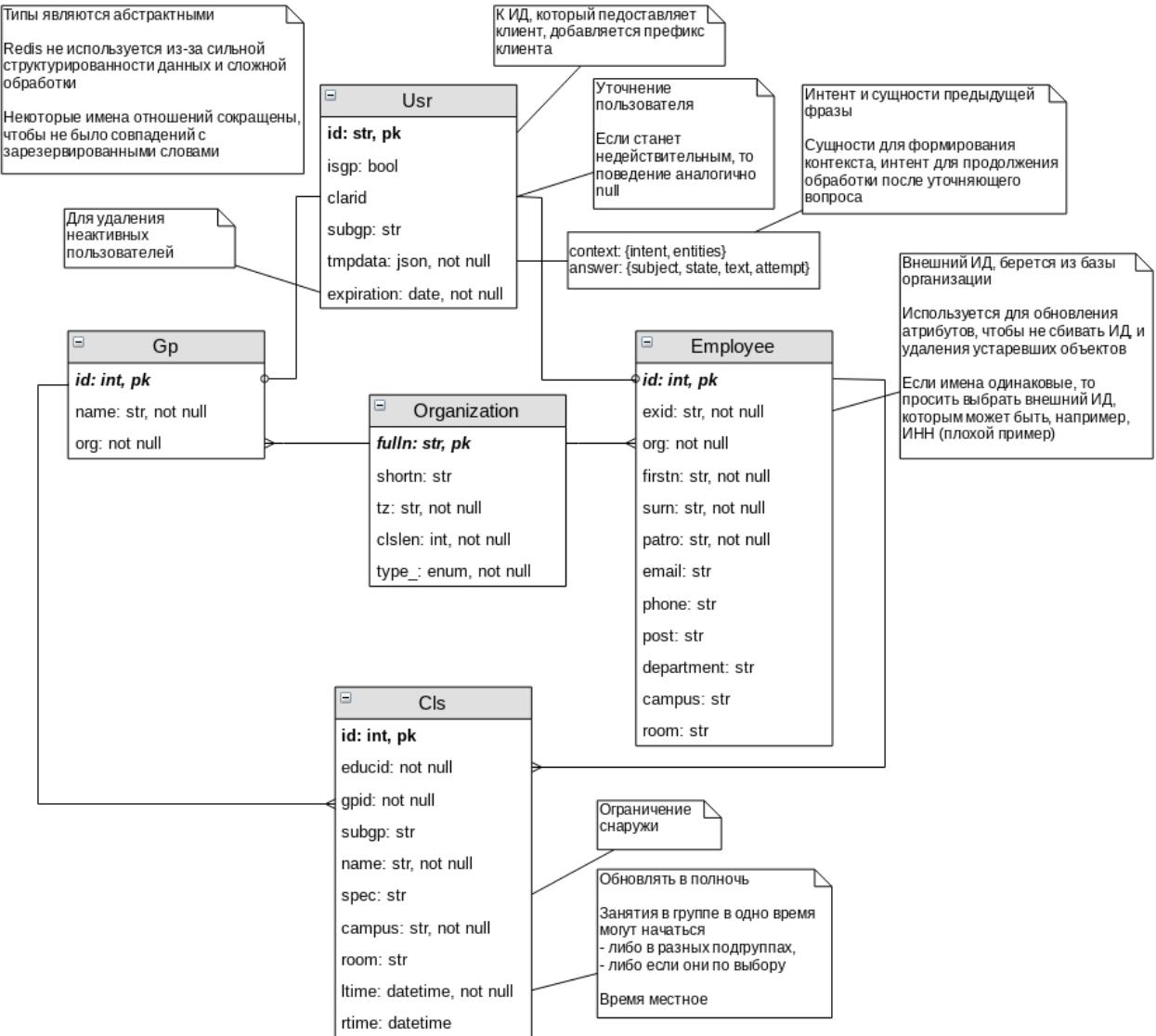


Рисунок 2.5 — Логическая модель базы данных

В системе используется 5 сущностей: пользователь (*Usr*), организация (*Organization*), группа (*Gp*), работник (*Employee*) и занятие (*Cls*).

Пользователь уточняется данными, полученными от ОО. Ссылкой на эти данные служит атрибут *clarid*. Чтобы определить, относится ссылка к группе или работнику, используется атрибут *isgp*.

Сохранять состояние диалога с пользователем позволяет атрибут *tmpdata*. Данными, которые требуют временного хранения, являются: контекст разговора, ответный вопрос и список непроверенных именованных сущностей.

Атрибут *expiration* используется во время обслуживания базы данных для удаления старых пользователей.

Пользователю не обязательно называть свою подгруппу, поэтому атрибут *subgp* связан с ним, а не с группой.

Работник имеет внешний номер *exid*, который является его уникальным ИД в базе данных ОО. Он необходим для сопоставления полученных записей с уже имеющимися.

Атрибут *type_* организации позволяет определить, является она школой, колледжем или университетом. Это позволит использовать меньше общих формулировок в ответном сообщении.

Скорее всего в предоставленном расписании занятий будет указано местное время. Поэтому необходимо знать часовой пояс организации. Его определяет атрибут *tz*.

Продолжительность занятий в ОО определяется локальными положениями. Знать только тип организации не достаточно, поэтому имеется атрибут *clslen*, определяющий продолжительность занятия.

Во время проектирования отдавалось предпочтение простым первичным ключам, вместо составных, так как с ними будет проще работать на этапе реализации сервиса. Полученная база данных соответствует 3 нормальной форме.

2.4 Моделирование процессов

Взаимодействие между сервисом и клиентом на примере использования платформы «Яндекс Диалоги» показано на рисунке 2.6.

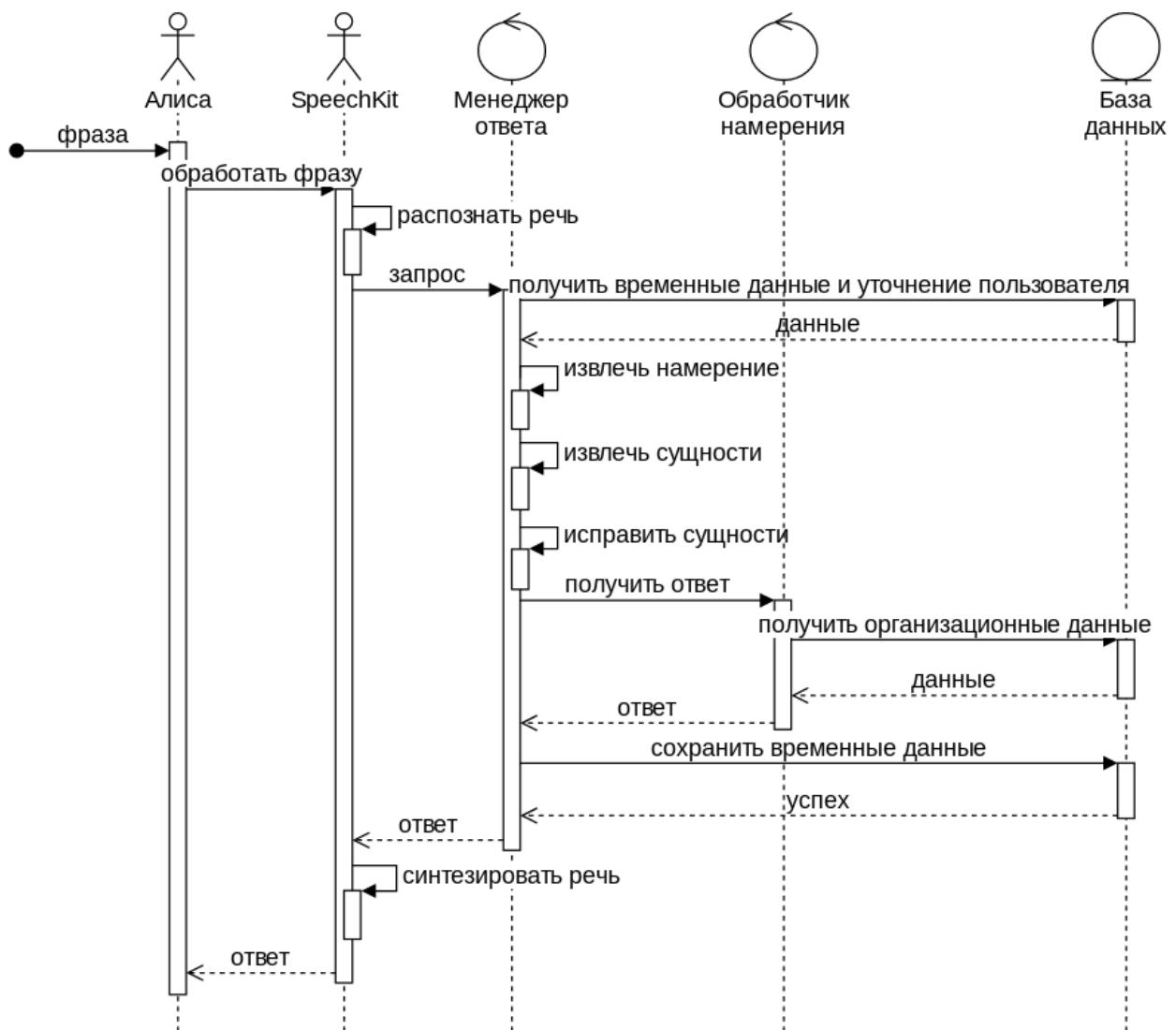


Рисунок 2.6 — Последовательность обработки сообщения

Пользователь вводит фразу в текстовом или аудио формате в клиентское приложение «Яндекс Алиса». Затем она передается на сервера распознавания речи «Yandex SpeechKit». Там голос преобразуется в текст, в нем выделяются слова и общие именованные сущности. Затем данные передаются в менеджер ответа сервиса. Он управляет процессом формирования ответа пользователю. Сначала извлекаются данные пользователя из базы данных, затем распознается намерение и формируется набор именованных сущностей.

Набор проходит процедуру проверки, которая заключается в приведении сущностей в начальную форму и дополнении недостающих сущностей. После этого вызывается обработчик намерения, на вход которого поступает проверенный набор именованных сущностей. Результатом работы обработчика является ответ пользователю, который обратным путем к нему возвращается.

Согласно протоколу работы навыка Алисы, интервал активности менеджера ответа не должен превышать 3 секунды, иначе ответ не будет принят.

На рисунке 2.7 представлена декомпозиция обработки одного из намерений. Обработка сводится к работе с сущностями и базой данных.

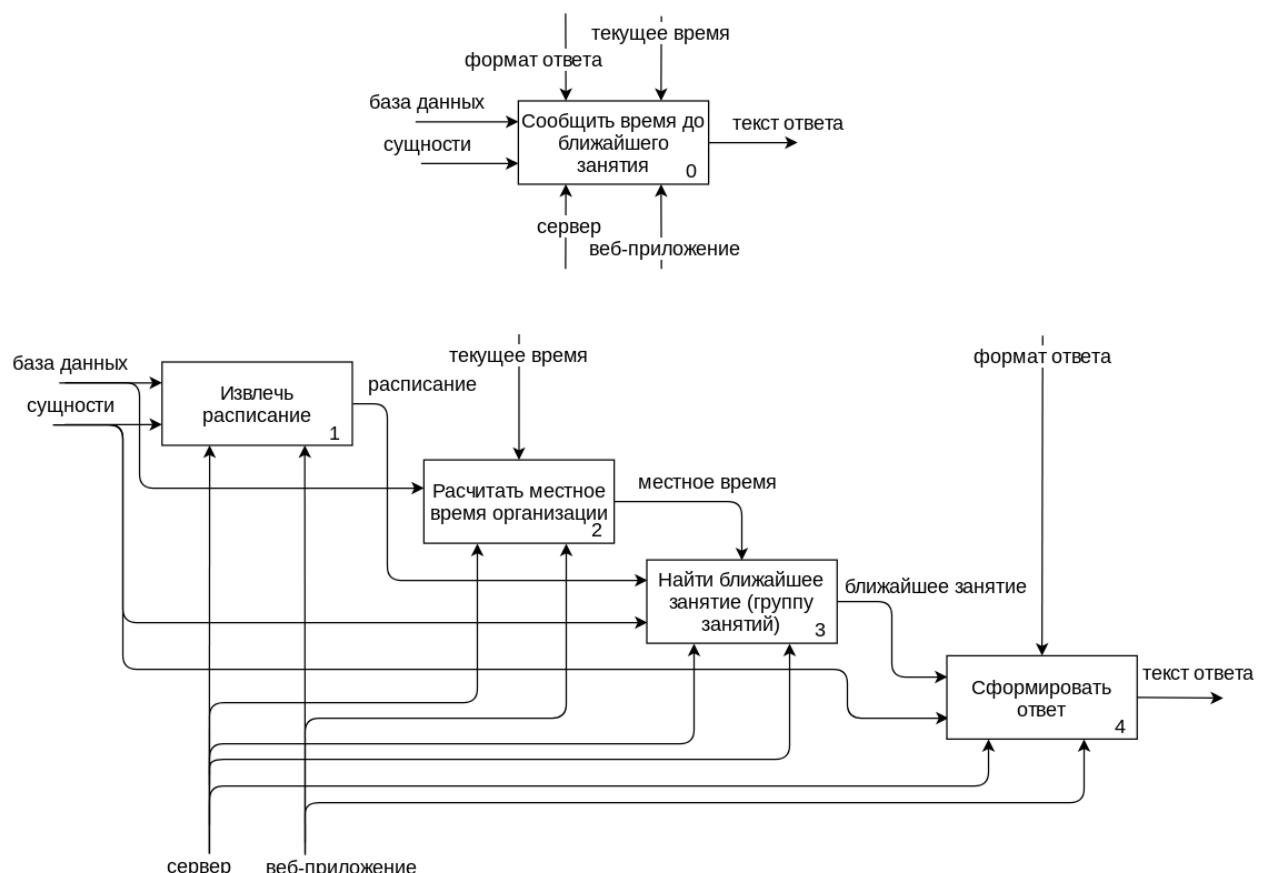


Рисунок 2.7 — Контекстная IDEF0-диаграмма и ее декомпозиция для обработки намерения

На рисунке 2.8 показаны этапы извлечения намерения из фразы пользователя. Сначала фраза нормализуется, затем из нее извлекается семантический вектор. Он при помощи классификатора сопоставляется с кодом намерения, который преобразуется в имя намерения.

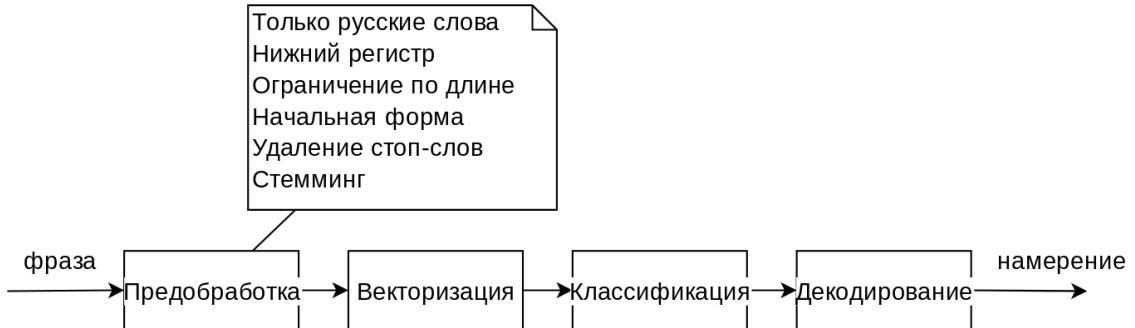


Рисунок 2.8 — Этапы извлечения намерения

Определены несколько наборов именованных сущностей: сущности, найденные во фразе пользователя (*Фраза*), сущности, которые распознал клиент (*Клиент*), сущности, которые были сохранены в контекст (*Контекст*) и сущности из имеющейся информации о пользователе (*Уточнение*). Как из нескольких наборов формируется итоговый, который передается обработчику намерения, показано на рисунке 2.9.

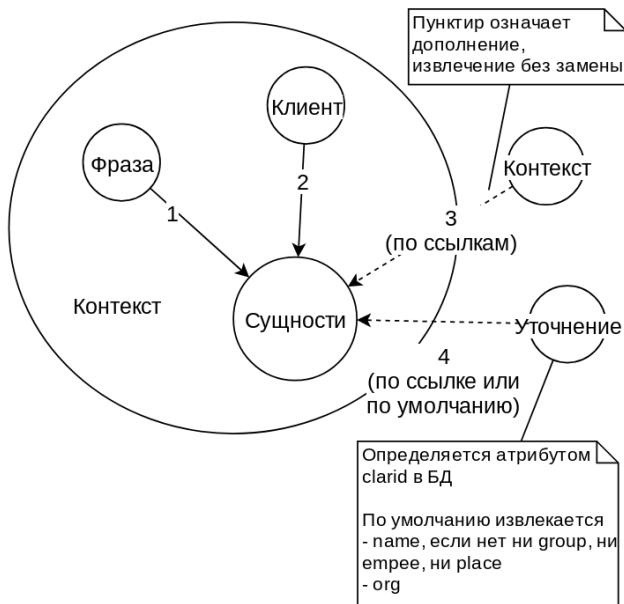


Рисунок 2.9 — Поиск именованных сущностей

Наборы *Фраза* и *Клиент* образуют новый контекст. Сущности из *Контекст* берутся только по ссылке. Сущности из *Уточнение* берутся только по ссылке или, если в итоговом наборе они отсутствуют.

Таким образом, пользователь может не произносить каждый раз информацию о себе и ссылаться на сущности из прошлой фразы с помощью местоимений и наречий. Ссылаться на контекст необходимо явно.

2.5 Размещение системы

Изначально сервис проектировался для размещения на сервере ОО. Чтобы максимально упростить процесс внедрения было решено использовать свой сервер, который будет обслуживать несколько ОО.

Сервер находится под управлением ОС GNU/Linux. Она является свободной и обеспечивает гибкое и удобное решение задач, связанных с системным администрированием.

В качестве дистрибутива используется Ubuntu 18.04 LTS. Он достаточно популярен, благодаря чему можно с легкостью найти решение многих проблем. Дистрибутив обеспечивает поддержку определенных версий программ, что делает его достаточно надежным для использования на сервере.

На рисунке 2.10 представлена конфигурация сервера.

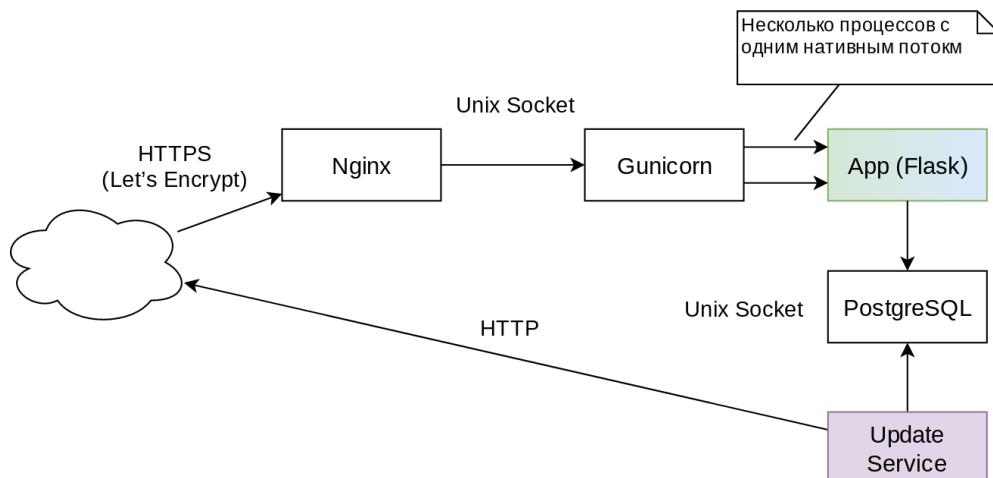


Рисунок 2.10 — Конфигурация сервера

Сервер принимает запросы по протоколу HTTPS. Необходимые для безопасной передачи данных сертификаты выдает центр сертификации Let's Encrypt.

В качестве веб-сервера и реверсивного прокси используется Nginx. Он маршрутизирует запросы в специальный Unix Socket, откуда оничитываются сервером WSGI-приложений Gunicorn и передаются в веб-приложение, построенное при помощи микро-фреймворка Flask.

И Nginx, и Gunicorn для обработки запросов используют модель рабочих процессов (worker process). В данной конфигурации Nginx использует

зует один рабочий процесс. Одновременную обработку нескольких запросов обеспечивает Gunicorn. Он настроен на использование нескольких процессов, количество которых зависит от количества ядер процессора (включая виртуальные) и расчитывается по формуле: $2 * CPU + 1$. Используются процессы класса Gevent. Особенностью этого класса является использование «зеленых» потоков.

Зеленые потоки создаются и переключаются самим приложением, в отличие от нативных потоков, которыми управляет ОС. Возможность в нужный момент переключать контекст на другой поток позволяет использовать синхронный и даже потоконебезопасный код. Однако, нужно оставаться внимательным к критическим секциям и не допускать, например, одновременного использования одного сокета несколькими потоками. Как правило, переключение осуществляют во время IO-блокировки.

Зеленые потоки работают внутри одного нативного потока. Это значит, что все они могут выполняться только на одном ядре процессора, а значит не могут быть параллельными. Поддержка нескольких ядер в данной архитектуре обеспечивают несколько рабочих процессов, где каждый имеет один нативный поток.

Для лучшей масштабируемости Nginx может выступать в роли балансировщика нагрузки. Это позволит создать несколько экземпляров Gunicorn, когда пользователей будет достаточно много.

Данные находятся под управлением СУБД Postgres SQL. Для их обновления используется Systemd-таймер.

Раньше для установки сервиса использовалась система виртуализации на уровне ОС Docker. От нее было решено отказаться, чтобы упростить взаимодействие между компонентами системы. Другой, более простой вариант конфигурации сервера, представлен на рисунке 2.11.

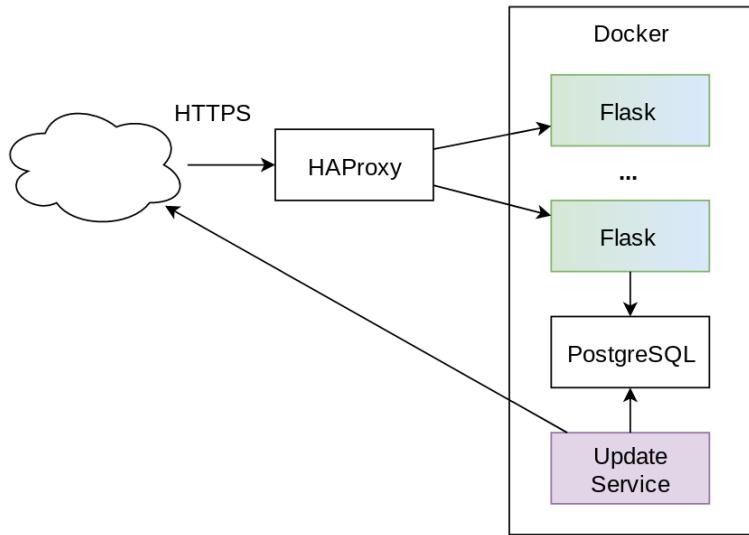


Рисунок 2.11 — Упрощенный вариант конфигурации

3 Разработка системы

3.1 Выбор средств разработки

Для реализации сервиса выбран язык программирования Python. Выбор связан с набором ПО, необходимым для функционирования сервиса, который был определен на последнем этапе проектирования. Также Python является наиболее подходящим языком для работы с искусственным интеллектом, для него имеется много библиотек, нацеленных на решение конкретных проблем.

Для реализации установки сервиса выбран язык Bash. Этот язык используется в одноименной командной оболочке, которая является стандартной для выбранного дистрибутива.

Для управления версиями проекта, куда кроме исходного кода входит документация, выбрана система Git. Для архивирования и совместной разработки выбран хостинг проектов Bitbucket. Выбор сделан за счет возможности создавать бесплатные закрытые репозитории с незначительными на данном этапе разработки ограничениями.

Для написания исходного кода выбран текстовый редактор Vim. Он удобен тем, что есть практически во всех дистрибутивах GNU/Linux, ориентирован на использование без компьютерной мыши и не требует графического интерфейса. Его удобно использовать для разработки как на личном компьютере, так и на удаленном сервере. Минимальная конфигурация Vim приведена в приложении Б.

Для безопасного подключения к удаленному серверу, где располагается сервис, используется протокол SSH. Графический интерфейс не используется на сервере, поэтому доступ к рабочему столу по протоколам RDP или RFB не осуществляется.

3.2 Организация файлов

Схема организации файлов проекта в виде дерева показана в листинге 3.1.

Листинг 3.1 — Схема организации файлов

```
1  |-- preinstall.sh
2  |-- install.sh
3  |-- crtinstall.sh
4  |-- config.sh
5  |-- config.py.in
6  |-- dbupdater.service.in
7  |-- dbupdater.timer
8  |-- initdb.sql
9  |-- nginx.conf.in
10 |-- plan.drawio
11 |-- postgresql.changes.conf
12 |-- README_en.md
13 |-- README.md
14 |-- doc
15 |-- test
16 |-- requirements.txt
17 |-- server
18 |  |-- common.py
19 |  |-- data
20 |  |
21 |  |  ...
22 |  |  |-- intents
23 |  |  |  ...
24 |  |  `-- югу
25 |  |-- db
26 |  |  |-- common.py -> ../common.py
27 |  |  |-- config.py -> ../config.py
28 |  |  |-- dbupdater.py
29 |  |  ...
30 |  |-- site
31 |  `-- webapp
32 |    |-- common.py -> ../common.py
33 |    |-- config.py -> ../config.py
34 |    |-- ansmanager.py
35 |    ...
36 |    |-- clienthandlers
37 |    |-- grammars
38 |    `-- intenthandlers
39 `-- webapp.service.in
```

Файл *preinstall.sh* содержит сценарий подготовки сервера к установке. В ходе выполнения создается специальный пользователь, от имени которого будет работать сервис, устанавливаются зависимости сервиса. Как правило, этот файл выполняется один раз, на новом сервере.

install.sh содержит основной сценарий установки. В ходе выполнения на сервер копируются файлы сервиса, осуществляется конфигурация систем, с которыми он взаимодействует. Допускается повторное выполнение файла.

crtinstall.sh содержит сценарий установки сертификатов для сервиса.

config.sh содержит параметры установки сервиса.

config.py.in является шаблоном файла с параметрами, который использует сервис во время работы.

dbupdate.service.in является шаблоном сервиса Systemd, который выполняет обновление базы данных.

dbupdate.timer является таймером Systemd, по которому обновляется база данных.

initdb.sql используется для создания структуры базы данных.

nginx.conf.in является шаблоном конфигурации Nginx.

plan.drawio содержит визуальное описание работы сервиса.

postgresql.changes.conf содержит изменения в конфигурации PostgreSQL.

README_en.md содержит краткое описание сервиса на английском языке.

README.md содержит краткое описание сервиса на русском языке.

Директория *doc* содержит документацию к сервису.

Директория *test* содержит юнит-тесты.

requirements.txt содержит список необходимых библиотек языка Python.

Директория *server* содержит исходный код сервиса. Директория *data* содержит обучающие выборки (директория *intents*) и архивы данных, полученных от ОО (директория *югу*). Директория *db* содержит модули для обновления БД. Директория *site* содержит веб-сайт с описанием бота. Директория *webapp* содержит модули бота.

3.3 Извлечение именованных сущностей

Именованные сущности — это полезные данные, которые извлекаются из фразы пользователя и используются для формирования ответа. Для извлечения некоторых сущностей можно использовать готовое решение. Им может быть библиотека для ЯП или сервис, который предоставляет услуги распознавания. Например, сервис «GATE Cloud» может использоваться для распознавания имен, дат, адресов и других сущностей в тексте [6]. Пример запроса к сервису приведен в листинге 3.2.

Листинг 3.2 — Распознавание имени в тексте

```
1 import requests
2 import json
3
4 url = 'https://cloud-api.gate.ac.uk/process-document/' \
5     'russian-ner-with-inflexional-gazetteer' \
6     '-and-orthomatcher'
7 headers = {'Content-Type': 'text/plain'}
8 document = 'Когда пара у Ивана Романовича?'
9
10 response = requests.post(url, data=document.encode('utf-8'),
11                           headers=headers)
11 print(json.dumps(response.json(), indent=2))
```

Для сущностей из конкретной предметной области готового решения может не найтись. Можно выделить два подхода к извлечению именованных сущностей, которые могут сочетаться: статистический и на основе правил. Статистический подход относится к машинному обучению и связан с формированием обучающей выборки, от полноты и качества которой зависит качество распознавания. Подход на основе правил заключается в описании структуры именованной сущности. Чем точнее будет найдена и описана структура, тем лучше будет распознавание.

В данной работе для извлечения именованных сущностей используются результаты поиска клиентского приложения, а также формальные языки.

К числу формальных языков относится регулярный язык. Он хорошо подходит для распознавания слов, которые имеют специальный синтаксис. В данной работе регулярный язык используется для распознавания группы, подгруппы, кабинета и корпуса. Регулярный язык можно определить автоматом, регулярной грамматикой или регулярным выражением. В данной работе используется последний способ в силу своей простоты. За поддержку регулярных выражений отвечает стандартная библиотека *re*.

К числу формальных языков также относится контекстно-свободный язык. Он определяется контекстно-свободной грамматикой, которую описывают формулы 3.1.

$$G = \langle V, \Sigma, P, S \rangle, \\ S \in V, \\ P = \left\{ \begin{array}{l} A \rightarrow \alpha, \\ A \rightarrow \epsilon \end{array} \middle| \begin{array}{l} A \in V, \\ \alpha \in (V \cup \Sigma)^* \end{array} \right\}, \quad (3.1)$$

где G — КС-грамматика,

V — конечное множество нетерминалов,

Σ — конечное множество терминалов,

P — множество правил вывода,

S — аксиома,

ϵ — пустое слово.

Таким образом, КС-грамматика — это кортеж из множества нетерминалов, терминалов, правил и аксиомы. Аксиома лежит во множестве нетерминалов. Правила определяются как замена нетерминала на строку, которая может включать как терминалы, так и нетерминалы или быть пустой.

Язык, заданный грамматикой, — это множество строк из терминалов, которые можно получить из аксиомы за одну или несколько замен. Формальным определением является уравнение 3.2.

$$L(G) = \left\{ w \in \Sigma^* \mid S \xrightarrow{*} w \right\}, \quad (3.2)$$

где w — слово.

В данном проекте было решено использовать распознающие КС-грамматики. Для русского языка можно выделить два инструмента, которые позволяют их реализовать: Ярги-парсер и Томита-парсер. Оба имеют открытый исходный код и подходят для коммерческого использования, однако, имеются существенные различия:

- Ярги реализует алгоритм Эрли, Томита использует GLR-парсер (обобщенный восходящий магазинный анализатор);
- Ярги использует Pymorph2 для работы с морфологией, Томита делает это самостоятельно.

Остальные различия приведены в обзоре Ярги-парсера [7].

В данной работе было решено использовать Ярги-парсер. Он относится к проекту Natasha, куда также входит одноименная библиотека готовых

грамматик для парсера. Эти грамматики были использованы для распознавания именованных сущностей *Работник* и *Организация*. Для сущностей *День*, *Занятие*, а также для поиска ссылок на контекст грамматики были написаны самостоятельно.

Реализованная грамматика для *Занятие* представлена в виде дерева в приложении В. Пример дерева разбора представлен на рисунке 3.1.

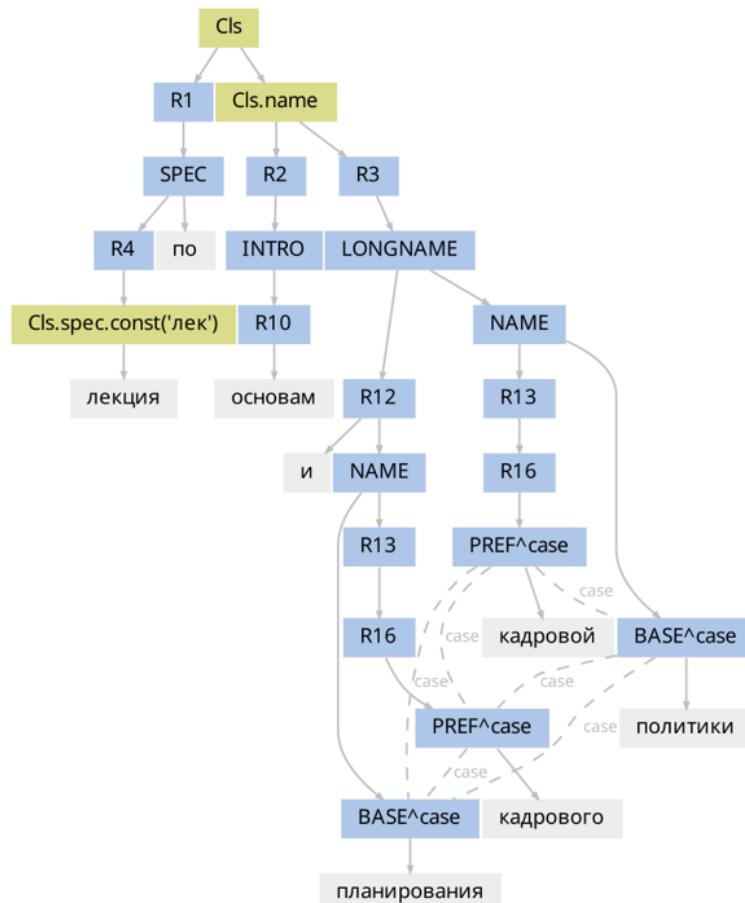


Рисунок 3.1 — Разбор строки «лекция по основам кадрового планирования и кадровой политики»

Здесь серым выделены терминалы, голубым – нетерминалы, желтым – метки для извлечения терминалов. Правила замены представлены стрелками, пунктирные case-стрелки указывают на согласование по падежу.

Проблемы, с которыми пришлось столкнуться в процессе написания грамматик с использованием Ярги-парсера:

- Не предусмотрено удаление стоп-слов, нужно наследоваться от токенизатора и реализовывать эту возможность самостоятельно.
- Если в правиле слова согласуются по падежу, и это правило исполь-

зуется несколько раз, то ограничить согласованность слов в пределах одного правила нельзя.

– Чем длиннее строку распознает грамматика, тем сложнее ее нормализовать. По этой причине нормализация осуществляется при помощи нечеткого поиска по словарю, о чем написано более подробно в подразделе 3.4.

После извлечения сущности она заменяется на общее слово, чтобы сократить количество уникальных слов на этапе классификации.

3.4 Приведение сущности к начальной форме

Русский язык имеет сложную морфологию. Это свойство приводит к возникновению проблемы сравнения слов и словосочетаний. Например, для нас очевидно, что словосочетания «чашечку кофе» и «чашка с кофе» обозначают один предмет, но компьютеру это понять сложно.

Пользователь может указывать именованные сущности в любой форме, кроме того они могут содержать орфографические ошибки и опечатки. Чтобы использовать эти сущности, например, для поиска в базе данных, нужно привести их к начальной форме. Это можно сделать путем нечеткого поиска по словарю.

Поиск осуществляется следующим образом. Из базы данных извлекаются все значения определенной именованной сущности. Затем эти значения сравниваются с тем, что указал пользователь. Наиболее близкое значение считается начальной формой. В данном проекте в качестве метрики близости используется расстояние Дамерау-Левенштейна. Для расчета расстояния используется библиотека *ruxDamerauLevenshtein*, которая обеспечивает наилучшую скорость работы.

3.5 Классификация фраз пользователей

В данном случае документом является фраза пользователя. Намерение — это класс документа, который позволяет определить, какой ответ нужно дать пользователю.

Для получения нормальной формы слова используется *pymorph2*. Для стемминга используется *nltk*.

Перед тем как перейти к классификации документа, необходимо получить его числовой вектор. Этот процесс называется векторизацией текста. Он может быть основан на определении веса каждого слова из словаря в документе. Словарь формируется из уникальных слов в корпусе. Таким образом, размер каждого числового вектора равен размеру словаря. Если в документе, который не учитывался при составлении словаря, находится новое слово, то его можно просто отбросить. Обычно, вектор оказывается слишком большим для дальнейших вычислений, поэтому его сжимают, например, с помощью метода главных компонент, что приводит к частичной потери информации. На рисунке 3.2 показан описываемый процесс векторизации.

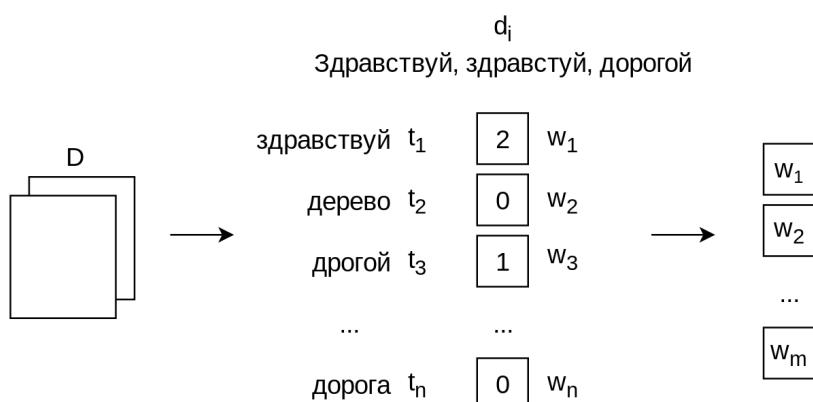


Рисунок 3.2 — Векторизация взвешиванием

В простейшем случае вес определяется как количество вхождений слова в документ. Чтобы убрать зависимость веса от размера документа, его можно разделить на количество слов в документе. Так будет получена TF-мера, которая выражает частоту соответствующего слова. Чтобы учесть специфику слов в документе, вес дополнительно умножают на логарифм отношения общего числа документов к числу документов, которые содержат определенное слово. Этот множитель называют IDF-мерой. Основание логарифма может быть любым, но чем оно меньше, тем сильнее будет увеличен вес специфичных слов. Таким образом, вес можно расчитать по формуле 3.3.

$$w_t = \frac{n_t}{n} \cdot \log \frac{|D|}{|\{d \mid t \in d\}|}, \quad (3.3)$$

где n_t — количество вхождений слова t в документ,
 n — общее количество слов в документе,
 D — множество документов d .

В больших документах некоторые слова имеют тенденцию повторяться чаще, поэтому колебание частоты слов сглаживают путем нормализации TF-меры. Методов нормализации существует множество, один из них описан уравнением 3.4.

$$ntf_{t,d} = a + (1 - a) \frac{tf_{t,d}}{tf_{\max}(d)}, \quad (3.4)$$

где a — сглаживающий коэффициент в диапазоне от 0 до 1,
 $tf_{t,d}$ — частота слова t в документе d .

Другой подход к векторизации основан на использовании Word2Vec модели. С ее помощью расчитывают вектор каждого слова из словаря. Затем, чтобы получить вектор документа, из словаря извлекают векторы всех слов и усредняют их покомпонентно. Если слово в документ не входит, то берется вектор, заполненный нулями, как показано на формулах 3.5.

$$V_d = \frac{\sum_{i=1}^n V(i)}{n},$$

$$V(i) = \begin{cases} V_i, & t_i \in d \\ V_0, & t_i \notin d \end{cases}, \quad (3.5)$$

где V — числовой вектор,
 V_0 — вектор, заполненный нулями.

Простейшая Word2Vec модель приведена на рисунке 3.3. Это нейронная сеть прямого распространения с одним скрытым слоем, размер которого определяет размер семантического вектора слова.

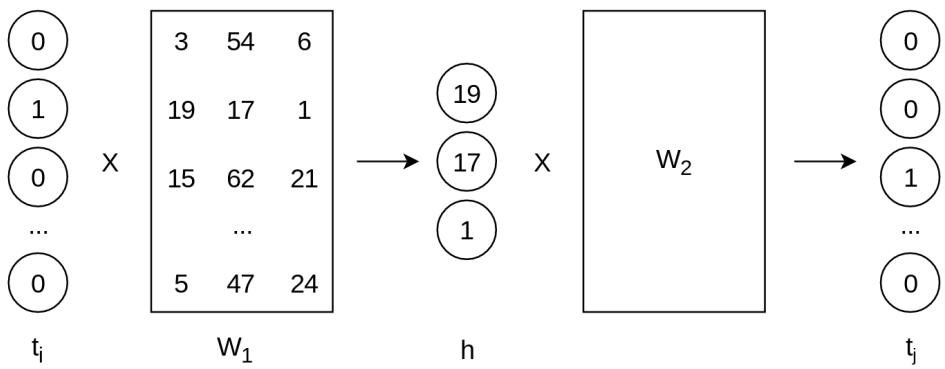


Рисунок 3.3 — Простейшая Word2Vec модель

На входе и выходе модели находятся унитарные коды слов. По этой причине каждое новое слово добавляет по одному нейрону на внешние слои. Слова для обучения выбираются из скользящего окна по всему корпусу. На каждой позиции окна выбирается целевое слово, а остальные считаются контекстными. Для приведенной модели используется окно шириной в 2 слова. На вход модели подается контекстное слово, на выходе образуется целевое. Скрытый слой имеет линейную функцию активации, а выходной — софтмакс (softmax). Модель обучается методом обратного распространения ошибки. После обучения строки входной матрицы весов W_1 будут семантическими векторами слов.

Модель Word2Vec состоит из двух моделей. Рассмотренная имеет название «Continuous Bag of Word» (CBOW), так как порядок слов в скользящем окне не имеет значения. Обычно, ширину окна выбирают больше двух, тогда на вход модели подается несколько контекстных слов, когда которых устредняются. Выход внутреннего слоя можно рассчитать по общей формуле 3.6.

$$h = \frac{t_1 + t_2 + \dots + t_c}{c} W_1. \quad (3.6)$$

где c - количество контекстных слов.

Другая модель называется «Skip-Gram» (SG). В ней, наоборот, на вход подается целевое слово, а на выходе образуются контекстные [8].

В данном проекте используется модель Doc2Vec, которая основана на Word2Vec [9]. Отличие заключается в том, что на вход модели кроме слов подается документ, который кодируется так же, как слово. На входе и выходе модели могут быть слова только из данного документа. После обучения модели нейроны документов отбрасываются. Для векторизации нового доку-

мента добавляется один нейрон, веса которого корректируются в процессе повторного обучения при фиксированной матрице весов слов. Такой подход называется замораживанием весов (weight freezing). Doc2Vec тоже состоит из двух моделей: «Distributed Memory» (PV-DM), которая соответствует CBOW, и «Distributed Bag of Words» (PV-DBOW), которая соответствует SG.

Подобранные гиперпараметры для векторизатора, который используется в проекте, приведены в таблице 3.1. Для реализации Doc2Vec модели использовалась библиотека Gensim.

Таблица 3.1 — Значения гиперпараметров векторизатора

| Гиперпараметр | Значение |
|--------------------------------|----------|
| Модель | PV-DBOW |
| Размер вектора | 100 |
| Слов в отрицательной выборке | 1 |
| Начальная скорость обучения | 0,1 |
| Размер окна | 7 |
| Минимальная частота слова | 1 |
| Понижение высокочастотных слов | 0 |

В данной работе используются модели, которые обучаются с учителем, поэтому им требуется размеченная обучающая выборка. Для каждого намерения имеется файл в формате JSON, в котором перечислены примеры пользовательских фраз. В каждом примере выделены именованные сущности. Структура файла представлена в листинге 3.3.

Листинг 3.3 — Структура обучающей выборки для намерения *nextClass*

```
1 [  
2   ...  
3   {  
4     "example": "Как попасть на пару к Иванову?",  
5     "entities" : [  
6       {  
7         "pos": [22, 29],  
8         "type": "empee",  
9         "value": {  
10           "name" : {"surn": "Иванов"}  
11         }  
12       }  
13     ],  
14     "comment": 2  
15   },  
16   ...  
17 ]
```

В каждом файле находится примерно 70 фраз. Визуализация обучающей выборки приведена на рисунке 3.4. Каждый объект имеет цвет того класса, к которому он принадлежит. Класс каждого цвета приведен в легенде. На рисунке видно, что объекты одного класса находятся рядом друг с другом, образуя кластеры.

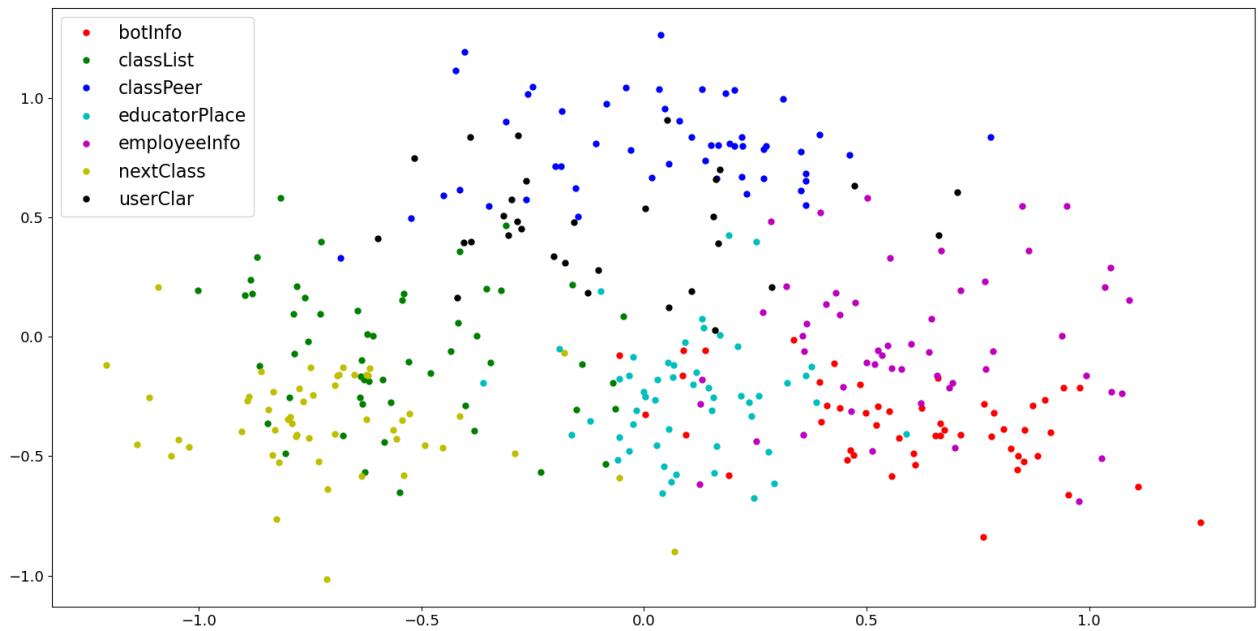


Рисунок 3.4 — Визуализация обучающей выборки

Для решения задачи классификации используется метод опорных векторов. Обоснование выбора данного метода приведено в разделе 4. Для реализации и тестирования классификаторов использовалась библиотека Scikit-learn.

4 Тестирование

Результаты интеграционного тестирования приведены в таблице 4.1. Для проверки правильности работы конкретных модулей системы использовались юнит-тесты.

Таблица 4.1 — Результаты интеграционного тестирования

| ИД | Вопрос | Ответ |
|------------------|---------------------------------|---|
| FUN_1 | Что ты умеешь? | Я могу рассказать вам, когда начинается ближайшее занятие, кто будет присутствовать на занятии, какие занятия будут проходить в интересующий вас день. Я знаю многих сотрудников, могу рассказать вам о ком-нибудь или подсказать, где найти интересующего вас человека. Чтобы вам было удобней общаться со мной, назовите свою группу, если вы учащийся, или имя, если вы сотрудник. Пропробуйте спросить прямо сейчас: «Когда пара?». |
| FUN_2 | Я учусь в группе 1162б | Группа 1162б. Запомнила! |
| FUN_3 | Когда пара? | В группе 1162б через 3 часа в корпусе 3, кабинете 445к Робототехника. |
| FUN_4 | Какие занятия завтра у Орлова? | Расписание у преподавателя Орлов С.А. на 20 июня отсутствует. |
| FUN_5 | Кто у меня ведет Робототехнику? | В группе 1162б дисциплину Робототехника ведет Долматов Алексей Викторович. |
| FUN_6 | Где находится Долматов? | У преподавателя Долматов А.В. сейчас перерыв. Вы можете поискать его в столовой или у себя, в корпусе 3, кабинете 145. |
| FUN_7, FUN_11 | Расскажи мне о нем | Вот, что мне известно о работнике Долматов Алексей Викторович: должность: доцент, отдел: Институт цифровой экономики, корпус: 3, кабинет: 145, почта: foo@ugrasu.ru. |

Продолжение таблицы 4.1

| ИД | Вопрос | Ответ |
|--------|-----------------------|--|
| FUN_8 | Какие новости? | Кажется, что-то интересное: «66 югорских старшеклассников прокачали soft skills в университете». |
| FUN_9 | Кабинет 210 свободен? | Пожалуйста, назовите корпус. |
| FUN_9 | Забудь | Ладно... Поговорим о чем-нибудь еще. |
| FUN_13 | – | Здравствуйте! Пожалуйста, назовите свою образовательную организацию, а также группу, если вы учащийся, или имя, если вы преподаватель. Можно спросить, что я умею. |

Нагрузочное тестирование проводилось с помощью утилиты Apache Benchmark. На сервер отправлялись 4 запроса одновременно. Когда сервер на них отвечал, к нему отправлялись еще 4, всего было 80 запросов. В теле запроса находилась копия данных, которые «ВКонтакте» отправляет на сервер для взаимодействия с ботом. В таблице 4.2 приведены результаты тестирования.

Таблица 4.2 — Результаты нагружочного тестирования

| Свойство | Значение |
|---|------------|
| Время тестирования | 17,219 с |
| Успешных запросов | 80 |
| Неудачных запросов | 0 |
| Всего получено | 13360 байт |
| Всего передано | 83360 байт |
| Запросов в секунду (в среднем) | 4,65 |
| Времени на один запрос | 215.237 мс |
| Времени на каждый запрос (в среднем) | 860,950 мс |
| Наибольшее количество времени на запрос | 1231 мс |
| Время соединения (в среднем) | 469 мс |
| Время обработки (в среднем) | 353 мс |
| Время ожидания (в среднем) | 349 мс |

Таким образом, требование PER_1 выполняется.

В листинге 4.1 приведены несколько примеров распознавания именованных сущностей.

Листинг 4.1 — Результат распознавания именованных сущностей

```
1 кто читает лекцию по современным технологиям разработки
  ↳ веб-приложений в группе 1381м?
2 кто читает class веб-приложений в группе group?
3 {'group': '1381м', 'class': {'name': 'современным технологиям
  ↳ разработки', 'spec': 'лек'}}
4
5 кто ведет основы кадровой политики в Югорском государственном
  ↳ университете в группе 1681?
6 кто ведет class в org в группе group?
7 {'org': 'Югорском государственном университете', 'group': '1681',
  ↳ 'class': {'name': 'основы кадровой политики', 'spec': None}}
8
9 кто придет на практику по правоведению завтра
10 кто придет на class day
11 {'class': {'name': 'правоведению', 'spec': 'пр'}, 'day': 1, 'org':
  ↳ 'Югорский государственный университет', 'group': '11626'}
```

Распознавание считается успешным, если границы сущности во фразе определены верно. Если сущности в предложении нет, то она не должна быть распознана. В качестве тестовой выборки использовались примеры фраз пользователя, о которых написано в подразделе 3.5. Из 100 предложений только в 4-х название дисциплины было распознано некорректно. Точность может существенно снизиться после добавления новых данных.

Оценка качества нескольких классификаторов проводилась по следующей методике. Данные разбивались на две части: обучающую и тестовую. На обучающей части подбиралась модель и ее гиперпараметры методом кросс-валидации по 5 блокам данных. Качество модели с оптимальными параметрами определялось на тестовых данных, с которыми ранее модель не встречалась. Метрикой качества являлась простая точность (accuracy).

В таблице 4.3 представлена метаинформация о данных, которые участвовали в оценке.

Таблица 4.3 — Метаинформация о данных

| Свойство | Значение |
|--------------------------------------|---|
| Время предобработки всего корпуса | 3,6497 с |
| Размер всего корпуса | 462 |
| Размер тестового корпуса | 116 |
| Время обучения векторизатора | 0,1308 с |
| Первый элемент обучающего корпуса | ['напомн', 'сво', 'возможн'], ['botInfo'] |
| Начало первого обучающего объекта | [-0.15747002 -0.02240924 0.04583127 ...] |
| Первая обучающая метка | botInfo |
| Первая обуч. метка после кодирования | [1. 0. 0. 0. 0. 0. 0.] |

Для всех моделей использовался один предварительно обученный векторизатор, данные для обучения и тестирования между моделями не отличались. Результаты оценки приведены в таблице 4.4

Таблица 4.4 — Результаты тестирования

| Классификатор | Средняя точность на кросс-валидации | Точность на тестовой выборке | Оптимальные значения гиперпараметров |
|---------------------|-------------------------------------|------------------------------|--|
| Ближайших соседей | 0,8989 | 0,7328 | Метрика: евклидова, соседей: 7 |
| Случайный лес | 0,8613 | 0,7155 | Деревьев: 150, макс. глубина: 5, макс. признаков: 85 |
| Опорные векторы | 0,9191 | 0,8017 | Ядро: линейное |
| Наивный байесовский | 0,9076 | 0,7931 | — |

Оптимизация гиперпараметров проводилась с помощью поиска по решетке. На рисунках 4.1 и 4.2 показана зависимость точности от значений основных гиперпараметров для нескольких моделей.

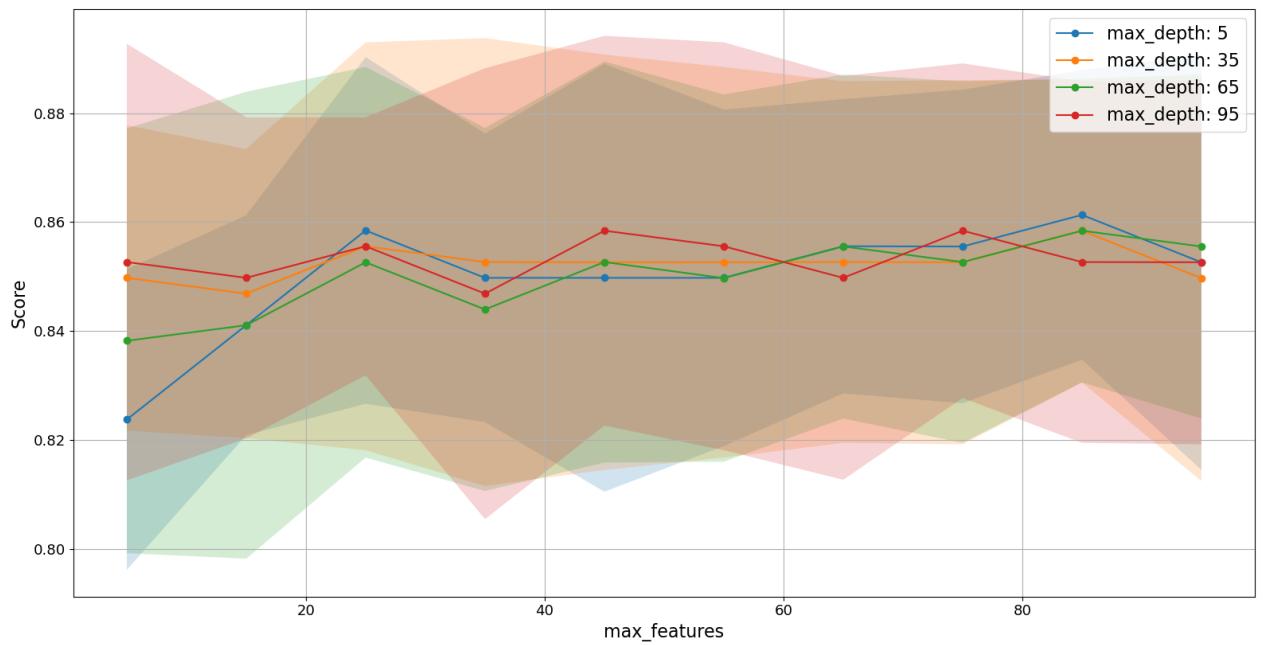


Рисунок 4.1 — График изменения точности для случайного леса

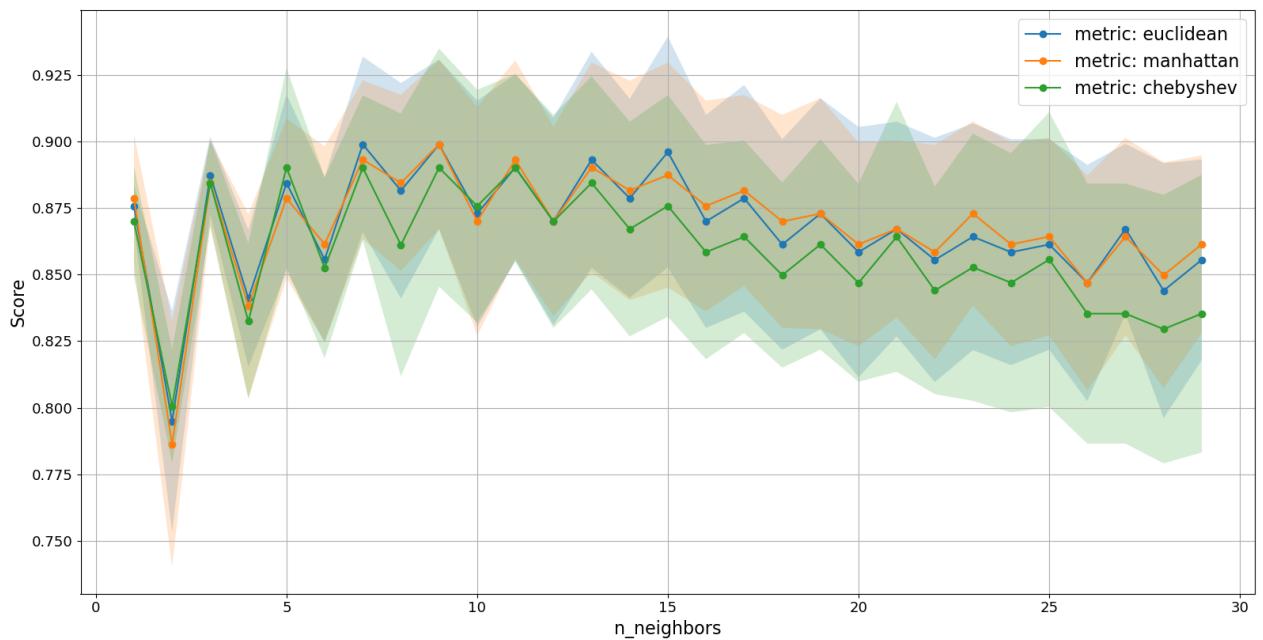


Рисунок 4.2 — График изменения точности для ближайших соседей

ЗАКЛЮЧЕНИЕ

На данный момент сервис полностью соответствует описанным требованиям. Поставленная цель достигнута. Проведена оценка сервиса, проектирование, реализация и тестирование. Собственное клиентское приложение, которое позволит посмотреть расписание занятий и свой дневник, находится в разработке.

Бот получил название «Югорский мудрец». Он прошел модерацию на платформе «Яндекс Диалоги» и доступен в приложении «Яндекс Алиса» в виде навыка. Пример обращения: «Алиса, спроси Юрского мудреца, где пара?». Демонстрация работы приведена в приложении Г.

Бот доступен во «ВКонтакте». Чтобы начать с ним диалог, нужно зайти в одноименное сообщество и нажать на кнопку «Написать сообщение». Демонстрация приведена в приложении Д.

Бот доступен в «Телеграм». Найти его можно по логину *@UgraSageBot*. Демонстрация приведена в приложении Е.

Данный сервис освещался на нескольких мероприятиях. Подтверждающие сертификаты и дипломы приведены в приложениях Ж, З, И, К. Упрощенная версия бота находится в открытом доступе в репозитории на GitHub [10].

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- 1 Какими соцсетями и мессенджерами пользуются мужчины и женщины, молодые и пожилые, в городах и сёлах [Электронный ресурс] : Пользование социальными сетями и мессенджерами / Фонд «Общественное мнение». — 2018. — Режим доступа: <https://fom.ru/Obraz-zhizni/14137>.
- 2 *Riepe J.* Why chatbots are not the future of student engagement [Electronic resource] : When to take advantage of AI and when to not. — 2018. — Mode of access: <https://www.ecampusnews.com/2018/02/06/chatbots/?all>.
- 3 *Mathew R. G.* Function Point Analysis [Electronic resource]. — 2017. — Mode of access: <https://sourceforge.net/p/functionpoints/wiki/Function%20Point%20Analysis/?version=35>.
- 4 *Jones C.* Software Economics and Function Point Metrics : Thirty years of IFPUG Progress. — 2017.
- 5 COCOMO II : Model Definition Manual / Center for Software Engineering, USC. — Version 2.0. — 2000.
- 6 Russian NER (with inflectional gazetteer and orthomatcher) [Electronic resource] / The University of Sheffield. — Mode of access: <https://cloud.gate.ac.uk/shopfront/displayItem/russian-ner-with-inflexional-gazetteer-and-orthomatcher>.
- 7 *Kukushkin A.* Yargsy [Electronic resource] : Getting started. — Mode of access: <https://nbviewer.jupyter.org/github/natasha/yargsy/blob/master/docs/index.ipynb>.
- 8 *Rong X.* word2vec Parameter Learning Explained [Electronic resource] // CoRR. — 2014. — Mode of access: <http://arxiv.org/abs/1411.2738>.
- 9 *Le Q. V., Mikolov T.* Distributed Representations of Sentences and Documents [Electronic resource]. — 2014. — Mode of access: https://cs.stanford.edu/~quocle/paragraph_vector.pdf.
- 10 *Panchishin I.* UgraSageDemo [Electronic resource] : Information bot for Ugra State University. — Mode of access: <https://github.com/vpunch/UgraSageDemo>.

ПРИЛОЖЕНИЕ А

Требования к сервису

Функциональные требования приведены в подразделе 1.4.

Требования удобства использования определяют, на сколько легко пользователь сможет взаимодействовать с сервисом. Они содержатся в таблице А.1.

Таблица А.1 — Удобство использования

| ИД | Описание |
|-------|---|
| USA_1 | Бот позволяет пользователю при обращении не использовать шаблонные фразы. |
| USA_2 | Бот предоставляет пользователю варианты ответа, когда это уместно. |
| USA_3 | В сервисе имеется краткая информация о том, что умеет бот. |
| USA_4 | Пользователю не требуется обучение для использования бота. |
| USA_5 | Бот позволяет пользователю использовать как текстовый, так и голосовой интерфейс для общения. |
| USA_6 | В своих ответах бот оставляет ссылки на внутренние ресурсы ОО, когда это уместно. |
| USA_7 | Информация, предоставляемая ботом, должна быть краткой и достаточной. |

Требования надежности приведены в таблице А.2. Они определяют точность вычислений, время работы и способность к восстановлению после сбоя.

Таблица А.2 — Надежность

| ИД | Описание |
|-------|--|
| REL_1 | Сервис должен быть доступен 24 часа в сутки, 7 дней в неделю. Допускаются небольшие перерывы. |
| REL_2 | Имеется возможность полностью восстановить работу сервиса после сбоя. |
| REL_3 | Точность предоставленной информации зависит от точности исходных данных, предоставленных ОО. |
| REL_4 | Бот игнорирует сообщения, которые приходят от одного пользователя чаще, чем раз в 2 секунды. |

Требования к производительности содержатся в таблице А.3. Они определяют скорость работы сервиса и пропускную способность.

Таблица А.3 — Производительность

| ИД | Описание |
|-------|---|
| PER_1 | На генерацию ответа пользователю у бота не должно уходить больше 2 секунд. Требование может не соблюдаться, если нагрузка на бота превышает допустимую. Нагрузка на бота превышает допустимую, когда он обрабатывает больше 4 запросов в один момент времени. |
| PER_2 | Сервис не использует больше 3 ГиБ памяти в пределах допустимой нагрузки. |

Требования поддержки содержатся в таблице А.4. Они определяют совместимость, масштабируемость, возможность установки и т.д.

Таблица А.4 — Поддерживаемость

| ИД | Описание |
|-------|--|
| SUP_1 | Взаимодействовать с ботом можно с помощью «ВКонтакте», «Telegram», «Яндекс.Алиса». |
| SUP_2 | Сервис должен поддерживать формат данных обслуживаемой ОО. |
| SUP_3 | Сервис должен обслуживать ЮГУ. |
| SUP_4 | Установка сервиса осуществляется с помощью набора сценариев. |

Ограничения проектирования содержатся в таблице А.5. Они позволяют указать на использование определенных архитектурных шаблонов.

Таблица А.5 — Ограничения проектирования

| ИД | Описание |
|-------|---|
| DES_1 | Внутри сервиса не должно быть зависимостей от какой-либо ОО или какого-либо клиентского приложения. |

Ограничения реализации содержатся в таблице А.6. Они касаются этапа реализации системы и позволяют определить конкретный язык или инструменты для реализации.

Таблица А.6 — Ограничения реализации

| ИД | Описание |
|-------|----------|
| IMP_1 | - |

Требования к интерфейсам содержатся в таблице А.7. Они определяют взаимодействие сервиса с внешним окружением.

Таблица А.7 — Требования к интерфейсам

| ИД | Описание |
|-------|--|
| INT_1 | Сервис должен использовать HTTPS для обмена данными. |
| INT_2 | Сервис периодически получает исходные данные от ОО, используя для этого специальный интерфейс. |
| INT_3 | Сервис должен иметь независимые интерфейсы для каждого поддерживаемого клиента. |
| INT_4 | Сервис может предоставлять данные клиентскому приложению. |

Физические ограничения содержатся в таблице А.8. Они определяют физические характеристики оборудования, на котором размещается система.

Таблица А.8 — Физические ограничения

| ИД | Описание |
|-------|----------|
| HAR_1 | - |

ПРИЛОЖЕНИЕ Б

Конфигурация Vim

Листинг Б.1 — Содержание конфигурационного файла Vim

```
1 "https://github.com/junegunn/vim-plug
2
3 if empty(glob('~/vim/autoload/plug.vim'))
4     silent !curl -fLo ~/.vim/autoload/plug.vim --create-dirs
5         \ https://raw.githubusercontent.com/junegunn/
6             \ vim-plug/master/plug.vim
7     autocmd VimEnter * PlugInstall --sync | source $MYVIMRC
8 endif
9
10 call plug#begin('~/vim/plugged')
11 Plug 'morhetz/gruvbox'
12 Plug 'easymotion/vim-easymotion'
13 Plug 'lervag/vimtex'
14 Plug 'sirver/ultisnips'
15 Plug 'honza/vim-snippets'
16 Plug 'suan/vim-instant-markdown', {'for': 'markdown'}
17 Plug 'KeitaNakamura/tex-conceal.vim'
18 call plug#end()
19
20 let g:tex_flavor='latex'
21 let g:vimtex_view_method='zathura'
22 let g:vimtex_quickfix_mode=0
23 let g:vimtex_compiler_latexmk = {
24     \ 'options' : [
25         \ '-pdf',
26         \ '-shell-escape',
27         \ '-verbose',
28         \ '-file-line-error',
29         \ '-synctex=1',
30         \ '-interaction=nonstopmode',
31         \ ],
32     \}
33 autocmd InsertLeave *.tex update
34
35 set concealevel=2
36 let g:tex_conceal="abdgm"
37
38 let g:UltiSnipsExpandTrigger='<tab>'
39 let g:UltiSnipsJumpForwardTrigger='<tab>'
40 let g:UltiSnipsJumpBackwardTrigger='<s-tab>'
41 let g:UltiSnipsSnippetDirectories=['UltiSnips']
```

Листинг Б.2 — Продолжение листинга Б.1

```
1 colorscheme gruvbox
2 set background=dark
3
4 let mapleader=', '
5 map <Leader> <Plug>(easymotion-prefix)
6
7 filetype plugin on
8 let g:instant_markdown_logfile = '/tmp/instant_markdown.log'
9 let g:instant_markdown_automscroll = 0
10
11 "Размер табуляции
12 set tabstop=4
13 "Размер отступа, который формируется из табуляций с пробелами
14 set softtabstop=4
15 "Отступ при нажатии >>
16 set shiftwidth=4
17 "Использовать больше цветов в терминале
18 set t_Co=256
19 "Нумерация строк
20 set number
21 "Подсвечивать вхождения при поиске
22 set hlsearch
23 "Подсвечивать первое вхождение во время ввода при поиске
24 set incsearch
25 "Подсвечивать синтаксис
26 syntax on
27 "Заменить табуляции на пробелы
28 "retab
29 set expandtab
30 "Ограничение строки
31 set colorcolumn=72,79
```

ПРИЛОЖЕНИЕ В

Распознающая грамматика для названия занятия

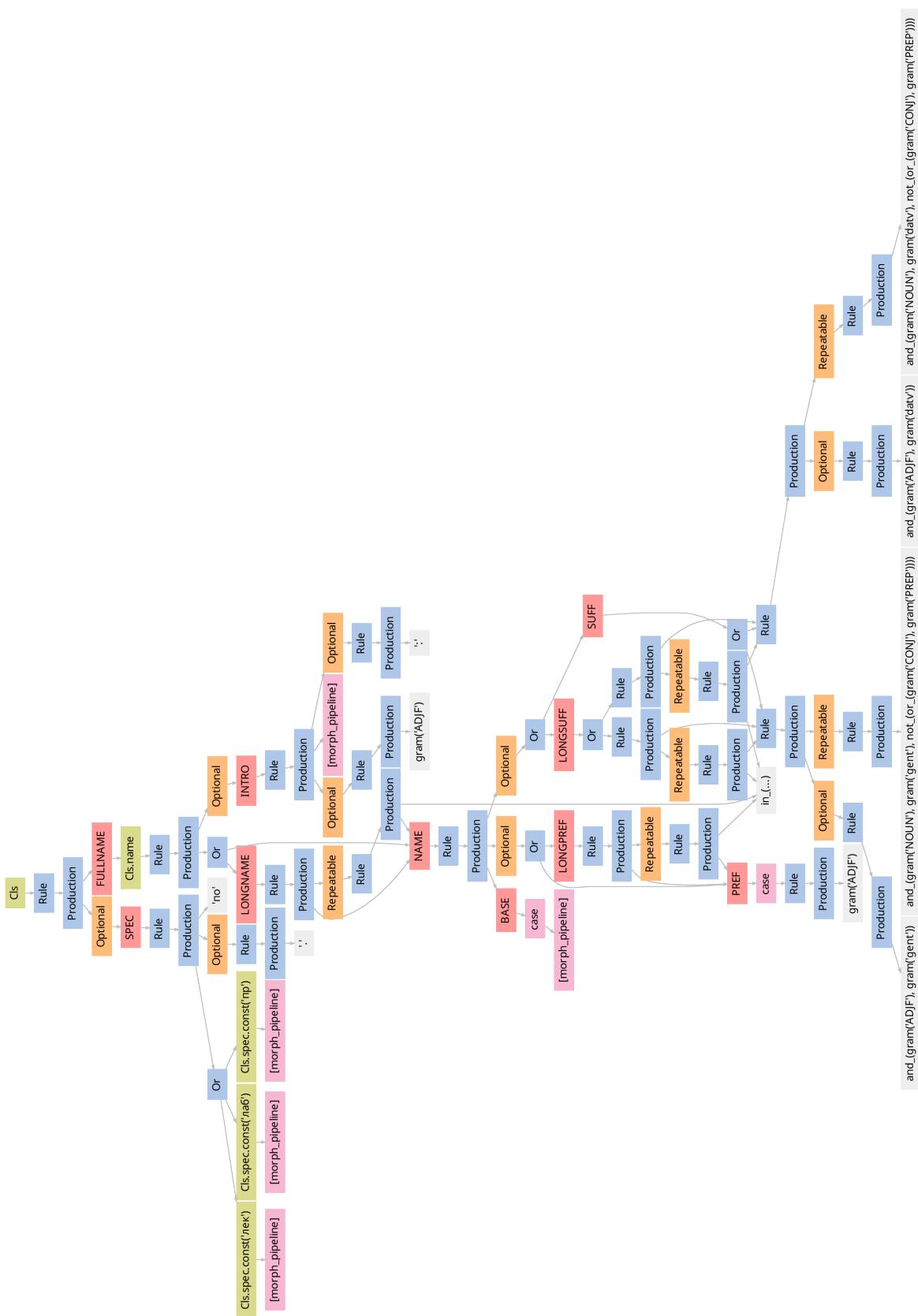


Рисунок B.1

ПРИЛОЖЕНИЕ Г

Бот в «Яндекс Алиса»

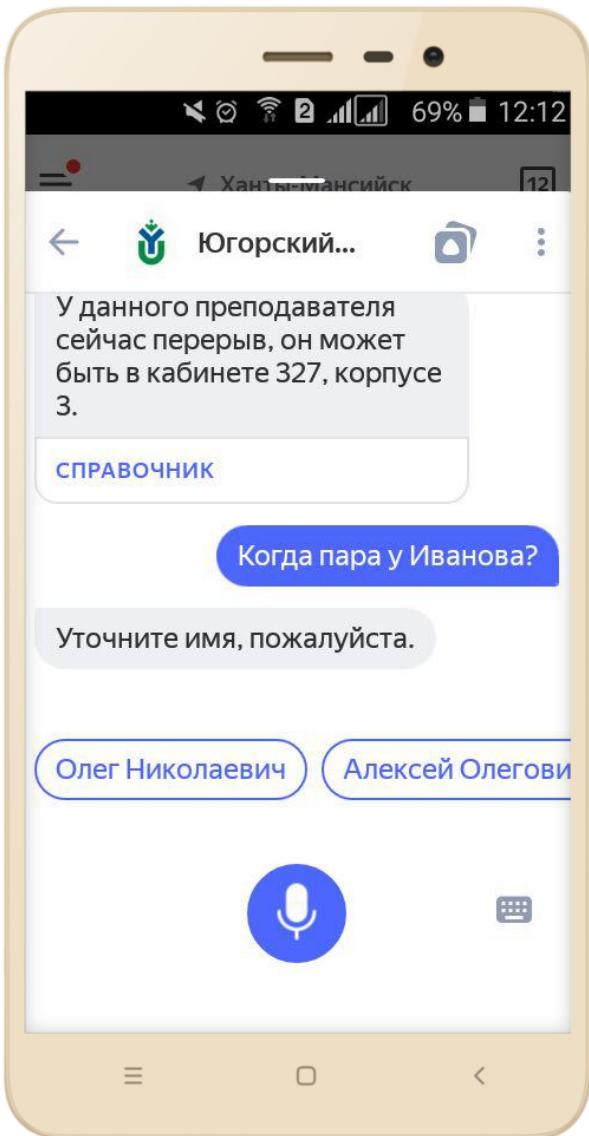


Рисунок Г.1

ПРИЛОЖЕНИЕ Д

Бот во «ВКонтакте»

< Назад Югорский мудрец ⚡ ... 

 Югорский мудрец 22:02
Я знаю несколько человек с таким именем. Пожалуйста, назовите полное имя.

 Иван 22:02
Орлов Алексей Владимирович

 Югорский мудрец 22:02
У преподавателя Орлов А.В. через 2 дня в корпусе 3, кабинете 246 Физика

 Иван 22:02
когда он будет вести занятие в группе 11626?

 Югорский мудрец 22:02
В группе 11626, у преподавателя Орлов А.В. неизвестно, когда начинается занятие

Рисунок Д.1

ПРИЛОЖЕНИЕ Е

Бот в «Телеграм»

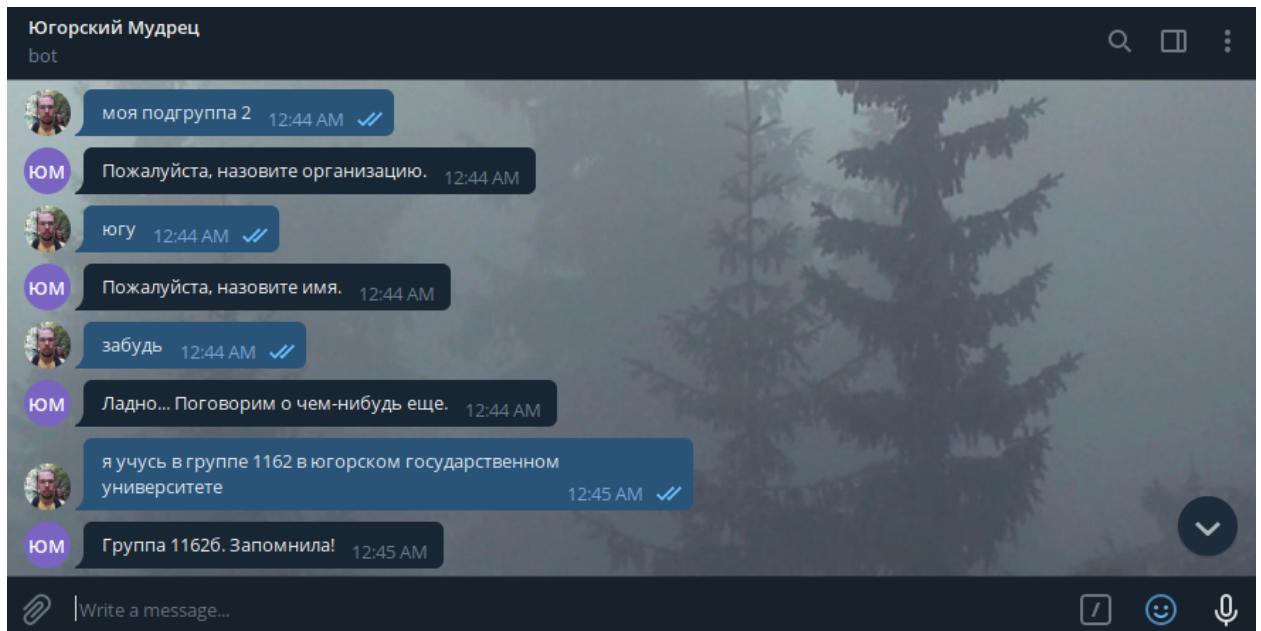


Рисунок Е.1

ПРИЛОЖЕНИЕ Ж
Диплом победителя конкурса докладов КМУ 2020



Рисунок Ж.1

ПРИЛОЖЕНИЕ З
Диплом участника «Славим человека труда!» 2019



КОНКУРС ПРОФЕССИОНАЛЬНОГО МАСТЕРСТВА «СЛАВИМ ЧЕЛОВЕКА ТРУДА»
УРАЛЬСКОГО ФЕДЕРАЛЬНОГО ОКРУГА
ПРИ ПОДДЕРЖКЕ ПОЛНОМОЧНОГО ПРЕДСТАВИТЕЛЯ
ПРЕЗИДЕНТА РОССИЙСКОЙ ФЕДЕРАЦИИ В УРАЛЬСКОМ ФЕДЕРАЛЬНОМ ОКРУГЕ



ДИПЛОМ

НАГРАЖДАЕТСЯ

**ПАНЧИШИН
ИВАН
РОМАНОВИЧ**

УЧАСТИК КОНКУРСА ПРОФЕССИОНАЛЬНОГО
МАСТЕРСТВА «СЛАВИМ ЧЕЛОВЕКА ТРУДА!»
УРАЛЬСКОГО ФЕДЕРАЛЬНОГО ОКРУГА
В НОМИНАЦИИ

**«ЛУЧШИЙ ИНЖЕНЕР-ПРОГРАММИСТ»
В КАТЕГОРИИ
«ИНЖЕНЕРНОЕ ИСКУССТВО МОЛОДЫХ»**

ГУБЕРНАТОР
СВЕРДЛОВСКОЙ ОБЛАСТИ



Е. В. КУЙВАШЕВ

Рисунок 3.1

ПРИЛОЖЕНИЕ И
Диплом призера «IT4U» 2019



МИНИСТЕРСТВО НАУКИ
и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное
учреждение высшего образования
«ЮГОРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

ДИПЛОМ
III степени

*За успешное выступление
в VIII Молодежной научно-практической конференции
«Информационные технологии Югры»*

Награждается

Панишин Иван Романович

*Тема выступления на конференции
Виртуальный помощник
Ильи Ягорского государственного
университета*

Научный руководитель

Шишлов Анатолий Валерьевич

Проректор по учебной работе



P.B. Кучин

г. Ханты-Мансийск



Рисунок И.1

ПРИЛОЖЕНИЕ К
Сертификат участия «Рост UP» 2019



Сертификат участия

Награждается проект

Виртуальный помощник для ЮГУ

Автор проекта:

Панчишин И.Р., студент ЮГУ



A blue ink signature of the name "B.M. Рулевский" (B.M. Rulovskiy).

B.M. Рулевский



Открытая выставка научных достижений молодых ученых | 2019

Рисунок К.1