ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Факультет компьютерных наук
Департамент программной инженерии

СОГЛАСОВАНО
Доцент
департамента анализа данных и
искусственного интеллекта

_____ А. А. Незнанов
«___» _____ 2020 г.

УТВЕРЖДАЮ
Академический руководитель
образовательной программы
«Программная инженерия»

_____ В.В. Шилов
«___» _____ 2020 г.

Инв. № подл	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата
RU.17701729.04.05-01 12 01-1				

ПРОГРАММА РЕАЛИЗУЮШАЯ ВОПРОСНО-ОТВЕТНУЮ ЭКСПЕРТНУЮ СИСТЕМУ В ОБЛАСТИ ОБРАЗОВАНИЯ НА ПЛАТФОРМЕ MICROSOFT AZURE

Текст программы

ЛИСТ УТВЕРЖДЕНИЯ

RU.17701729.04.05-01 12 01-1-ЛУ

Исполнитель:
студент группы БПИ192
_____ / Жупанов В. /
«____»_____ 2020 г.

2020

УТВЕРЖДЕНО
RU.17701729.04.05-01 12 01-1-ЛУ

ПРОГРАММА РЕАЛИЗУЮШАЯ ВОПРОСНО-ОТВЕТНУЮ ЭКСПЕРТНУЮ СИСТЕМУ В ОБЛАСТИ ОБРАЗОВАНИЯ НА ПЛАТФОРМЕ MICROSOFT AZURE

Текст программы

RU.17701729.507140-01 12 01-1

Листов 20

Инв. № подл	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата
RU.17701729.04.05-01 12 01-1				

2020

Аннотация

В данном документе приведет текст «Программа реализующая вопрос-ответную экспертную систему в области образования на платформе Microsoft Azure». Программа разработана на языке C# 7.0. Среда разработки - Microsoft Visual Studio 2019.

Функциональным назначением программы является ответ на пользовательские запросы на основе базы знаний, а также реализация прохождения тестирования по полученным знаниям.

Настоящий документ разработан в соответствии с требованиями:

1) ГОСТ 19.101-77 Виды программ и программных документов [1];
2) ГОСТ 19.102-77 Стадии разработки [2];
3) ГОСТ 19.103-77 Обозначения программ и программных документов [3];
4) ГОСТ 19.104-78 Основные надписи [4];
5) ГОСТ 19.105-78 Общие требования к программным документам [5];
6) ГОСТ 19.106-78 Требования к программным документам, выполненным печатным способом[6];
7) ГОСТ 19.401-78 Текст программы. Требования к содержанию и оформлению [7].

Изменения к данному документу оформляются согласно ГОСТ 19.603-78 [8], ГОСТ 19.604-78 [9].

Перед прочтением данного документа рекомендуется ознакомиться с терминологией, приведенной в Приложении 1.

СОДЕРЖАНИЕ

1 Текст программы

Текст программы загружен в репозиторий и его можно посмотреть по ссылке: https://github.com/daniil024/qnabotTsurcanZhupanov

Программа состоит из 5 модулей: Bots, Comands, Dialog, Controllers, Images.

1.1 Код модуля Bots

1.1.1 Код класса QnABot

```
using System.Collections.Generic;
using System.Threading;
using System.Threading.Tasks;
using Microsoft.Bot.Builder;
using Microsoft.Bot.Builder.Dialogs;
using Microsoft.Bot.Schema;
using QnABot.Bots;
using QnABot.Comands;
using QnABot.Images;
using QnABot.SystemMessages;
using QnABot.Users;

namespace Microsoft.BotBuilderSamples.Bots
{
    public class QnABot<T> : ActivityHandler where T :
Microsoft.Bot.Builder.Dialogs.Dialog
    {
        protected readonly BotState ConversationState;  // Defines a state management
object
        protected readonly Microsoft.Bot.Builder.Dialogs.Dialog Dialog;  // Object of
Base class for all Dialogs
        protected readonly BotState UserState;  // Defines a state management object
        protected SendImages send_image = new SendImages();  // Object creation
        protected Start start = new Start();  // Object, which stores info about Commands
        protected UserInfo current_user;  // Object that gives us info about current user

        public QnABot(ConversationState conversationState, UserState userState, T dialog)
        {
            ConversationState = conversationState;
            UserState = userState;
            Dialog = dialog;
        }

        /// <summary>
        /// Method handles an incoming Activity from user
        /// </summary>
        /// <param name="turnContext"></param>
        /// <param name="cancellationToken"></param>
        /// <returns>Activity</returns>
        public override async Task OnTurnAsync(ITurnContext turnContext,
CancellationToken cancellationToken = default)
        {
            await base.OnTurnAsync(turnContext, cancellationToken);
            // Save any state changes that might have occured during the turn.
```

```
            await ConversationState.SaveChangesAsync(turnContext, false,
cancellationToken);
            await UserState.SaveChangesAsync(turnContext, false, cancellationToken);
        }

        protected override async Task
OnMessageActivityAsync(ITurnContext<IMessageActivity> turnContext, CancellationToken
cancellationToken)
        {
            //check what type of message was sent
            if (turnContext.Activity.Text == null)
                await
turnContext.SendActivityAsync(MessageFactory.Text(Message.MessageType_Error),
cancellationToken);
            else
            {
                RestartUser(turnContext, cancellationToken);
                //user definitions
                foreach (var item in Users.UsersList)
                    if (item.Id == turnContext.Activity.Recipient.Id) current_user =
item;
                //checking the executable command
                CheckCommands(turnContext, current_user);
                //testing process implementation
                if (Check.Was_test)
                {
                    //sending the correct answer
                    if (turnContext.Activity.Text.ToLower() == "/prompt")
                    {
                        await send_image.SendRightReply(turnContext, cancellationToken,
current_user);
                        await send_image.SendImageAsync(turnContext, cancellationToken,
current_user);
                        //check if the user has passed all the questions
                        if (!SendImages.Was_ended)
                            await send_image.SendButtonAsync(turnContext,
cancellationToken, current_user);
                        SendImages.Was_ended = false;
                    }
                    else
                    {
                        //validation of the answer
                        if (!SendImages.Was_answer)
                            await send_image.CheckReplyAsync(turnContext,
cancellationToken, current_user);
                        //sending the next question if the previous one was correct
                        if (SendImages.Right_answer)
                        {
                            await send_image.SendImageAsync(turnContext,
cancellationToken, current_user);
                            if (!SendImages.Was_ended)
                                await send_image.SendButtonAsync(turnContext,
cancellationToken, current_user);
                            SendImages.Was_answer = false;
                            SendImages.Was_ended = false;
                        }
                    }
```

```
                }
                //sending a response from the knowledge base
                else
                {
                    if (turnContext.Activity.Text.ToLower() == "/prompt")
                        await
turnContext.SendActivityAsync(MessageFactory.Text(Message.Prompt_Error),
cancellationToken);
                    else
                        await Dialog.RunAsync(turnContext,
ConversationState.CreateProperty<DialogState>(nameof(DialogState)), cancellationToken);
                }
            }
        }

        /// <summary>
        /// Method greets new users
        /// </summary>
        /// <param name="membersAdded"></param>
        /// <param name="turnContext">activity received</param>
        /// <param name="cancellationToken"></param>
        /// <returns>Activity(Task)</returns>
        protected override async Task OnMembersAddedAsync(IList<ChannelAccount>
membersAdded, ITurnContext<IConversationUpdateActivity> turnContext, CancellationToken
cancellationToken)
        {
            foreach (var member in membersAdded)
            {
                if (member.Id != turnContext.Activity.Recipient.Id)
                {
                    await
turnContext.SendActivityAsync(MessageFactory.Text(start.Welcome_message),
cancellationToken);
                    //adding a new user to the user list
                    Users.AddUser(new UserInfo(turnContext.Activity.Recipient.Id));
                }
            }
        }

        /// <summary>
        /// Bot starts new dialog with the user
        /// </summary>
        /// <param name="turnContext">activity received</param>
        /// <param name="cancellationToken"></param>
        private void RestartUser(ITurnContext<IMessageActivity> turnContext,
CancellationToken cancellationToken)
        {
            if (turnContext.Activity.Text.ToLower() == "/start")
            {
                int delete_user = -1;
                //search user in list
                for (int i = 0; i < Users.UsersList.Count; i++)
                    if (Users.UsersList[i].Id == turnContext.Activity.Recipient.Id)
delete_user = i;
                //delete user from list
                if (delete_user != -1)
Users.UsersList.Remove(Users.UsersList[delete_user]);
```

```
                //add user from list
                Users.AddUser(new UserInfo(turnContext.Activity.Recipient.Id));
                Check.Was_test = false;
                turnContext.SendActivityAsync(MessageFactory.Text(start.Welcome_message),
cancellationToken);
            }
        }

        /// <summary>
        /// check if a command was entered
        /// </summary>
        /// <param name="turnContext">activity received</param>
        /// <param name="user">current user</param>
        private void CheckCommands(ITurnContext turnContext, UserInfo user)
        {
            if (turnContext.Activity.Text.ToLower() == "/test")
            {
                Check.Was_test = true;
                send_image = new SendImages();
            }
            if (turnContext.Activity.Text.ToLower() == "/answer")
            {
                Check.Was_test = false;
                SendImages.Was_answer = true;
                SendImages.Right_answer = true;
                if (user.Current != 0)
                    user.Current--;
            }
        }
    }
}
```

1.1.2 Код класса Check

```
namespace QnABot.Bots
{
    public static class Check
    {
        public static bool Was_test { get; set; }
    }
}
```

1.1.3 Код класса Startup

```
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Bot.Builder;
using Microsoft.Bot.Builder.Integration.AspNet.Core;
using Microsoft.BotBuilderSamples.Bots;
using Microsoft.BotBuilderSamples.Dialog;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;

namespace Microsoft.BotBuilderSamples
{
    public class Startup
```

```
    {
        public Startup(IConfiguration configuration)
        {
            Configuration = configuration;
        }

        public IConfiguration Configuration { get; }

        // This method gets called by the runtime. Use this method to add services to the
container.
        public void ConfigureServices(IServiceCollection services)
        {
            services.AddMvc().SetCompatibilityVersion(CompatibilityVersion.Version_2_1);

            // Create the Bot Framework Adapter with error handling enabled.
            services.AddSingleton<IBotFrameworkHttpAdapter, AdapterWithErrorHandler>();

            // Create the bot services(QnA) as a singleton.
            services.AddSingleton<IBotServices, BotServices>();

            // Create the storage we'll be using for User and Conversation state. (Memory
is great for testing purposes.)
            services.AddSingleton<IStorage, MemoryStorage>();

            // Create the User state. (Used in this bot's Dialog implementation.)
            services.AddSingleton<UserState>();

            // Create the Conversation state. (Used by the Dialog system itself.)
            services.AddSingleton<ConversationState>();

            // The Dialog that will be run by the bot.
            services.AddSingleton<RootDialog>();

            // Create the bot as a transient. In this case the ASP Controller is
expecting an IBot.
            services.AddTransient<IBot, QnABot<RootDialog>>();
        }

        // This method gets called by the runtime. Use this method to configure the HTTP
request pipeline.
        public void Configure(IApplicationBuilder app, IHostingEnvironment env)
        {
            if (env.IsDevelopment())
            {
                app.UseDeveloperExceptionPage();
            }
            else
            {
                app.UseHsts();
            }

            app.UseDefaultFiles();
            app.UseStaticFiles();

            app.UseWebSockets();
            app.UseMvc();
        }
```

```
    }
}
```

1.1.4 Код класса IBotServices

```
using Microsoft.Bot.Builder.AI.QnA;

namespace Microsoft.BotBuilderSamples
{
    public interface IBotServices
    {
        QnAMaker QnAMakerService { get; }
    }
}
```

1.1.5 Код класса BotServices

```
using Microsoft.Bot.Builder.AI.QnA;
using Microsoft.Extensions.Configuration;

namespace Microsoft.BotBuilderSamples
{
    public class BotServices : IBotServices
    {

        /// <summary>
        /// Constructor create new instance of the QnAMaker class
        /// </summary>
        /// <param name="configuration"></param>
        public BotServices(IConfiguration configuration)
        {
            QnAMakerService = new QnAMaker(new QnAMakerEndpoint
            {
                KnowledgeBaseId = configuration["QnAKnowledgebaseId"],
                EndpointKey = configuration["QnAAuthKey"],
                Host = GetHostname(configuration["QnAEndpointHostName"])
            });
        }

        public QnAMaker QnAMakerService { get; private set; }

        /// <summary>
        /// Method return name of the right host
        /// </summary>
        /// <param name="hostname"></param>
        /// <returns>hostname</returns>
        private static string GetHostname(string hostname)
        {
            if (!hostname.StartsWith("https://"))
            {
                hostname = string.Concat("https://", hostname);
            }

            if (!hostname.EndsWith("/qnamaker"))
            {
                hostname = string.Concat(hostname, "/qnamaker");
```

```
            }

            return hostname;
        }
    }
}
```

1.1.6 Код класса Messages

```
namespace QnABot.SystemMessages
{
    public class Message
    {
        public const string default_message = "Пока я не могу ответить вам на этот
вопрос, но в скором времени обещаю исправиться!";

        public static string Prompt_Error { get => "Данная команда доступна только в
разделе тестирования!"; }

        public static string MessageType_Error { get => "К сожалению я пока поддерживаю
только текстовые запросы"; }
    }
}
```

1.2 Код модуля Comands

1.2.1 Код класса ITool

```
namespace QnABot.Tools
{
    interface ITool
    {
        string Description { get; set; }
        string CommandsName { get; set; }
    }
}
```

1.2.2 Код класса Answer

```
using QnABot.Tools;

namespace QnABot.Comands
{
    public class Answers : ITool
    {
        public string Description { get; set; }
        public string CommandsName { get; set; }

        public Answers()
        {
            Description = "зададай мне вопрос и получи полезный ответ";
            CommandsName = "/answer";
        }
    }
}
```

1.2.3 Код класса Start

```
using QnABot.Tools;

namespace QnABot.Comands
{
    public class Start : ITool
    {
        protected Test test = new Test();
        protected Answers answers = new Answers();
        protected Prompt prompt = new Prompt();

        public string Description { get; set; }
        public string CommandsName { get; set; }

        public string Welcome_message { get => "Здравствуйте, уважаемый
пользователь!\n\nЯ - бот, который призван помочь Вам. На данный момент я могу:\n\n" +
                $"{answers.CommandsName} - {answers.Description}\n\n{test.CommandsName} -
{test.Description}\n\n"+
                $"{prompt.CommandsName} - {prompt.Description}\n\n{CommandsName} -
{Description}"; }

        public Start()
        {
            Description = "начать работу с ботом";
            CommandsName = "/start";
        }
    }
}
```

1.2.4 Код класса Prompt

```
using QnABot.Tools;

namespace QnABot.Comands
{
    public class Prompt : ITool
    {
        public string Description { get; set; }
        public string CommandsName { get; set; }

        public Prompt()
        {
            Description = "помощь при затруднении в процессе тестирования";
            CommandsName = "/prompt";
        }
    }
}
```

1.2.5 Код класса Test

```
using QnABot.Tools;

namespace QnABot.Comands
{
```

```
    public class Test : ITool
    {
        public string Description { get; set; }
        public string CommandsName { get; set; }

        public Test()
        {
            Description = "реализация тестирования по теоретической части
программирования";
            CommandsName = "/test";
        }
    }
}
```

1.3 Код модуля Dialog

1.3.1 Код класса RootDialog

```
using System.Collections.Generic;
using System.Threading;
using System.Threading.Tasks;
using Microsoft.Bot.Builder.AI.QnA;
using Microsoft.Bot.Builder.AI.QnA.Dialogs;
using Microsoft.Bot.Builder.Dialogs;

namespace Microsoft.BotBuilderSamples.Dialog
{
    /// <summary>
    /// This is an example root dialog. Replace this with your applications.
    /// </summary>
    public class RootDialog : ComponentDialog
    {
        /// <summary>
        /// QnA Maker initial dialog
        /// </summary>
        private const string InitialDialog = "initial-dialog";

        /// <summary>
        /// Initializes a new instance of the <see cref="RootDialog"/> class.
        /// </summary>
        /// <param name="services">Bot Services.</param>
        public RootDialog(IBotServices services)
            : base("root")
        {
            AddDialog(new QnAMakerBaseDialog(services));

            AddDialog(new WaterfallDialog(InitialDialog)
               .AddStep(InitialStepAsync));

            // The initial child Dialog to run.
            InitialDialogId = InitialDialog;
        }

        /// <summary>
        /// Method create and return new dialog that base on Waterfall way
        /// </summary>
        /// <param name="stepContext"></param>
```

```
        /// <param name="cancellationToken"></param>
        /// <returns>Activity</returns>
        private async Task<DialogTurnResult> InitialStepAsync(WaterfallStepContext
stepContext, CancellationToken cancellationToken)
        {
            return await stepContext.BeginDialogAsync(nameof(QnAMakerDialog), null,
cancellationToken);
        }
    }
}
```

1.3.2 Код класса QnAMakerBaseDialog

```
using System.Threading.Tasks;
using Microsoft.Bot.Builder;
using Microsoft.Bot.Builder.AI.QnA;
using Microsoft.Bot.Builder.AI.QnA.Dialogs;
using Microsoft.Bot.Builder.Dialogs;
using Microsoft.Bot.Schema;
using QnABot.SystemMessages;

namespace Microsoft.BotBuilderSamples.Dialog
{
    /// <summary>
    /// QnAMaker action builder class
    /// </summary>
    public class QnAMakerBaseDialog : QnAMakerDialog
    {
        // Dialog Options parameters
        public const string DefaultNoAnswer = Message.default_message;
        public const string DefaultCardTitle = "Did you mean:";
        public const string DefaultCardNoMatchText = "None of the above.";
        public const string DefaultCardNoMatchResponse = "Thanks for the feedback.";

        private readonly IBotServices _services;

        /// <summary>
        /// Initializes a new instance of the <see cref="QnAMakerBaseDialog"/> class.
        /// Dialog helper to generate dialogs.
        /// </summary>
        /// <param name="services">Bot Services.</param>
        public QnAMakerBaseDialog(IBotServices services): base()
        {
            this._services = services;
        }


        /// <summary>
        /// Method returns property, which provide access to QnA Maker Base
        /// </summary>
        /// <param name="dc"></param>
        /// <returns>Activity</returns>
        protected async override Task<IQnAMakerClient>
GetQnAMakerClientAsync(DialogContext dc)
        {
            return this._services?.QnAMakerService;
        }
```

```
        /// <summary>
        /// Method returns Task<TResult>, which finish successfully with specified result
        /// </summary>
        /// <param name="dc"></param>
        /// <returns>Activity</returns>
        protected override Task<QnAMakerOptions> GetQnAMakerOptionsAsync(DialogContext
dc)
        {
            return Task.FromResult(new QnAMakerOptions
            {
                ScoreThreshold = DefaultThreshold,
                Top = DefaultTopN,
                QnAId = 0,
                RankerType = "Default",
                IsTest = false
            });
        }

        /// <summary>
        /// Method handles Dialog Options parameters
        /// </summary>
        /// <param name="dc"></param>
        /// <returns>Activity</returns>
        protected async override Task<QnADialogResponseOptions>
GetQnAResponseOptionsAsync(DialogContext dc)
        {
            var noAnswer = (Activity)Activity.CreateMessageActivity();
            noAnswer.Text = DefaultNoAnswer;

            var cardNoMatchResponse =
(Activity)MessageFactory.Text(DefaultCardNoMatchResponse);


            var responseOptions = new QnADialogResponseOptions
            {
                ActiveLearningCardTitle = DefaultCardTitle,
                CardNoMatchText = DefaultCardNoMatchText,
                NoAnswer = noAnswer,
                CardNoMatchResponse = cardNoMatchResponse,
            };

            return responseOptions;
        }
    }
}
```

1.4 Код модуля Controllers

1.4.1 Код класса BotController

```
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Bot.Builder;
using Microsoft.Bot.Builder.Integration.AspNet.Core;
```

```
namespace Microsoft.BotBuilderSamples
{
    // This ASP Controller is created to handle a request. Dependency Injection will
provide the Adapter and IBot
    // implementation at runtime. Multiple different IBot implementations running at
different endpoints can be
    // achieved by specifying a more specific type for the bot constructor argument.
    [Route("api/messages")]
    [ApiController]
    public class BotController : ControllerBase
    {
        private readonly IBotFrameworkHttpAdapter _adapter;
        private readonly IBot _bot;

        public BotController(IBotFrameworkHttpAdapter adapter, IBot bot)
        {
            _adapter = adapter;
            _bot = bot;
        }

        [HttpGet, HttpPost]
        public async Task PostAsync()
        {
            // Delegate the processing of the HTTP POST to the adapter.
            // The adapter will invoke the bot.
            await _adapter.ProcessAsync(Request, Response, _bot);
        }
    }
}
```

1.4.2 Код класса AdapterWithErrorHandler

```
using System;
using Microsoft.Bot.Builder;
using Microsoft.Bot.Builder.Integration.AspNet.Core;
using Microsoft.Bot.Builder.TraceExtensions;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.Logging;

namespace Microsoft.BotBuilderSamples
{
    public class AdapterWithErrorHandler : BotFrameworkHttpAdapter
    {

        /// <summary>
        /// OnTurnError handler gets all Exceptions, which were throwed in conformity
with the bot's step logic.
        /// If Exception was throwed, then handler delete state of current dialog.
        /// </summary>
        /// <param name="configuration"></param>
        /// <param name="logger"></param>
        /// <param name="conversationState"></param>
        public AdapterWithErrorHandler(IConfiguration configuration,
ILogger<BotFrameworkHttpAdapter> logger, ConversationState conversationState = null)
            : base(configuration, logger)
        {
            OnTurnError = async (turnContext, exception) =>
```

```
            {
                // Log any leaked exception from the application.
                logger.LogError(exception, $"[OnTurnError] unhandled error :
{exception.Message}");

                if (conversationState != null)
                {
                    try
                    {
                        // Delete the conversationState for the current conversation to
prevent the
                        // bot from getting stuck in a error-loop caused by being in a
bad state.
                        // ConversationState should be thought of as similar to "cookie-
state" in a Web pages.
                        await conversationState.DeleteAsync(turnContext);
                    }
                    catch (Exception e)
                    {
                        logger.LogError(e, $"Exception caught on attempting to Delete
ConversationState : {e.Message}");
                    }
                }

                // Send a trace activity, which will be displayed in the Bot Framework
Emulator
                await turnContext.TraceActivityAsync("OnTurnError Trace",
exception.Message, "https://www.botframework.com/schemas/error", "TurnError");
            };
        }
    }
}
```

1.5 Код модуля Image

1.5.1 Код класса Parser

```
using System.Collections.Generic;
using System.Xml;

namespace QnABot.Images
{
    public class Parser
    {
        string path = @"C:\images_info.xml";
        XmlDocument xDoc = new XmlDocument();
        List<ImageProperties> images = new List<ImageProperties>();
        public List<ImageProperties> Images => images;

        public Parser()
        {
            xDoc.Load(path);
            Parse();
        }

        private void Parse()
        {
```

```
            XmlElement xEl = xDoc.DocumentElement;
            foreach (XmlNode xnode in xEl)
            {
                ImageProperties new_image = new ImageProperties();
                //read ittributes
                new_image.Number = xnode.Attributes["number"].Value;
                //read child nodes
                foreach (XmlNode childnode in xnode.ChildNodes)
                {
                    if (childnode.Name == "imagePath")
                    {
                        new_image.Image_path = childnode.InnerText;
                    }
                    if (childnode.Name == "webPath")
                    {
                        new_image.Web_path = childnode.InnerText;
                    }
                    if (childnode.Name == "rightAnswer")
                    {
                        new_image.Right_answer = childnode.InnerText;
                    }
                }
                images.Add(new_image);
            }
        }
    }
}
```

1.5.2 Код класса ImageException

```
using System;

namespace QnABot.Images
{
    [Serializable]
    public class ImageException : Exception
    {
        public ImageException() { }
        public ImageException(string message) : base(message) { }
        public ImageException(string message, Exception inner) : base(message, inner) { }
        protected ImageException(
          System.Runtime.Serialization.SerializationInfo info,
          System.Runtime.Serialization.StreamingContext context) : base(info, context) {
}
    }
}
```

1.5.3 Код класса ImageProperties

```
namespace QnABot.Images
{
    public class ImageProperties
    {
        public string Image_path { get; set; }
        public string Web_path { get; set; }
        public string Number { get; set; }
```

```
        public string Right_answer { get; set; }
    }
}
```

1.5.4 Код класса Users

```
using System.Collections.Generic;

namespace QnABot.Users
{
    public class Users
    {
        static List<UserInfo> users = new List<UserInfo>();
        public static List<UserInfo> UsersList { get => users; }

        /// <summary>
        /// add new user to list
        /// </summary>
        /// <param name="user">new user</param>
        public static void AddUser(UserInfo user) => users.Add(user);

    }
}
```

1.5.5 Код класса UsersInfo

```
namespace QnABot.Users
{
    /// <summary>
    /// Class keeps information about users
    /// </summary>
    public class UserInfo
    {
        public string Id { get; private set; }
        public int Current { get; set; }  // Number of image, at which user stopped

        public UserInfo(string id)
        {
            Current = 0;
            Id = id;
        }
    }
}
```

ПРИЛОЖЕНИЕ 1

ТЕРМИНОЛОГИЯ

Таблица 1

Термин	Определение
Развертка бота	Загрузка приложения на Bot Framework в облако.
QnA Bot	Чат-бот, отвечающее на пользовательские вопросы в соответствии базе знаний.
Чат-бот	Веб-приложение, имитирующее реальное общение с пользователем.

ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

Лист регистрации изменений

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.05-01 12				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

Номера листов (страниц)					Всего листов (страниц в докум.)	№ документа	Входящий № сопроводительного докум. и дата	Подп.	Дата
Изм.	Измененных	Замененных	Новых	Аннулированх					

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.05-01 12				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата