

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»

Кафедра Дискретной математики и информационных технологий

РАЗРАБОТКА ЧАТ-БОТА ДЛЯ СЕРВИСА ОБМЕНА СООБЩЕНИЯМИ
TELEGRAM С ИСПОЛЬЗОВАНИЕМ BOT API

КУРСОВАЯ РАБОТА

студента 2 курса 221 группы
направления 09.03.01 — Информатика и вычислительная техника
факультета КНиИТ
Давиденко Алексея Алексеевича

Научный руководитель

Ассистент кафедры ДМиИТ

А.А. Трунов

Заведующий кафедрой

к. ф.-м.н., доцент

Л.Б. Тяпаев

Саратов 2019

СОДЕРЖАНИЕ

ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ	3
ОПРЕДЕЛЕНИЯ	4
ВВЕДЕНИЕ	5
1 Теоретическая часть	6
1.1 Выбранный язык разработки backend	7
1.2 Работа с Telegram Bot API	8
1.3 Сцены	9
2 Практическая часть	12
2.1 Получение списка групп с сайта СГУ	12
2.2 Получение интересующей пользователя группы	14
2.3 Получение расписания на определённый день недели	15
2.4 Получение расписания преподавателя	16
2.5 Сцена получения расписания преподавателя	17
2.6 Главная сцена	17
2.7 Тестирование работы бота	17
ЗАКЛЮЧЕНИЕ	31
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	32
Приложение А Примеры программ	33
Приложение Б Структура списка групп с сайта СГУ	36
Приложение В Метод получения интересующей пользователя группы ...	46
Приложение Г Метод получения расписания на определённый день неде- ли	51
Приложение Д Метод получения расписания преподавателя	54
Приложение Е Реализация сцены получения расписания преподавателя .	55
Приложение Ж Реализация главной сцены	59

ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

Bot API — Telegram Bot API;

ЯП — язык программирования;

JS — JavaScript;

ОПРЕДЕЛЕНИЯ

Мессенджер — веб-сервис мгновенного обмена сообщениями;

Frontend — это то, что видит пользователь. То что отображается браузером, или в пользовательской части приложения, визуальная часть приложения;

Backend — это то, что скрыто от пользователя. То, что обрабатывается на сервере, логическая часть приложения;

REST — Representational State Transfer, стиль взаимодействия компонентов приложения в сети;

API — Application Programming Interface, описание способов взаимодействия программ;

Веб-скрапинг — сбор данных с различных интернет ресурсов;

Парсер — синтаксический анализатор, преобразующий входные данные в структурированный формат;

Middleware — функции промежуточной обработки, т.е. функции, имеющие доступ к объекту запроса, объекту ответа и к следующей функции промежуточной обработки в цикле “запрос-ответ”.

ВВЕДЕНИЕ

В настоящее время большинство людей для удалённого общения друг с другом используют различные мессенджеры. Используются они по ряду причин:

- Удобство использования;
- Бесплатность использования;
- Высокие скорости интернет-соединения;
- Распространённость смартфонов;
- Возможность создания групповых бесед;
- Наличие разнообразных полезных интерактивных помощников;
- И другие

Целью курсовой работы является разработка интерактивного помощника для мессенджера Telegram. Разработка подобных систем является очень актуальной в наше время. Их использование позволяет пользователям получать необходимую информацию более быстрым, простым и удобным способом, а для разработчиков это одна из возможностей представления информации вне сайта организации или приложения. Для достижения цели необходимо решить следующие задачи:

- провести анализ в предметной области;
- выбрать платформу для создания и размещения функциональных интерактивных помощников;
- разработать выбранный мобильный помощник с использованием выбранной платформы.

1 Теоретическая часть

Чат-боты — это некий интерактивный онлайн-помощник, который общается с пользователями посредством сообщений и обладает множеством функций. Чат-бота можно использовать для получения свежих новостей, делать массовые рассылки, управлять беседой, сервером и так далее.

Поддержкой чат-ботов обладают все ведущие мессенджеры, такие как Facebook Messenger, ВКонтакте, Telegram, WhatsApp, Viber и другие. При выборе платформы я поставил несколько критериев для выбора:

- Распространённость
- Приватность пользователя
- Скорость ответа платформы на сервер
- Наличие подробной документации

Мой выбор пал на Telegram как на одну из самых перспективных платформ.

Чат-ботов можно создавать несколькими способами — используя конструктор, который даёт сама платформа, но тогда бот будет способен совершать ограниченное число функций, например, отвечать на сообщение тем же сообщением, либо разработать с помощью одного из языков серверного программирования: Python, PHP, Node.JS, Ruby, Perl, Java.

Во втором случае разработка заключается в написании backend, поскольку frontend предоставляется Telegram.

Для написания бота необходимо знать хотя бы один из языков серверного программирования, для написания backend; уметь работать с REST API, который предоставляет Telegram, Telegram Bot API [2]; определиться с типом бота:

1. Самообучающийся бот, использующий машинное обучение для обработки сообщений и построения диалога с пользователем на естественном языке;
2. Скриптовый бот, использующий для общения с пользователем использует заранее описанный разработчиком шаблон, в котором с помощью дерева возможных решений определяется ответ бота пользователю. Диалог представляет собой линейный, структурированный сценарий и определиться с задачей, которую будет решать бот.

1.1 Выбранный язык разработки backend

Для разработки backend-а мной был выбран Node.JS, как наиболее близкий и удобный для меня язык программирования.

Node.JS — это среда выполнения JavaScript, которая выполняет JavaScript вне браузера [3]. Говоря простым языком, если программист знает JavaScript и может написать с помощью него клиентскую часть приложения, то ему не нужно изучать дополнительный язык программирования, для написания серверной части.

Благодаря относительно низкому порогу вхождения, наличия огромного числа учебников, методических материалов, онлайн курсов, JavaScript имеет очень большую популярность среди разработчиков, и благодаря этому так же очень популярен Node.JS. Код, написанный на JS, лаконичен и понятен даже тому, кто никогда не писал на этом ЯП. За счёт простоты кода, дальнейшее сопровождение программ намного легче, чем, например, программ, написанных на Java или C++ [6].

Так же одним из достоинств языка является наличие асинхронного ввода-вывода, а именно модель с неблокирующими операциями ввода-вывода. Цикл событий, лежащий в основе этой модели, позволяет системе работать более эффективно, приложение может выполнять какие-либо функции, пока ждёт завершения других операций ввода-вывода. Это так же делает программы, написанные на Node.JS, лёгкими и эффективными.

Благодаря большому сообществу Node.JS разработчиков, были написаны более 350000 подключаемых к программе модулей, обеспечивающих различные дополнительные возможности: работа с файловой системой, работа с базами данных, веб-скрапинг, логирование и другие полезные функции [2].

К недостаткам же можно отнести малую эффективность в операциях, интенсивно использующих CPU. К таким операциям можно отнести, например, обработку изображений и генерацию графики. Однако когда требуется выполнение этих задач, обычно разработчик будет сразу думать не про Node.JS, а про, например, Python, который решает эти проблемы более эффективно. Но этот недостаток с лихвой окупается достоинствами языка при написании веб-приложений, в которых ценится устойчивость к отказам и поддержка большого числа одновременных подключений.

1.2 Работа с Telegram Bot API

Для работы с Telegram Bot API, сообществом было написано большое количество модулей, самые популярные из которых это `node-telegram-bot-api` и `Telegraf`. В своём проекте я использовал `Telegraf`, так как он, в отличие от первого, использует последнюю версию Telegram BOT API, имеет более удобные функции и имеет поддержку чистых функций Telegram Bot API [1].

Бот, создаваемый модулем `Telegraf`, представляет собой объект, содержащий массив `middlewares`, которые по запросу составляются и выполняются из стека.

Middleware позволяет изменять запросы и ответы при их передаче между Telegram Bot API и ботом. Его можно представить как цепочку логических связей между ботом и Telegram Bot API.

С помощью `middlewares` можно между запросом и ответом, например, определять язык, который использует пользователь, и в зависимости от него отвечать на нужном языке, или запустить расширенный логгинг бота.

Middleware обычно имеет два параметра: `ctx` - контекст для одного запроса или ответа Telegram, и `next` - функция, которая выполнется после выполнения `ctx`.

Контекст `ctx` содержит в себе ответ Telegram Bot API. Контекст создаётся для каждого запроса и содержит свойства, представленные в таблице 1.1 [1]

Таблица 1.1 – Используемые свойства

<code>ctx.telegram</code>	Экземпляр клиента Telegram
<code>ctx.updateType</code>	Изменение типа (<code>message</code> , <code>inline_query</code> , и др.)
<code>[ctx.updateSubTypes]</code>	Изменение подтипов (<code>text</code> , <code>sticker</code> , <code>audio</code> , etc.)
<code>[ctx.message]</code>	Полученное сообщение
<code>[ctx.editedMessage]</code>	Изменённое сообщение
<code>[ctx.inlineQuery]</code>	Полученный встроенный запрос
<code>[ctx.chosenInlineResult]</code>	Полученный результат встроенного запроса
<code>[ctx.callbackQuery]</code>	Полученный обратный запрос
<code>[ctx.shippingQuery]</code>	Запрос доставки
<code>[ctx.preCheckoutQuery]</code>	Запрос предварительной проверки
<code>[ctx.channelPost]</code>	Новая входящая запись на канале любого типа - <code>text</code> , <code>photo</code> , <code>sticker</code> , и др.
<code>[ctx.editedChannelPost]</code>	Изменённое сообщение на канале, которое до этого было известно боту
<code>[ctx.poll]</code>	Изменение в анонимном опросе
<code>[ctx.chat]</code>	Информация о текущем чате
<code>[ctx.from]</code>	Информация об отправителе
<code>[ctx.match]</code>	Соответствие в регулярном выражении
<code>ctx.webhookReply</code>	Ответ на произошедшее событие

Так же ctx содержит оболочку для функций Telegram Bot API, предложенных на Официальной странице Bot API [2], расположенной на <https://core.telegram.org/bots/api> [1]. Некоторые оболочки представлены в таблице 1.2

Таблица 1.2 – Некоторые оболочки Telegraf

Оболочка	Функция Telegram Bot API	Описание
deleteMessage	deleteMessage	Удалить сообщение
forwardMessage	forwardMessage	Ответить на выбранное сообщение
getChat	getChat	Получить свежую информацию о беседе
reply	sendMessage	Отправить сообщение
replyWithHTML	sendMessage	Отправить сообщение, содержащее HTML
replyWithLocation	sendLocation	Отправить местоположение на карте

Поддерживаемые Telegraf типы обновлений [1]

- message
- edited_message
- callback_query
- inline_query
- shipping_query
- pre_checkout_query
- chosen_inline_result
- channel_post
- edited_channel_post

Таким образом, все типы, определённые в Telegraf, соответствуют определениям типов Bot API.

Пример использования Telegraf находится в приложении А на странице 33, Листинг 3

1.3 Сцены

Для того чтобы построить диалог пользователя с ботом можно использовать множество if...else, но это влечёт за собой множество проблем, таких как смешивание контекста, высокая сложность структуры диалога, непредсказуемость работы программы при обращении к предыдущему шагу диалога, т. е. когда, например, пользователь нажал кнопку “назад”. Для того чтобы не было непредсказуемых участков, переменные имели правильные значения, чтобы была более простая структура диалога используются сцены — Scene и

WizardScene. Они позволяют объединять несколько функций в один сценарий [4, 6].

Например, в у бота есть несколько функций — покупка билетов на самолёт и показ анекдотов. Чтобы купить билеты, нужно сделать много шагов: ввести все свои данные, указать пункты отправки и назначения и так далее, а для просмотра анекдотов нужно просто нажать одну кнопку. Чтобы их разграничить Удобнее всего будет использовать сцены.

Допустим, у нашего бота имеются 2 кнопки: “Покупка билетов” и “Анекдоты”. Чтобы пользователь смог купить билеты, он должен нажать на кнопку “Покупка билетов”. И при нажатии на эту кнопку бот переводит пользователя в сцену покупки билетов. В ней есть несколько шагов, например, такие:

1. Ввод фамилии
2. Ввод имени
3. Ввод отчества
4. Ввод номера паспорта
5. Ввод города отправления
6. Ввод города назначения
7. Ввод дня отправления
8. Показ возможных вариантов полёта
9. Выбор рейса
10. Переход на сайт авиакомпании и покупка билета по заданным критериям

При нажатии пользователем кнопки “Покупка билетов”, пользователь переходит в сцену покупки билетов и ему предлагается ввести фамилию. При отправке пользователем фамилии, бот сначала сохранит отправленное пользователем сообщение в переменную, отвечающую за фамилию, например, `lastName`. Выглядеть эта переменная будет следующим образом:

`ctx.wizard.state.lastName`, где

- `ctx` — это контекст;
- `wizard` — свойство контекста, отвечающее за мастера сцен, `WizardScene`;
- `state` — менеджер свойств сцены, в котором хранятся значения нужных нам переменных. Сохраняются они только внутри сцены;
- `lastName` — свойство, отвечающее за фамилию потенциального покупателя

Затем бот должен проверить оценить валидность введенных данных, не является ли введенная “фамилия” простым набором букв и знаков. Это можно выполнить с помощью регулярных выражений в `middlewares`. Если введенные данные валидны, то производится переход к следующему шагу сцены с помощью команды `return ctx.wizard.next()`. `next` - следующий описанный шаг в сцене. Иначе, если данные не валидны, то можно перейти к предыдущей сцене с помощью команды `return ctx.wizard.back()`, либо к любому другому шагу в сцене с помощью команды `return ctx.wizard.selectStep(i)`, где `i` - нужный нам шаг [1]. И так далее для каждого шага. На последнем этапе, когда пользователь уже купил билеты, мы должны выйти из сцены и перейти в сцену по-умолчанию, в которой бот предлагает выбрать одну из кнопок. Это можно сделать с помощью `return ctx.scene.enter('default')`, где `default` - сцена по умолчанию, и в сцене по умолчанию выйти из сцены покупки с помощью команды `leave('buy')`, где `buy` - сцена покупки [4].

Пример сцены представлен в приложении А на странице 33, Листинг 4

2 Практическая часть

В качестве выполняемой чат—ботом задачи был выбран вывод расписания СГУ.

Была спроектирована схема взаимодействия пользователя с базой данных, которая определяет, как сервер должен взаимодействовать с Telegram Bot API и сайтом расписания СГУ, расположенном по адресу <https://www.sgu.ru/schedule>.

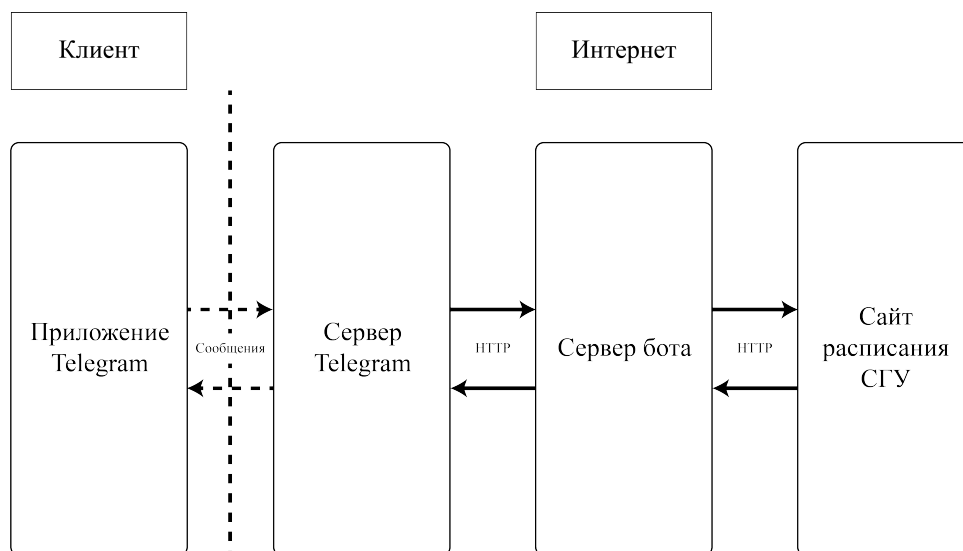


Рисунок 2.1 – Схема взаимодействия пользователя с сайтом расписания СГУ

Для взаимодействия с Telegram Bot API сначала нужно получить ключи API. Ключ API — это секретный код, который идентифицирует определённую учетную запись и позволяет использовать методы, определяемые API. В случае Bot API, ключ API представляет собой запись бота в Telegram.

Для того, чтобы бот функционировал, его сначала нужно зарегистрировать в Telegram, получить bot token. Для этого в Telegram есть специальный мета-бот BotFather (@BotFather). Нужно добавить этого бота в свой список диалогов через поиск в клиенте Telegram. Список доступных ему команд можно получить, написав в чате с ним команду /help. Для создания бота нужно использовать команду /newbot и следовать предложенной инструкции. В ответ придёт сообщение с API ключом (примерно такого вида: 123456789:AbCdfGhIJKlmNoQQRsTUVwxyz).

2.1 Получение списка групп с сайта СГУ

Для получения расписания с сайта, нужно было разработать парсер, который бы преобразовывал данные со страницы <https://www.sgu.ru/schedule>

в список факультетов ВУЗа, а затем для каждого факультета преобразовывал страницу <https://www.sgu.ru/schedule/faculty>, где faculty - один из факультетов.

Примерная структура данных после парсинга представлена в приложении Б на странице 36, Листинг 5

Для этого были написаны следующие функции:

Листинг 1: Функция получения списка факультетов

```
1 function getFacultiesList() {
2     xray('https://www.sgu.ru/schedule', 'div.panes_item.
   panes_item__type_group > ul > li', [{
3         name: 'a',
4         path: 'a@href'
5     }])
6     .then(async function (res) {
7         await res.forEach(async faculty => {
8             await readFacultyInfo(faculty);
9         });
10    })
11
12    .catch(function (err) {
13        console.log(err);
14    });
15
16 }
```

В этой функции с помощью модуля объектом модуля Xray [5, 7] — xray — который является веб скрапером, получаем данные со страницы со списком факультетов. С помощью селекторов выбираем факультеты и для каждого факультета переходим на страницу с его группами.

Листинг 2: Функция получения списка групп факультета

```
1
2 function readFacultyInfo(faculty) {
3     xray(faculty.path, 'fieldset.form_education.form-wrapper', [{
4         form: 'legend > span',
5         group_types: xray('fieldset.group-type.form-wrapper', [{
6             group_type: 'legend > span',
7             courses: xray('fieldset.course.form-wrapper', [{
8                 course: 'legend > span',
9                 groups: xray('div', [{
10                     number: ['a'],
11                     path: ['a@href']
12                 }])
13             }])
14         }])
15     }])
16 }
```

```

14
15         })
16
17     })
18     .write(__dirname + '/paths/' + faculty.name + '.json');
19 }

```

В этой функции из параметров получается ссылка на страницу со списком групп факультета и с помощью селекторов группы делятся по формам обучения, типам обучения и курсам. Затем полученные данные записываются в соответствующий названию факультета файл.

Результат работы функций для факультета компьютерных наук и информационных технологий представлен в приложении Б на странице 37, Листинг 6

2.2 Получение интересующей пользователя группы

Для получения группы, интересующей пользователя, была написана сцена student.

Схема работы сцены:

1. Получение и вывод списка факультетов пользователю, добавление в меню кнопки “Назад” для возвращения в главное меню
2. — Если была выбрана кнопка “Назад” или введено не верное значение, то возврат в сцену по—умолчанию, в которой предлагается выбрать, какое расписание нужно: студентов или преподавателей.
— Иначе получение и вывод доступных форм обучения, добавление кнопки “Назад” для возвращения к выбору факультета
3. — Если была выбрана кнопка “Назад”, то возврат к выбору факультета
— Если было введено не верное значение, то возврат в сцену по—умолчанию
— Иначе получение и вывод доступных типов обучения, добавление кнопки “Назад” для возвращения к выбору формы обучения
4. — Если была выбрана кнопка “Назад”, то возврат к выбору формы обучения
— Если было введено не верное значение, то возврат в сцену по—умолчанию
— Иначе получение и вывод доступных курсов, добавление кнопки

- “Назад” для возвращения к выбору типа обучения
5. — Если была выбрана кнопка “Назад”, то возврат к выбору типа обучения
 - Если было введено не верное значение, то возврат в сцену по—умолчанию
 - Иначе получение и вывод доступных групп, добавление кнопки “Назад” для возвращения к выбору курса
 6. — Если была выбрана кнопка “Назад”, то возврат к выбору курса
 - Если было введено не верное значение, то возврат в сцену по—умолчанию
 - Иначе вывод расписания группы и предложение выбора другой группы, добавление кнопок “Да” и “Нет”
 7. — Если была выбрана кнопка “Да”, то переход к первому шагу
 - Иначе выход из сцены, переход в сцену по—умолчанию

Реализация предложенной схемы представлена в приложении В на странице 46, Листинг 7

2.3 Получение расписания на определённый день недели

Функция получения расписания на день недели является общей для студентов и преподавателей. Схематически, функция выглядит следующим образом:

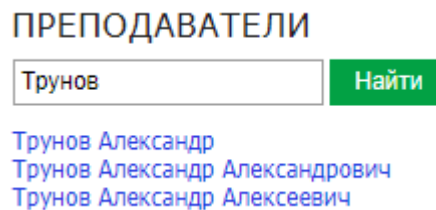
1. Переход по ссылке с расписанием группы/преподавателя
2. Получение временных рамок занятий
3. Считывание расписания всей таблицы. На этом этапе получены все занятия - для каждого дня для каждого времени
4. Преобразуем полученные занятия в следующий вид:
 - а) Чётность — числитель или знаменатель
 - б) Тип — практическое, лекционное или лабораторное занятие
 - в) Номер подгруппы
 - г) Название занятия
 - д) Имя преподавателя
 - е) Номер аудитории и корпуса
 - ж) Номер группы — используется для расписания преподавателей
5. Разбиваем занятия по дням недели
6. Преобразуем занятия каждого дня в строчный вид

7. Передаём результат в точку вызова программы

Реализация предложенной схемы представлена в приложении Г на странице 51, Листинг 8

2.4 Получение расписания преподавателя

Для получения расписания преподавателя на сайте СГУ есть специальная форма на странице <https://www.sgu.ru/schedule>, при вводе в которую фамилии или части фамилии преподавателя выдаётся список преподавателей, подходящих запросу.



ПРЕПОДАВАТЕЛИ

Трунов

Трунов Александр
Трунов Александр Александрович
Трунов Александр Алексеевич

Рисунок 2.2 – Пример работы формы

При нажатии на кнопку “Найти”, происходит обновление содержимого страницы, появляется новый блок, в котором содержится список преподавателей. Поэтому использовать модуль `request` [8,10] будет нельзя. Для того чтобы решить эту проблему, был использован модуль `Nightmare` [9]. Этот модуль создаёт виртуально “окно браузера”, в котором он может взаимодействует со страницей сайта, и затем выдаёт содержимое страницы после выполнения операций.

Схематически работа с `Nightmare` для поиска преподавателей выглядит следующим образом:

1. Подключение к странице <https://www.sgu.ru/schedule>
2. Ожидание появления элемента `body` страницы
3. Ввод в форму фамилии или части фамилии преподавателя
4. Нажатие на кнопку “Нажать”
5. Ожидание ответа сервер СГУ — списка преподавателей; преобразование динамической страницы в статическую
6. Получение списка преподавателей с полученной страницы

Реализация предложенной схемы представлена в приложении Д на странице 54, Листинг 9

2.5 Сцена получения расписания преподавателя

Схематически сцена выглядит следующим образом:

1. Считывание фамилии, введенной пользователем
2. Поиск фамилии на сайте СГУ
 - Если поиск не дал результатов, то вывод сообщения об этом пользователю и предложение ввести фамилию заново, переход к шагу 1
 - Если поиск дал более 15 результатов, то вывод пользователю сообщения о том, что поиск не точный и предложение уточнить введенную фамилию
 - Иначе вывод пользователю списка преподавателей с заданной фамилией
3. Выбор пользователем нужного преподавателя
4. Вывод расписание выбранного преподавателя на сегодня

Реализация предложенной схемы представлена в приложении Е на странице 55, Листинг 10

2.6 Главная сцена

На главной сцене пользователь может выбрать, что он хочет найти: расписание группы или расписание преподавателя. Если нажата кнопка “Группы”, то производится переход в сцену поиска расписания группы, если нажата кнопка “Преподавателя” — расписание преподавателя. Так же на главной сцене пользователь может узнать доступные ему команды, введя команду “/help”.

Реализация главной сцены представлена в приложении Ж на странице 59, Листинг 11

2.7 Тестирование работы бота

При первом входе пользователя боту посылается стандартная команда /start, она обрабатывается ботом и производится вход в главную сцену. Пользователь может выбрать между поиском расписания групп и расписанием преподавателя. Пример входа можно увидеть на рисунке 2.3

Затем, если пользователь выбрал кнопку “Группы”, он попадает в сцену поиска расписания группы. Пользователь выбирает необходимый факультет, рисунок 2.4. После пользователю предлагается выбрать форму обучения (ри-

сунок 2.5). Затем предлагается выбрать тип обучения, рисунок 2.6. Предлагается выбрать курс, рисунок 2.7. Предлагается выбрать номер группы, рисунок 2.7. После этого пользователю выводится расписание на сегодня, либо на понедельник, если сегодня воскресенье, рисунок 2.9. Затем пользователю предлагается найти расписание другой группы, рисунок 2.10. Если пользователь выбрал “Да”, то он просто помещается в начало сцены, рисунок 2.11.

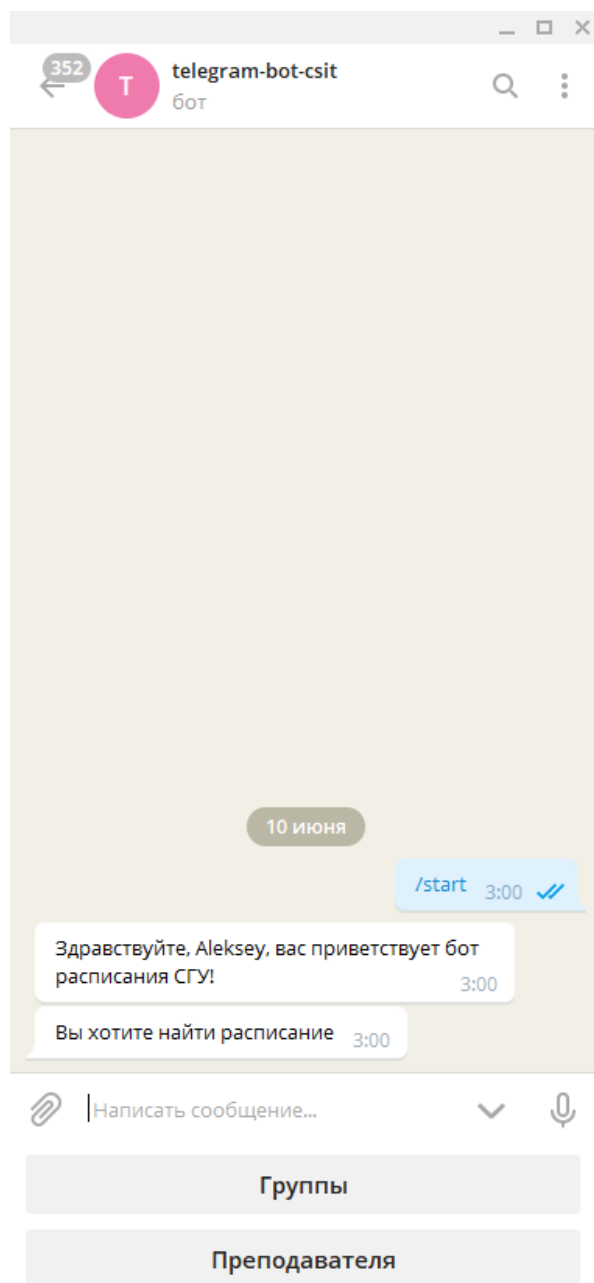


Рисунок 2.3 – Первый вход пользователя

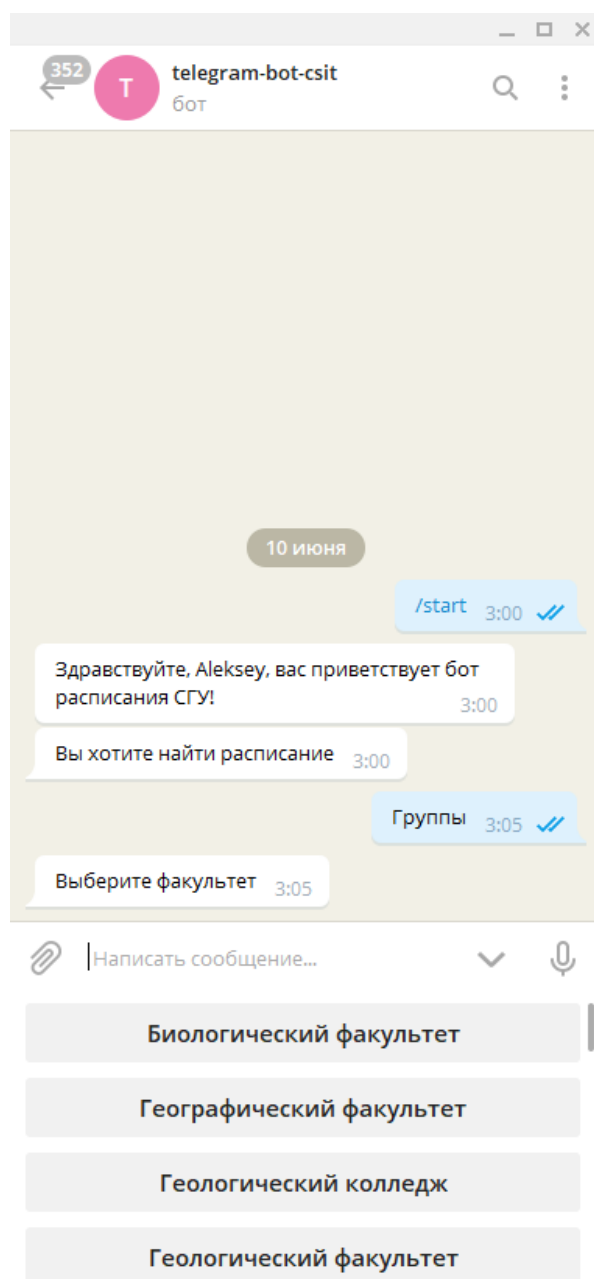


Рисунок 2.4 – Выбор факультета

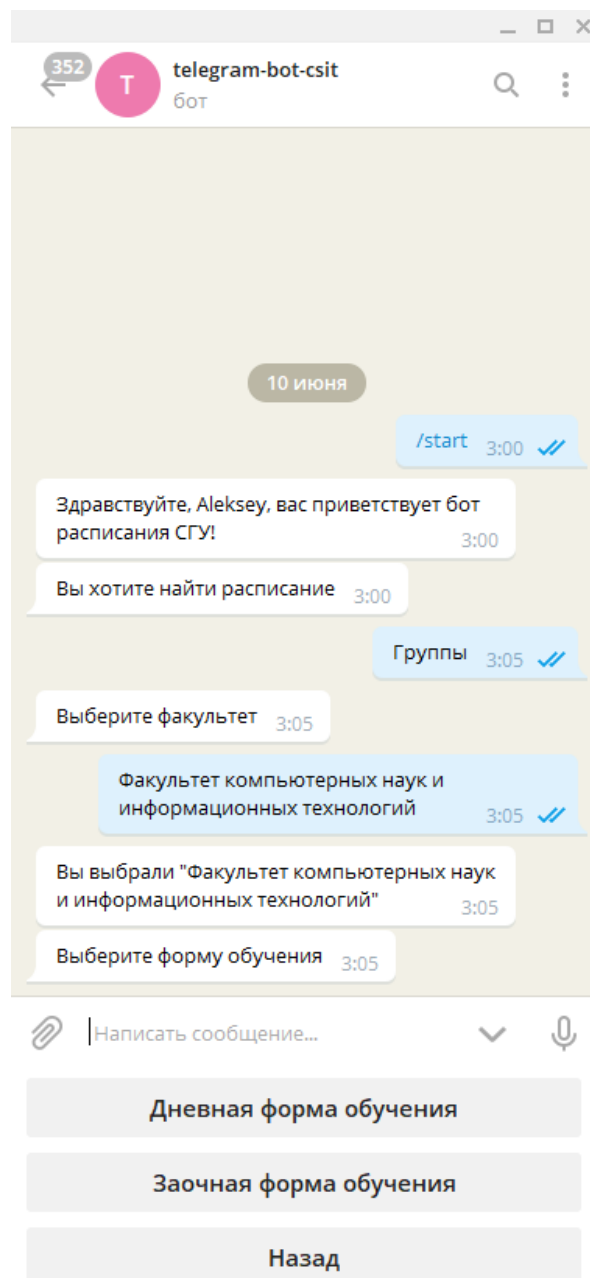


Рисунок 2.5 – Выбор формы обучения

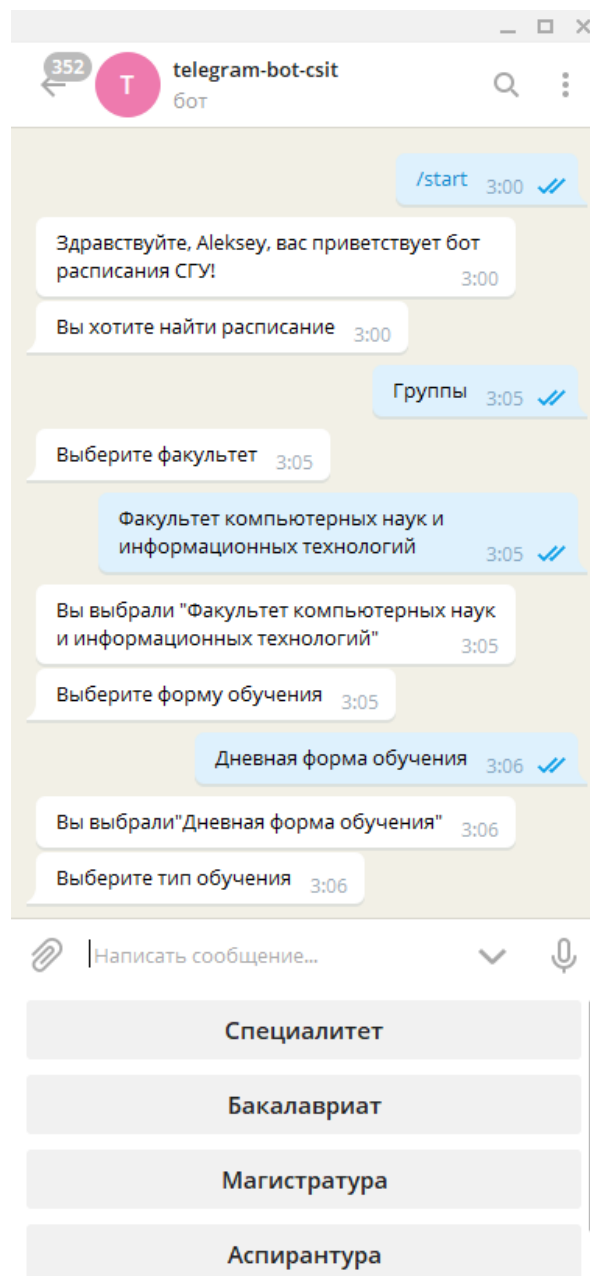


Рисунок 2.6 – Выбор типа обучения

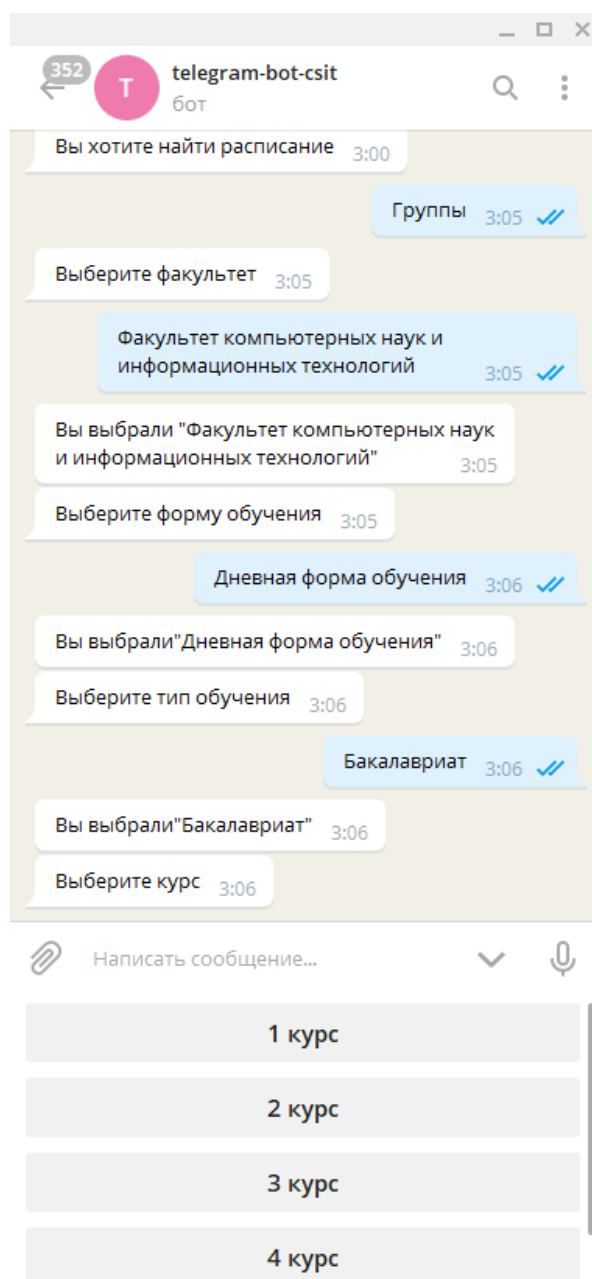


Рисунок 2.7 – Выбор курса

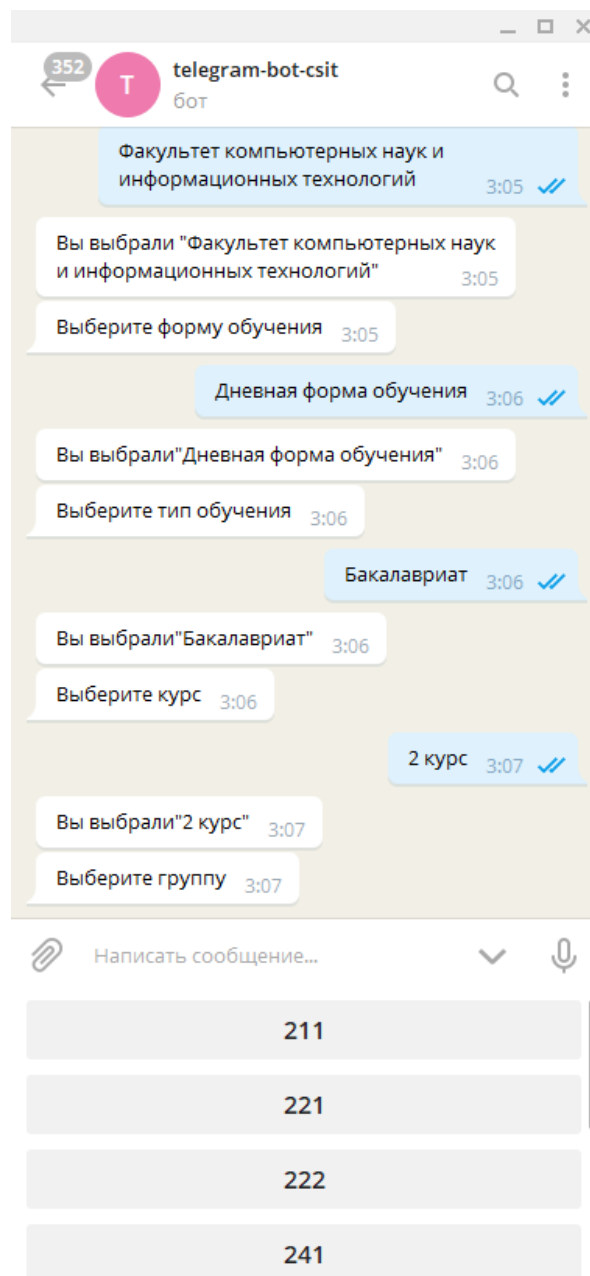


Рисунок 2.8 – Выбор номера группы

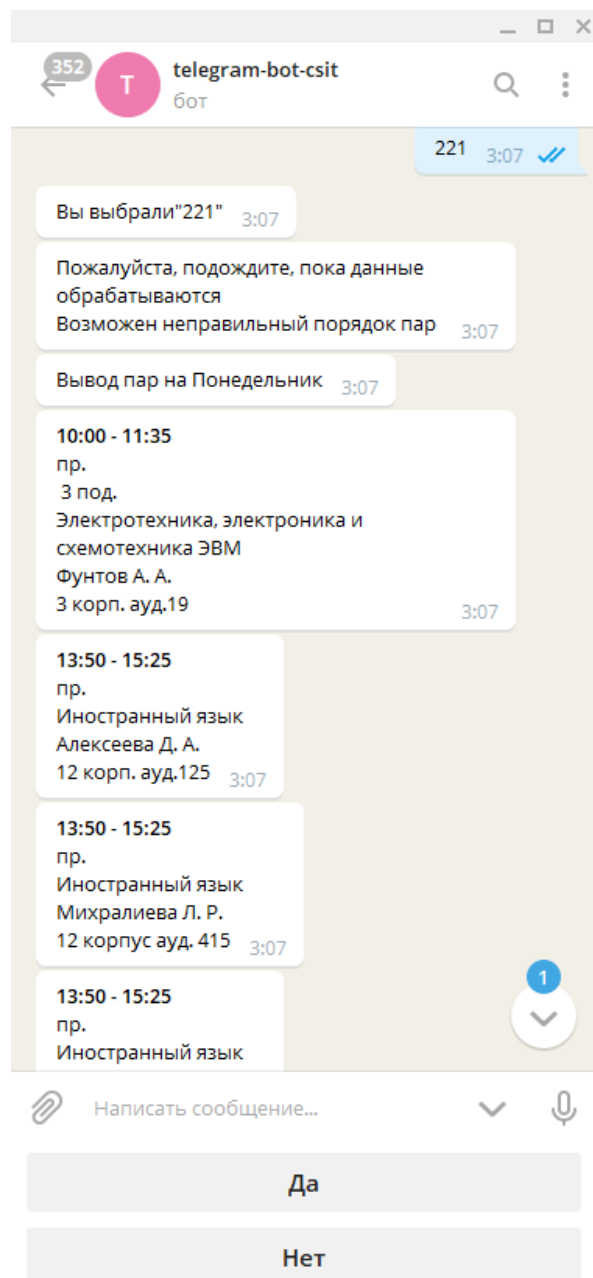


Рисунок 2.9 – Вывод расписания группы

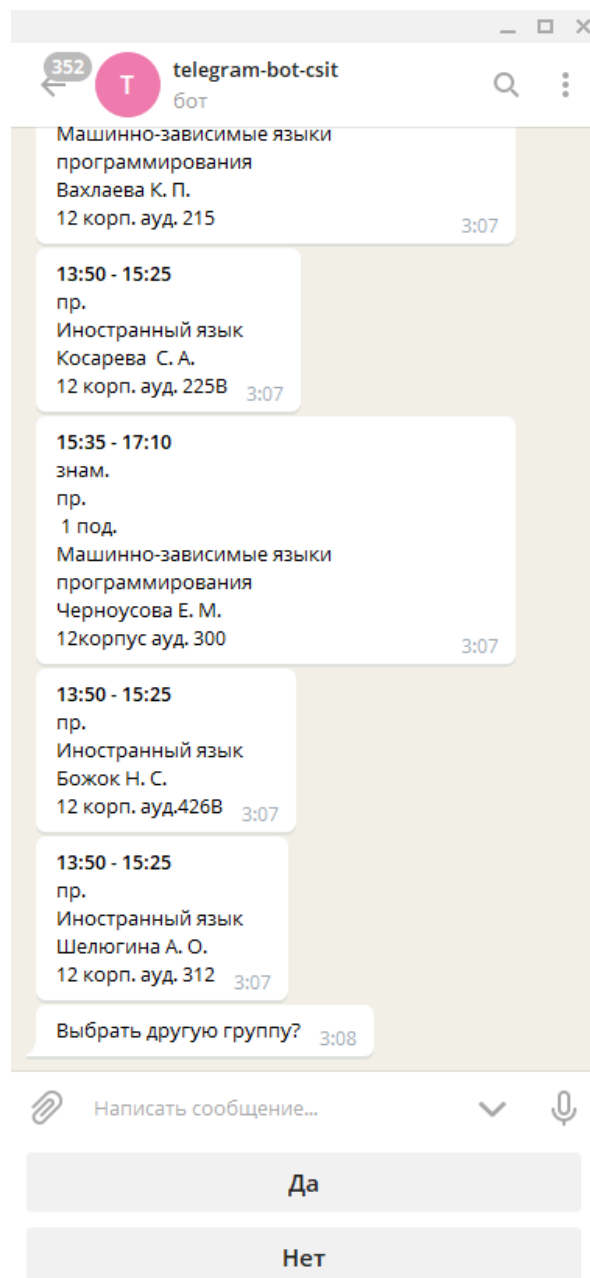


Рисунок 2.10 – Предложение выбора другой группы

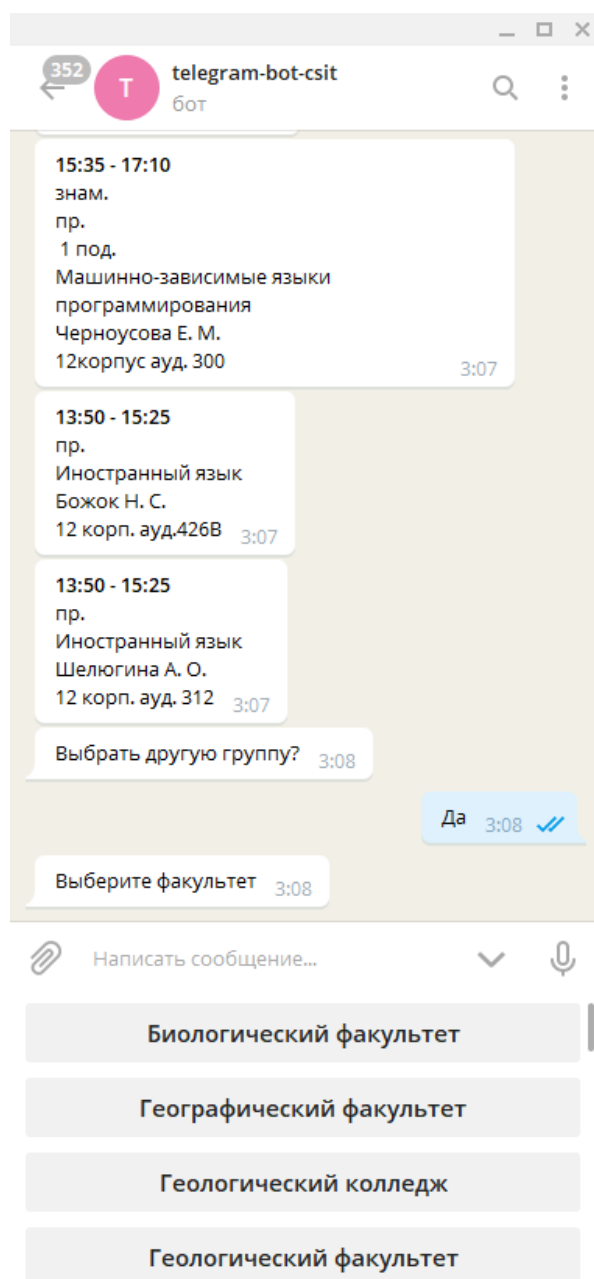


Рисунок 2.11 – Пользователь выбрал “Да”

Если же пользователь в главном меню выбрал “Преподавателя”, то он помещается в сцену поиска расписания преподавателя. Пользователю предлагается ввести фамилию преподавателя, рисунок 2.12. Пользователь вводит фамилию преподавателя, расписание которого он хочет посмотреть, рисунок 2.13, и бот выводит ему расписание преподавателя на сегодня. Если занятий сегодня нет, то выводится сообщение об этом, рисунок 2.14. Бот предлагает выбрать другого преподавателя. Если пользователь выбрал “Да”, то он просто перемещается в начало сцены, рисунок 2.15.

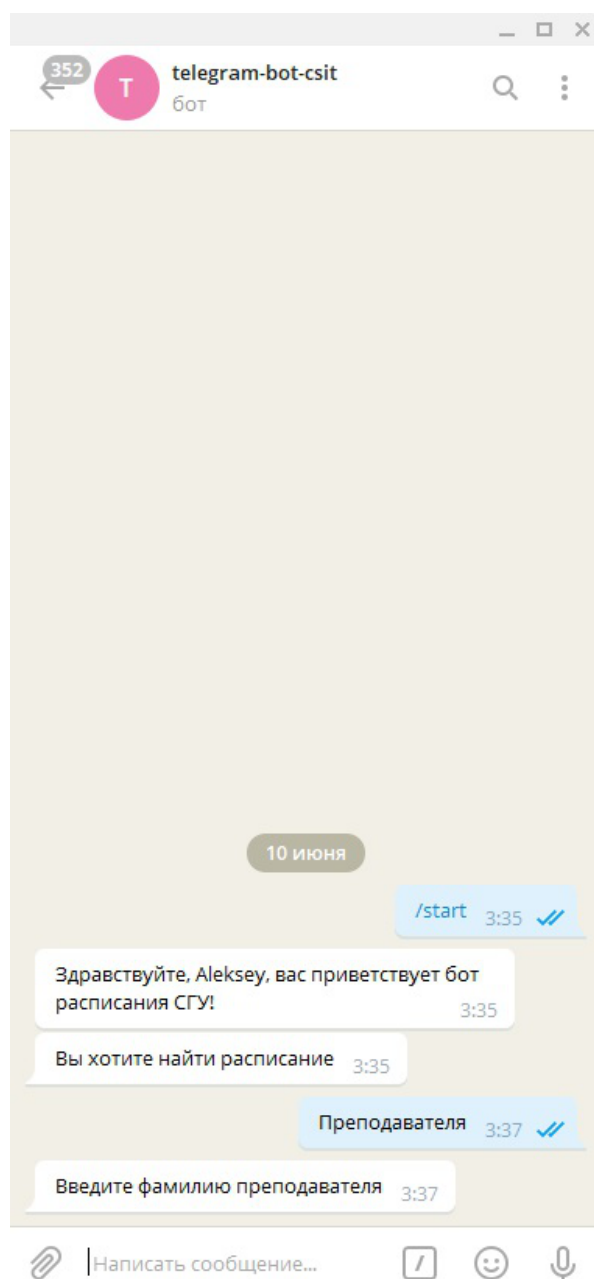


Рисунок 2.12 – Вход в сцену

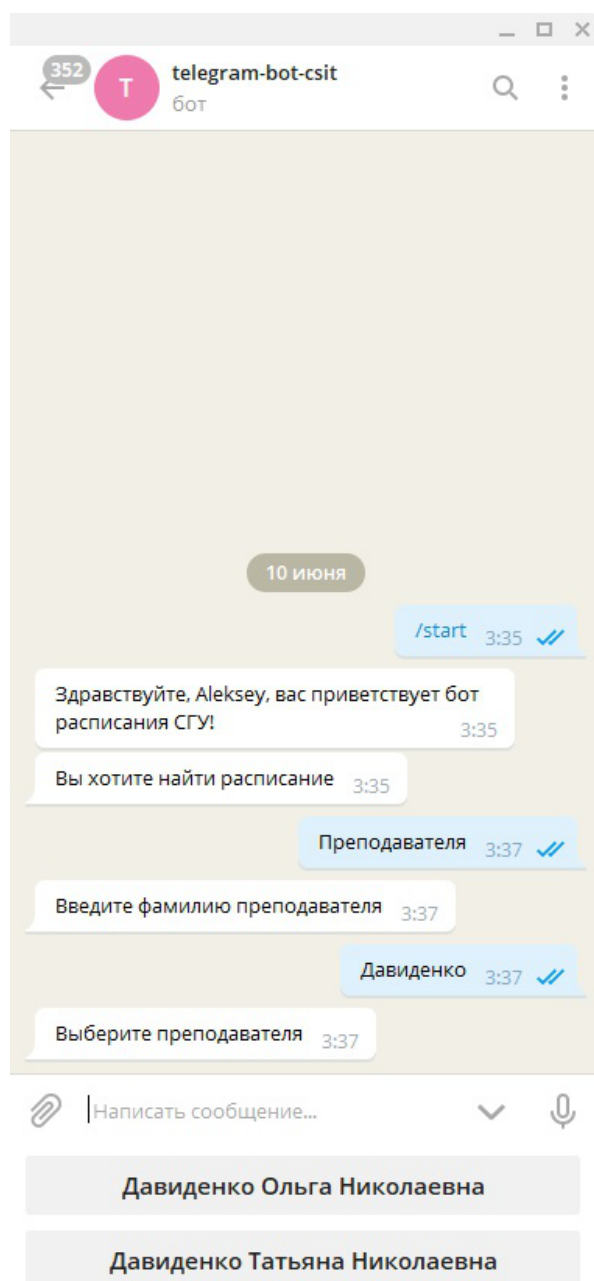


Рисунок 2.13 – Ввод фамилии преподавателя

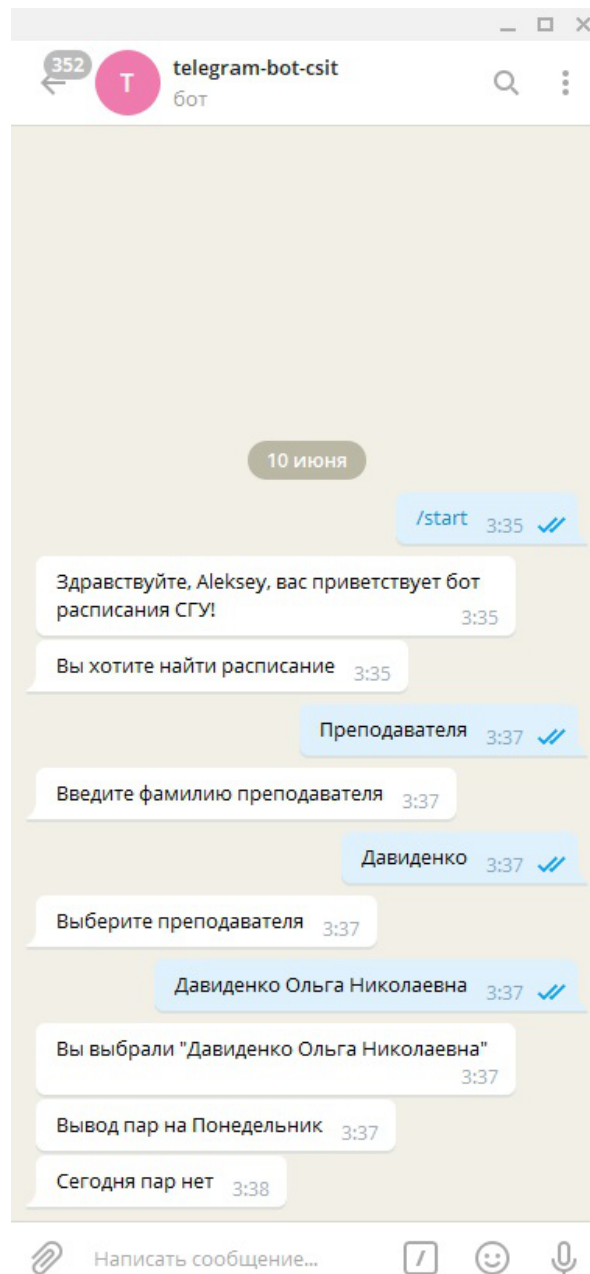


Рисунок 2.14 – Вывод расписания преподавателя

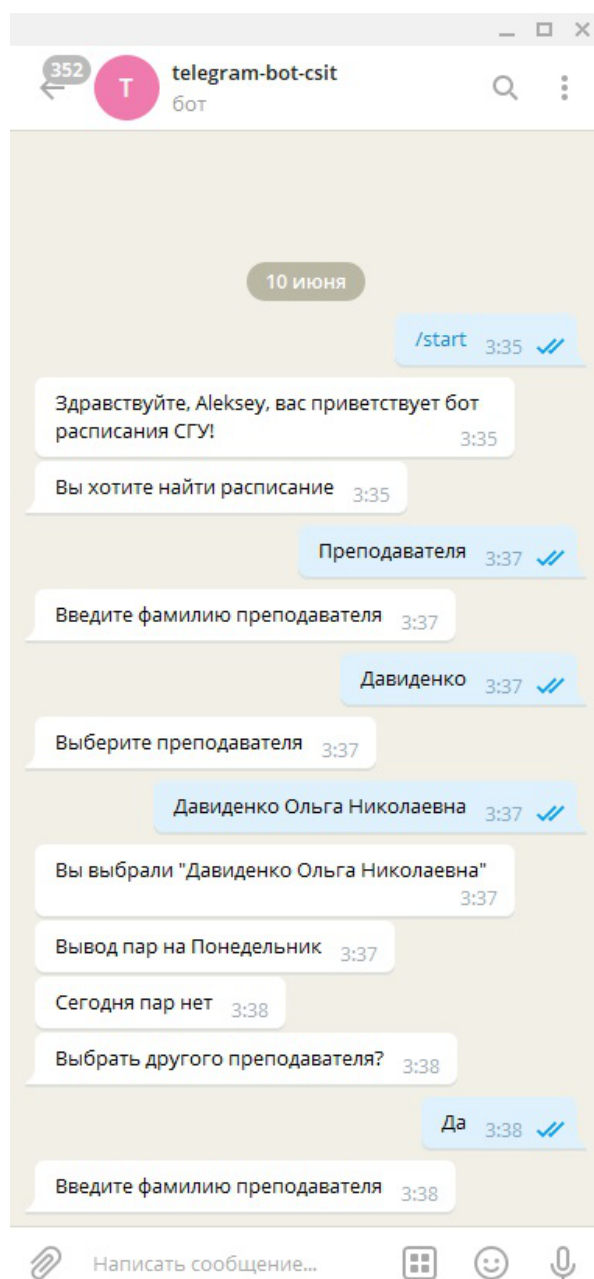


Рисунок 2.15 – Пользователь выбрал “Да”

ЗАКЛЮЧЕНИЕ

В процессе работы были выявлены качественные превосходства и недостатки использования Telegram как платформы для создания и размещения многофункциональных интерактивных помощников. Был проведён анализ предметной области, включающий в себя изучение документации Telegram Bot API, поиск актуальных методов решения поставленных задач. Было спроектировано и разработано программное обеспечение для работы с мессенджером Telegram Bot API с помощью Telegram Bot API, и для работы с сайтом СГУ, на котором представлено расписание студентов и преподавателей ВУЗа. Впоследствии, программное обеспечение было протестировано и отлажено.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- 1 telegraf.js [Электронный ресурс] // <https://telegraf.js.org> – Telegraf.js. URL: <https://telegraf.js.org/> (дата обращения 10.01.2019) Загл. с экрана Яз. рус
- 2 Telegram Bot API [Электронный ресурс] // <https://core.telegram.org/> – Telegram Bot API. URL: <https://core.telegram.org/bots/api> (дата обращения 08.01.2019) Загл. с экрана Яз. рус
- 3 Node.JS [Электронный ресурс] // <https://nodejs.org/> – node.JS URL: <https://nodejs.org/en/> (дата обращения 10.01.2019) Загл. с экрана Яз. рус
- 4 Medium [Электронный ресурс] // <https://chatbotslife.com/> – [Part 1]-Build a Simple Currency Converter Bot with Node.Js. URL: <https://chatbotslife.com/build-a-simple-telegram-currency-converter-bot-with-node-js-84d15b10597c> (дата обращения 04.03.2019) Загл. с экрана Яз. рус
- 5 Medium [Электронный ресурс] // <https://medium.com/> – Веб-скрапинг с помощью Node.js-Часть 2. URL: <https://medium.com/nuances-of-programming/как-выполнить-веб-скрапинг-с-помощью-node-js-часть-2-4476884ebf11> (дата обращения 01.03.2019) Загл. с экрана Яз. рус
- 6 Habr [Электронный ресурс] // <https://habr.com/> – Telegram-bot: моя история. Часть первая URL: <https://habr.com/ru/post/316868/> (дата обращения 06.02.2019) Загл. с экрана Яз. рус
- 7 npm [Электронный ресурс] // <https://npm.com/> – xray URL: <https://www.npmjs.com/package/x-ray> (дата обращения 04.02.2019) Загл. с экрана Яз. рус
- 8 npm [Электронный ресурс] // <https://npm.com/> – request URL: <https://www.npmjs.com/package/request> (дата обращения 16.03.2019) Загл. с экрана Яз. рус
- 9 npm [Электронный ресурс] // <https://npm.com/> – nightmare URL: <https://www.npmjs.com/package/nightmare> (дата обращения 26.03.2019) Загл. с экрана Яз. рус
- 10 npm [Электронный ресурс] // <https://npm.com/> – cheerio URL: <https://www.npmjs.com/package/cheerio> (дата обращения 16.03.2019) Загл. с экрана Яз. рус

ПРИЛОЖЕНИЕ А

Примеры программ

Листинг 3: Пример использования Telegraf

```
1 const bot = new Telegraf(BOT_TOKEN) // Token Telegram Bot API
2
3 bot.use(async (ctx, next) => {
4   const start = new Date()
5   await next()
6   const ms = new Date() - start
7   console.log('Response time %sms', ms)
8 })
9
10 bot.on('text', (ctx) => ctx.reply('Hello World'))
11
12 bot.launch()
13
14 bot.catch((err) => {
15   console.log('Ooops', err)
16 })
```

Листинг 4: Пример использования сцен

```
1 const Telegraf = require("telegraf"); // import telegraf lib
2 const Markup = require("telegraf/markup"); // Get the markup module
3 const Stage = require("telegraf/stage");
4 const session = require("telegraf/session");
5 const WizardScene = require("telegraf/scenes/wizard");
6
7 const bot = new Telegraf(process.env.BOT_TOKEN); // Get the token from the
   environment variable
8
9 // Start Bot
10 bot.start(ctx => {
11   return ctx.scene.enter('default');
12 });
13
14 // Go back to menu after action
15 const default = new WizardScene({
16   "default",
17   ctx => {
18     ctx.reply(`Glad I could help`);
19     ctx.reply(
20       `What are you want to do, ${ctx.from.first_name}?`,
21       Markup.inlineKeyboard([
22         Markup.callbackButton("Buy tickets", "BUY_TICKETS"),
23         Markup.callbackButton("Watch memes", "WATCH_MEMES")
24       ]).extra()
25     );
26   }
27 });
```

```

25         );
26     },
27     ctx => {return ctx.scene.leave()};
28
29 });
30 bot.action("BUY_TICKETS", ctx => {
31     return ctx.scene.enter('ticketsBuyer');
32 });
33 bot.action("WATCH_MEMES", ctx => {
34     return ctx.scene.enter('memesPresenter');
35 });
36
37 // Tickets buyer Wizard
38 const ticketsBuyer = new WizardScene(
39     "tickets_buyer",
40     ctx => {
41         ctx.reply("Please, type in your last name");
42         /*some actions*/
43         return ctx.wizard.next();
44     },
45     ctx => {
46         /*
47         * ctx.wizard.state is the state management object which is persistent
48         * throughout the wizard
49         * we pass to it the previous user reply (supposed to be the source
50         Currency )
51         * which is retrieved through `ctx.message.text`
52         */
53         ctx.wizard.state.lastName = ctx.message.text;
54         ctx.reply(
55             `Got it, your last name ${
56                 ctx.wizard.state.lastName
57             }`
58         );
59         ctx.reply(`So, what is your first name?`)
60         /*some actions*/
61         return ctx.wizard.next();
62     },
63     /*some steps*/
64     ctx => {
65         /*some action*/
66         return ctx.scene.enter('default')
67     }
68 );
69 // Memes presenter Wizard
70 const memesPresenter = new WizardScene("memesPresenter", /*...*/);
71

```

```
72 // Scene registration
73 const stage = new Stage([ticketsBuyer, memesPresenter], { default: "default"
    });
74 bot.use(session());
75 bot.use(stage.middleware());
76
77 bot.launch()
78
79 bot.catch((err) => {
80   console.log('Ooops', err)
81 })
```

ПРИЛОЖЕНИЕ Б

Структура списка групп с сайта СГУ

Листинг 5: Структура данных после обработки парсером

```
[
  {
    "form": "form",
    "group_types": [
      {
        "group_type": "groupType",
        "courses": [
          {
            "course": "course",
            "groups": [
              {
                "number": [
                  "number"
                ],
                "path": [
                  "https://www.sgu.ru/schedule/faculty/form/number"
                ]
              }
            ]
          },
          {
            "course": "...",
            "groups": [
              {
                "number": [
                  "..."
                ],
                "path": [
                  "https://www.sgu.ru/schedule/faculty/.../..."
                ]
              }
            ]
          },
          {...}
        ]
      },
      {
        "group_type": "...",
        "courses": [
          {
            "course": "...",
            "groups": [
              {
```

```

        "number": [
            "...",
            "...",
            "...",
            "...",
            "...",
            "..."
        ],
        "path": [
            "...",
            "...",
            "...",
            "...",
            "...",
            "..."
        ]
    }
}
},
{...}
]
},
{...}
]
},
{...}
]

```

Листинг 6: Пример результата парса страницы расписания ф. КН_иИТ

```
[
{
  "form": "Дневная" форма обучения",
  "group_types": [
    {
      "group_type": "Специалитет",
      "courses": [
        {
          "course": "1 курс:",
          "groups": [
            {
              "number": [
                "131",
                "132"
              ],
              "path": [
                "https://www.sgu.ru/schedule/knt/do/131",
                "https://www.sgu.ru/schedule/knt/do/132"
              ]
            }
          ]
        }
      ]
    }
  ]
}
```

```

    }
  ]
},
{
  "course": "2 кypc:",
  "groups": [
    {
      "number": [
        "231",
        "232"
      ],
      "path": [
        "https://www.sgu.ru/schedule/knt/do/231",
        "https://www.sgu.ru/schedule/knt/do/232"
      ]
    }
  ]
},
{
  "course": "3 кypc:",
  "groups": [
    {
      "number": [
        "331",
        "332"
      ],
      "path": [
        "https://www.sgu.ru/schedule/knt/do/331",
        "https://www.sgu.ru/schedule/knt/do/332"
      ]
    }
  ]
},
{
  "course": "4 кypc:",
  "groups": [
    {
      "number": [
        "431"
      ],
      "path": [
        "https://www.sgu.ru/schedule/knt/do/431"
      ]
    }
  ]
},
{
  "course": "5 кypc:",

```

```

    "groups": [
      {
        "number": [
          "531"
        ],
        "path": [
          "https://www.sgu.ru/schedule/knt/do/531"
        ]
      }
    ]
  },
  {
    "group_type": "Бакалавриат",
    "courses": [
      {
        "course": "1 курс",
        "groups": [
          {
            "number": [
              "111",
              "112",
              "121",
              "122",
              "141",
              "142",
              "151",
              "152",
              "161",
              "181"
            ],
            "path": [
              "https://www.sgu.ru/schedule/knt/do/111",
              "https://www.sgu.ru/schedule/knt/do/112",
              "https://www.sgu.ru/schedule/knt/do/121",
              "https://www.sgu.ru/schedule/knt/do/122",
              "https://www.sgu.ru/schedule/knt/do/141",
              "https://www.sgu.ru/schedule/knt/do/142",
              "https://www.sgu.ru/schedule/knt/do/151",
              "https://www.sgu.ru/schedule/knt/do/152",
              "https://www.sgu.ru/schedule/knt/do/161",
              "https://www.sgu.ru/schedule/knt/do/181"
            ]
          }
        ]
      }
    ]
  },
  {

```

```

"course": "2 кypc:",
"groups": [
  {
    "number": [
      "211",
      "221",
      "222",
      "241",
      "242",
      "251",
      "261",
      "281"
    ],
    "path": [
      "https://www.sgu.ru/schedule/knt/do/211",
      "https://www.sgu.ru/schedule/knt/do/221",
      "https://www.sgu.ru/schedule/knt/do/222",
      "https://www.sgu.ru/schedule/knt/do/241",
      "https://www.sgu.ru/schedule/knt/do/242",
      "https://www.sgu.ru/schedule/knt/do/251",
      "https://www.sgu.ru/schedule/knt/do/261",
      "https://www.sgu.ru/schedule/knt/do/281"
    ]
  }
]
},
{
  "course": "3 кypc:",
  "groups": [
    {
      "number": [
        "311",
        "321",
        "341",
        "351",
        "361",
        "381"
      ],
      "path": [
        "https://www.sgu.ru/schedule/knt/do/311",
        "https://www.sgu.ru/schedule/knt/do/321",
        "https://www.sgu.ru/schedule/knt/do/341",
        "https://www.sgu.ru/schedule/knt/do/351",
        "https://www.sgu.ru/schedule/knt/do/361",
        "https://www.sgu.ru/schedule/knt/do/381"
      ]
    }
  ]
}
]

```



```

    },
    {
      "course": "4 кypc:",
      "groups": [
        {
          "number": [
            "411",
            "421",
            "441",
            "451",
            "461",
            "481"
          ],
          "path": [
            "https://www.sgu.ru/schedule/knt/do/411",
            "https://www.sgu.ru/schedule/knt/do/421",
            "https://www.sgu.ru/schedule/knt/do/441",
            "https://www.sgu.ru/schedule/knt/do/451",
            "https://www.sgu.ru/schedule/knt/do/461",
            "https://www.sgu.ru/schedule/knt/do/481"
          ]
        }
      ]
    }
  ],
  {
    "group_type": "Maиicтpиaтyпa",
    "courses": [
      {
        "course": "1 кypc:",
        "groups": [
          {
            "number": [
              "171",
              "172",
              "173"
            ],
            "path": [
              "https://www.sgu.ru/schedule/knt/do/171",
              "https://www.sgu.ru/schedule/knt/do/172",
              "https://www.sgu.ru/schedule/knt/do/173"
            ]
          }
        ]
      }
    ]
  },
  {
    "course": "2 кypc:",

```

```

    "groups": [
      {
        "number": [
          "271",
          "272",
          "273"
        ],
        "path": [
          "https://www.sgu.ru/schedule/knt/do/271",
          "https://www.sgu.ru/schedule/knt/do/272",
          "https://www.sgu.ru/schedule/knt/do/273"
        ]
      }
    ]
  },
  {
    "group_type": "Аспирантура",
    "courses": [
      {
        "course": "1 курс:",
        "groups": [
          {
            "number": [
              "191",
              "192",
              "193"
            ],
            "path": [
              "https://www.sgu.ru/schedule/knt/do/191",
              "https://www.sgu.ru/schedule/knt/do/192",
              "https://www.sgu.ru/schedule/knt/do/193"
            ]
          }
        ]
      },
      {
        "course": "2 курс:",
        "groups": [
          {
            "number": [
              "291",
              "292",
              "293"
            ],
            "path": [
              "https://www.sgu.ru/schedule/knt/do/291",

```

```

        "https://www.sgu.ru/schedule/knt/do/292",
        "https://www.sgu.ru/schedule/knt/do/293"
    ]
}
]
},
{
    "course": "3 курс:",
    "groups": [
        {
            "number": [
                "391",
                "392",
                "393"
            ],
            "path": [
                "https://www.sgu.ru/schedule/knt/do/391",
                "https://www.sgu.ru/schedule/knt/do/392",
                "https://www.sgu.ru/schedule/knt/do/393"
            ]
        }
    ]
},
{
    "course": "4 курс:",
    "groups": [
        {
            "number": [
                "492"
            ],
            "path": [
                "https://www.sgu.ru/schedule/knt/do/492"
            ]
        }
    ]
}
]
}
]
},
{
    "form": "Заочная" форма обучения",
    "group_types": [
        {
            "group_type": "Бакалавриат",
            "courses": [
                {
                    "course": "1 курс:",

```

```

    "groups": [
      {
        "number": [
          "151",
          "152",
          "161"
        ],
        "path": [
          "https://www.sgu.ru/schedule/knt/zo/151",
          "https://www.sgu.ru/schedule/knt/zo/152",
          "https://www.sgu.ru/schedule/knt/zo/161"
        ]
      }
    ]
  },
  {
    "course": "2 кypc:",
    "groups": [
      {
        "number": [
          "251"
        ],
        "path": [
          "https://www.sgu.ru/schedule/knt/zo/251"
        ]
      }
    ]
  },
  {
    "course": "3 кypc:",
    "groups": [
      {
        "number": [
          "351",
          "361"
        ],
        "path": [
          "https://www.sgu.ru/schedule/knt/zo/351",
          "https://www.sgu.ru/schedule/knt/zo/361"
        ]
      }
    ]
  },
  {
    "course": "4 кypc:",
    "groups": [
      {
        "number": [

```


ПРИЛОЖЕНИЕ В

Метод получения интересующей пользователя группы

Листинг 7: Реализация функции, возможен неправильный порядок пробельных и специальных символов

```
1  const student = new WizardScene(  
2      "student",  
3      // Начало меню выбора  
4      async (ctx) => {  
5          leave("def");  
6          ctx.wizard.state.faculties = await new readFacultiesList();  
7          ctx.wizard.state.faculties.push("Назад");  
8  
9          ctx.reply("Выберите факультет", Markup.keyboard(ctx.wizard.state.  
10             faculties).resize().extra());  
11  
12             return ctx.wizard.next();  
13         },  
14         // Выбор факультета, предложение выбора формы обучения  
15         async (ctx) => {  
16             ctx.wizard.state.faculty = ctx.message.text;  
17  
18             if (ctx.wizard.state.faculty === "Назад") {  
19                 ctx.reply('Вы выбрали Назад""');  
20                 return ctx.scene.enter('def');  
21             }  
22  
23             if (ctx.wizard.state.faculties.indexOf(ctx.wizard.state.faculty) ==  
24                 -1) {  
25                 ctx.reply("Вы ввели неверное значение, выход из меню выбора", Markup.  
26                     removeKeyboard(true).oneTime().resize().extra());  
27                 return ctx.scene.enter('def');  
28             }  
29  
30             ctx.reply('Вы выбрали "' + ctx.wizard.state.faculty + '"');  
31  
32             ctx.wizard.state.forms = await new readFacultyForms(ctx.wizard.state.  
33                 .faculty);  
34             ctx.wizard.state.forms.push("Назад");  
35             ctx.reply('Выберите форму обучения', Markup.keyboard(ctx.wizard.state.  
36                 forms).resize().extra());  
37  
38             return ctx.wizard.next();  
39         },  
40         // Выбор формы обучения, предложение выбора типа обучения
```

```

38     async (ctx) => {
39         ctx.wizard.state.form = ctx.message.text;
40
41         if (ctx.wizard.state.form === "Назад") {
42             ctx.reply('Вы выбрали Назад"');
43             ctx.reply("Выберите факультет", Markup.keyboard(ctx.wizard.state.
faculties).resize().extra());
44             return ctx.wizard.back();
45         }
46
47         if (ctx.wizard.state.forms.indexOf(ctx.wizard.state.form) == -1) {
48             ctx.reply("Вы ввели неверное значение, выход из меню выбора", Markup.
removeKeyboard(true).oneTime().resize().extra());
49             return ctx.scene.enter('def');
50         }
51
52         ctx.reply('Вы выбрали"' + ctx.wizard.state.form + '"');
53
54         ctx.wizard.state.types = await new readFormGroupTypes(ctx.wizard.
state.faculty, ctx.wizard.state.form);
55         ctx.wizard.state.types.push("Назад");
56         ctx.reply('Выберите тип обучения', Markup.keyboard(ctx.wizard.state.
types).resize().extra());
57
58         return ctx.wizard.next();
59     },
60
61     // Выбор типа обучение, предложение выбора курса
62     async (ctx) => {
63         ctx.wizard.state.type = ctx.message.text;
64
65         if (ctx.wizard.state.type === "Назад") {
66             ctx.reply('Вы выбрали Назад"');
67             ctx.reply('Выберите форму обучения', Markup.keyboard(ctx.wizard.
state.forms).resize().extra());
68             return ctx.wizard.back();
69         }
70
71         if (ctx.wizard.state.types.indexOf(ctx.wizard.state.type) == -1) {
72             ctx.reply("Вы ввели неверное значение, выход из меню выбора", Markup.
removeKeyboard(true).oneTime().resize().extra());
73             return ctx.scene.enter('def');
74         }
75
76         ctx.reply('Вы выбрали"' + ctx.wizard.state.type + '"');
77
78         ctx.wizard.state.courses = await new readCourses(ctx.wizard.state.
faculty, ctx.wizard.state.form, ctx.wizard.state.type);

```

```

79         ctx.wizard.state.courses.push("Назад");
80         ctx.reply('Выберите курс', Markup.keyboard(ctx.wizard.state.courses).
resize().extra());
81
82         return ctx.wizard.next();
83     },
84
85     // Выбор курса, предложение выбора группы
86     async (ctx) => {
87         ctx.wizard.state.course = ctx.message.text;
88
89         if (ctx.wizard.state.course === "Назад") {
90             ctx.reply('Вы выбрали Назад');
91             ctx.reply('Выберите тип обучения', Markup.keyboard(ctx.wizard.state
.types).resize().extra());
92             return ctx.wizard.back();
93         }
94
95         if (ctx.wizard.state.courses.indexOf(ctx.wizard.state.course) == -1)
{
96             ctx.reply("Вы ввели неверное значение, выход из меню выбора", Markup.
removeKeyboard(true).oneTime().resize().extra());
97             return ctx.scene.enter('def');
98         }
99
100         ctx.reply('Вы выбрали' + ctx.wizard.state.course + '');
101
102         ctx.wizard.state.temp = await new readGroups(ctx.wizard.state.
faculty, ctx.wizard.state.form, ctx.wizard.state.type, ctx.wizard.state.
course);
103
104         ctx.wizard.state.groups = [];
105         ctx.wizard.state.paths = [];
106
107         ctx.wizard.state.temp.number.forEach(element => {
108             ctx.wizard.state.groups.push(element);
109         });
110
111         ctx.wizard.state.temp.path.forEach(element => {
112             ctx.wizard.state.paths.push(element);
113         });
114
115         ctx.wizard.state.groups.push("Назад");
116         ctx.reply('Выберите группу', Markup.keyboard(ctx.wizard.state.groups).
resize().extra());
117
118         return ctx.wizard.next();
119     },

```



```

120
121 // Выбор группы
122 async (ctx) => {
123     ctx.wizard.state.group = ctx.message.text;
124
125     if (ctx.wizard.state.group === "Назад") {
126         ctx.reply('Вы выбрали Назад');
127         ctx.reply('Выберите курс', Markup.keyboard(ctx.wizard.state.
courses).resize().extra());
128         return ctx.wizard.back();
129     }
130
131     if (ctx.wizard.state.groups.indexOf(ctx.wizard.state.group) == -1) {
132         ctx.reply("Вы ввели неверное значение, выход из меню выбора", Markup.
removeKeyboard(true).oneTime().resize().extra());
133         return ctx.scene.enter('def');
134     }
135
136     await ctx.reply('Вы выбрали"' + ctx.wizard.state.group + '"', Markup.
removeKeyboard(true).oneTime().resize().extra());
137     await ctx.reply("Пожалуйста, подождите, пока данные обрабатываются\Возможнн
неправильный порядок пар");
138
139     let weekday = weekdays[date.getDay()][0];
140
141     if (weekday !== "Воскресенье") {
142         await ctx.reply("Вывод пар на " + weekday);
143         await read(ctx.wizard.state.paths[ctx.wizard.state.groups.
indexOf(ctx.wizard.state.group)], weekday, (err, res) => {
144             if (res === "Сегодня пар нет" || res.length === 0) ctx.reply("
Сегодня пар нет");
145
146             else
147                 for (let index = 0; index < res.length; index++) {
148                     const element = res[index];
149                     ctx.replyWithHTML(element);
150                 }
151             });
152     }
153     else {
154         await ctx.reply("Так как сегодня Воскресенье, вывод пар на Понедельник")
;
155         await read(ctx.wizard.state.paths[ctx.wizard.state.groups.
indexOf(ctx.wizard.state.group)], "Понедельник", (err, res) => {
156             if (res === "Сегодня пар нет") ctx.reply("Сегодня пар нет");
157             else
158                 for (let index = 0; index < res.length; index++) {
159                     const element = res[index];

```

```

160             ctx.replyWithHTML(element);
161         }
162     });
163 }
164
165     setTimeout(() => {
166         ctx.reply("Выбрать другую группу?", Markup.keyboard(['Да', 'Нет']).
oneTime().resize().extra());
167     }, 10000);
168
169     return ctx.wizard.next();
170 },
171
172     async (ctx) => {
173         let answer = ctx.message.text;
174         if (answer === "Да") {
175             ctx.wizard.state.faculties = await new readFacultiesList();
176
177             ctx.reply("Выберите факультет", Markup.keyboard(ctx.wizard.state.
faculties).resize().extra());
178
179             return ctx.wizard.selectStep(1);
180         }
181         else return ctx.scene.enter('def');
182     }
183 );

```

ПРИЛОЖЕНИЕ Г

Метод получения расписания на определённый день недели

Листинг 8: Реализация предложенной схемы, возможен неправильный порядок пробельных и специальных символов

```
1 var cheerio = require('cheerio');
2 const request = require('request');
3
4 function read(url, day, callback) {
5     request(url, (err, res, body) => {
6         if (err)
7             return callback(err);
8         let $ = cheerio.load(body);
9         // Получаем время занятий
10        let classTimes = [];
11        $('table#schedule tr').each(function (index) {
12            let temp = $(this).find("th").first().text();
13            if (temp && temp.length !== 0)
14                classTimes.push(temp.slice(0, 5) + " - " + temp.slice(5));
15        });
16        let result = [];
17        // Получаем все занятия
18        $('table#schedule tr td').each(function (index) {
19            let str = "";
20            // Находим ячейку с расписанием и считываем из неё все данные
21            $(this).find("div.l").each(function (index) {
22                $(this).find("div.l-pr").each(function (index) {
23                    str += $(this).find("div.l-pr-r").text() + "\t" + $(this)
24                        .find("div.l-pr-t").text() + "\t" + $(this).find("div.l-pr-g").text() +
25                        "\t";
26                });
27                str += $(this).find("div.l-dn").text() + "\t";
28                str += $(this).find("div.l-tn").text() + "\t";
29                str += $(this).find("div.l-p").text() + "\t";
30                str += $(this).find("div.l-g").text() + "\t";
31            });
32            result.push(str);
33        });
34        for (let i = 0; i < result.length; i++) {
35            // Все проанализированные занятия
36            let classes = [];
37            result.forEach(_class => {
38                // Занятия в это время одним объектом
39                _class = _class.split("\t");
40                _class.pop();
41                // Проанализированные занятия в это время
```

```

41         let w = [];
42         // Количество занятий
43         let z = Array.from(_class).length / 7.0;
44         for (let j = 0; j < z; j++) {
45             let q = {
46                 parity: _class[0 + j * 7],
47                 type: _class[1 + j * 7],
48                 subgroup: _class[2 + j * 7],
49                 name: _class[3 + j * 7],
50                 professor: _class[4 + j * 7],
51                 classroom: _class[5 + j * 7],
52                 group_for_professors: _class[6 + j * 7] // Группа
студентов, для преподавателя
53             };
54             w.push(q);
55         }
56         classes.push(w);
57     });
58     // занятия, разбитые по времени и по дню недели
59     let time = {
60         Понедельник: [],
61         Вторник: [],
62         Среда: [],
63         Четверг: [],
64         Пятница: [],
65         Суббота: []
66     };
67     // Разбили занятия по дням
68     for (let i = 0; i < 8; i++) {
69         timeПонедельник..push(classes[0 + i * 6]);
70         timeВторник..push(classes[1 + i * 6]);
71         timeСреда..push(classes[2 + i * 6]);
72         timeЧетверг..push(classes[3 + i * 6]);
73         timeПятница..push(classes[4 + i * 6]);
74         timeСуббота..push(classes[5 + i * 6]);
75     }
76     // Преобразуем данные каждого дня в строку
77     timeString = [];
78     //console.log(time[day]);
79     if (time[day].includes(undefined))
80         return callback(null, "Сегодня пар нет");
81     for (let i = 0; i < 8; i++) {
82         time[day][i].forEach(_class => {
83             let str = "";
84             str += "<b>" + classTimes[i] + "</b>\n";
85             if (_class.parity.length > 0)
86                 str += _class.parity + "\n";
87             if (_class.type.length > 0)

```

```

88         str += _class.type + "\n";
89     if (_class.subgroup.length > 0)
90         str += _class.subgroup + "\n";
91     if (_class.name.length > 0)
92         str += _class.name + "\n";
93     if (_class.professor.length > 0)
94         str += _class.professor + "\n";
95     if (_class.classroom.length > 0)
96         str += _class.classroom + "\n";
97     if (_class.group_for_professors.length > 0)
98         str += _class.group_for_professors;
99     timeString.push(str);
100     });
101     }
102     //console.log(timeString);
103     callback(null, timeString);
104     });
105 }
106
107 module.exports = { read };

```

ПРИЛОЖЕНИЕ Д

Метод получения расписания преподавателя

Листинг 9: Реализация предложенной схемы, возможен неправильный порядок пробельных и специальных символов

```
1 function getData(html, callback) {
2     const $ = cheerio.load(html);
3     let resp = [];
4     $("#results > div > a").each(function () {
5         let name = $(this).text();
6         let path = $(this).attr("href");
7         let temp = {
8             name: name,
9             path: path
10        };
11        resp.push(temp);
12    });
13    return callback(null, resp);
14 }
15
16 function get(lastName, callback) {
17     Nightmare({
18         show: false
19     })
20         .goto(url)
21         .wait("body")
22         .type("#schedule_page > div > div.panes_item.
panes_item__type_teacher > div.teacher-form > input[type=text]:nth-child
(1)", lastName)
23         .click("#schedule_page > div > div.panes_item.
panes_item__type_teacher > div.teacher-form > input[type=button]:nth-
child(2)")
24         .wait(1500)
25         .evaluate(() => document.querySelector('body').innerHTML)
26         .end()
27         .then(response => {
28             getData(response, (err, res) => {
29                 return callback(null, res);
30             });
31         })
32         .catch(err => {
33             return callback(err);
34         });
35 }
```

ПРИЛОЖЕНИЕ Е

Реализация сцены получения расписания преподавателя

Листинг 10: Реализация предложенной схемы, возможен неправильный порядок пробельных и специальных символов

```
1  const Stage = require("../node_modules/telegraf/stage");           //
   не помню
2  const WizardScene = require("../node_modules/telegraf/scenes/wizard"); //
   Многоуровневые вопросы
3  const { enter, leave } = Stage;                                     // Не помню
4  const Markup = require('../node_modules/telegraf/markup');         //
   Настройки клавиатуры, форматирования текста и тп
5
6  const Nightmare = require('nightmare');
7  const cheerio = require('cheerio');
8  const url = "https://www.sgu.ru/schedule";
9
10 const { read } = require("../helpers/read");
11
12 const weekdays = new Array(['Воскресенье'],
13   ['Понедельник'], ['Вторник'], ['Среда'],
14   ['Четверг'], ['Пятница'], ['Суббота']);
15 let date = new Date();
16
17 function getData(html, callback) {
18   const $ = cheerio.load(html);
19   let resp = [];
20   $("#results > div > a").each(function () {
21     let name = $(this).text();
22     let path = $(this).attr("href");
23     let temp = {
24       name: name,
25       path: path
26     };
27     resp.push(temp);
28   });
29   return callback(null, resp);
30 }
31
32 function get(lastName, callback) {
33   Nightmare({
34     show: false
35   })
36     .goto(url)
37     .wait("body")
38     .type("#schedule_page > div > div.panes_item.
   panes_item__type_teacher > div.teacher-form > input[type=text]:nth-child
```

```

(1)", lastName)
39     .click("#schedule_page > div > div.panes_item.
panes_item__type_teacher > div.teacher-form > input[type=button]:nth-
child(2)")
40     .wait(1500)
41     .evaluate(() => document.querySelector('body').innerHTML)
42     .end()
43     .then(response => {
44         getData(response, (err, res) => {
45             return callback(null, res);
46         });
47     })
48     .catch(err => {
49         return callback(err);
50     });
51 }
52
53 const professor = new WizardScene(
54     "professor",
55     (ctx) => {
56         ctx.reply("Введите фамилию преподавателя", Markup.removeKeyboard(true).
oneTime().resize().extra());
57         return ctx.wizard.next();
58     },
59     (ctx) => {
60         ctx.wizard.state.lastName = ctx.message.text;
61         get(ctx.wizard.state.lastName, async (err, res) => {
62             ctx.wizard.state.res = res;
63             let names = [];
64             res.map(professor => {
65                 names.push(professor.name);
66             });
67             if (names.length == 0) {
68                 await ctx.reply("Поиск не дал результатов" +
69                     "\Пожалуйста, введите фамилию заново", Markup.removeKeyboard
(true).oneTime().resize().extra());
70                 ctx.reply("Введите фамилию преподавателя");
71                 return ctx.wizard.selectStep(1);
72             }
73             if (names.length > 15) {
74                 await ctx.reply("Поиск дал большое число результатов (" + names.
length + ") " +
75                     "\Пожалуйста, уточните фамилию преподавателя", Markup.
removeKeyboard(true).oneTime().resize().extra());
76                 ctx.reply("Введите фамилию преподавателя");
77                 return ctx.wizard.selectStep(1);
78             }

```



```

79         ctx.reply("Выберите преподавателя", Markup.keyboard(names).resize().
extra());
80         return ctx.wizard.next();
81     });
82 },
83     async (ctx) => {
84         let names = [];
85         let paths = [];
86         ctx.wizard.state.res.map(professor => {
87             names.push(professor.name);
88             paths.push("https://www.sgu.ru" + professor.path);
89         });
90
91         if (names.includes(ctx.message.text) == false) {
92             ctx.reply("Введена неверная фамилия.\Сделайте выбор заново");
93             return ctx.wizard.selectStep(2);
94         }
95         await ctx.reply('Вы выбрали "' + ctx.message.text + '"', Markup.
removeKeyboard(true).oneTime().resize().extra());
96         let index = names.indexOf(ctx.message.text);
97
98         let weekday = weekdays[date.getDay()][0];
99
100        if (weekday != "Воскресенье") {
101            await ctx.reply("Вывод пар на " + weekday);
102            await read(paths[index], weekday, (err, res) => {
103                if (res === "Сегодня пар нет" || res.length === 0) ctx.reply("
Сегодня пар нет");
104                else
105                    for (let index = 0; index < res.length; index++) {
106                        const element = res[index];
107                        ctx.replyWithHTML(element);
108                    }
109            });
110        }
111        else {
112            await ctx.reply("Так как сегодня Воскресенье, вывод пар на Понедельник")
;
113            await read(paths[index], "Понедельник", (err, res) => {
114                if (res === "Сегодня пар нет") ctx.reply("Сегодня пар нет");
115                else
116                    for (let index = 0; index < res.length; index++) {
117                        const element = res[index];
118                        ctx.replyWithHTML(element);
119                    }
120            });
121        }
122        setTimeout(() => {

```

```
123         ctx.reply("Выбрать другого преподавателя?", Markup.keyboard(['Да', '
Нет'])).oneTime().resize().extra());
124     }, 10000);
125
126     return ctx.wizard.next();
127 },
128 async (ctx) => {
129     let answer = ctx.message.text;
130     if (answer === "Да") {
131         ctx.reply("Введите фамилию преподавателя");
132         return ctx.wizard.selectStep(1);
133     }
134     else return ctx.scene.enter('def');
135 }
136 );
137
138 module.exports = { professor };
```

ПРИЛОЖЕНИЕ Ж

Реализация главной сцены

Листинг 11: Реализация главной сцены, возможен неправильный порядок пробельных и специальных символов

```
1 const Markup = require('../node_modules/telegraf/markup');
2 const WizardScene = require("../node_modules/telegraf/scenes/wizard");
3 const Stage = require("../node_modules/telegraf/stage");
4 const { enter, leave } = Stage;
5
6
7 const def = new WizardScene('def',
8   async (ctx) => {
9     leave("professor");
10    leave("student");
11    ctx.wizard.state.anss = ["Группы", "Преподавателя"];
12    ctx.reply("Вы хотите найти расписание", Markup.keyboard(ctx.wizard.state
13      .anss).resize().extra());
14    return ctx.wizard.next();
15  },
16  async (ctx) => {
17    if (ctx.message.text === "/help") {
18      await ctx.reply(`
19      Список доступных команд:
20      • /start - начать диалог с ботом
21      • /help - вывод доступных команд
22      `);
23      leave('def');
24      return ctx.scene.enter('def');
25    }
26    if (ctx.message.text === "/start") {
27      await ctx.reply("Здравствуйте, " + ctx.from.first_name + ", вас
28      приветствует " +
29      "бот расписания СГУ!");
30      leave('def');
31      return ctx.scene.enter('def');
32    }
33    if (ctx.message.text.match(Группы(/i)) {
34      leave('def');
35      return ctx.scene.enter("student");
36    }
37    if (ctx.message.text.match(Преподавателя(/i)) {
38      leave('def');
39      return ctx.scene.enter('professor');
```

```
40         await ctx.reply('"' + ctx.message.text + '" - не распознанная
команда');
41         leave('def');
42         return ctx.scene.enter('def');
43     }
44
45 });
46 module.exports = { def };
```