**Integrazione della piattaforma EVA per AI-based chatbots con il Microsoft Bot Framework**

Design Requirement Specification Document

DIBRIS – Università di Genova. Scuola Politecnica, Corso di Ingegneria del Software 80154

DATA: 10/07/2020

VERSIONE: 1.0

**Autori**

Marco Scanu

Elisa Zilich

**REVISION HISTORY**

|  |  |  |  |
| --- | --- | --- | --- |
| Versione | Data | Autori | Note |
| 1.0 | 21/052020 | MS, EZ | Prima versione |
| 2.0 | 29/05/2020 | MS, EZ | Seconda versione |
| 3.0 | 19/06/2020 | MS, EZ | Terzo versione |
| 4.0 | 10/07/2020 | MS, EZ | Quarta versione |

Indice dei contenuti

[**1** **Introduzione** 3](#_Toc45235177)

[**1.1** **Definizioni e Acronimi** 3](#_Toc45235178)

[**1.2** **Overview del Documento** 3](#_Toc45235179)

[**1.3** **Bibliografia** 3](#_Toc45235180)

[**2** **Descrizione del progetto** 4](#_Toc45235181)

[**2.1** **Introduzione** 4](#_Toc45235182)

[**2.2** **Architettura del sistema** 4](#_Toc45235183)

[**2.3** **Vincoli e assunzioni** 4](#_Toc45235184)

[**3** **Interfacce** 5](#_Toc45235185)

[3.1 Modalità d’uso dell’interfaccia 5](#_Toc45235186)

[3.2 Customizzazione dell’interfaccia della chat 7](#_Toc45235187)

[**4** **Data Stores** 9](#_Toc45235188)

[**5** **Use Cases** 9](#_Toc45235189)

[**6** **Structural Design** 10](#_Toc45235190)

[**6.1** **Class Diagram** 10](#_Toc45235191)

[**6.1.1** **Descrizione delle classi** 10](#_Toc45235192)

[**7** **Dynamic Model** 13](#_Toc45235193)

[**7.1** **Diagramma delle macchine a stati per la modalità sviluppo:** 13](#_Toc45235194)

[**7.2** **Diagramma delle macchine a stati per la modalità produzione:** 13](#_Toc45235195)

# **1 Introduzione**

Questo documento specifica l’architettura di un sistema software per l’integrazione di EVA con i framework Microsoft. È stato redatto per app2check e tutti gli utenti amministratori/gestori che effettuano il training del chatbot in modalità sviluppo e il testing sia nella modalità sviluppo che produzione.

L'obiettivo del progetto, è quello di progettare e implementare un modulo di integrazione software utilizzando il componente open source "Bot Framework SDK" di Microsoft, che consente di gestire la logica dei bot nel back-end, e la piattaforma EVA. In particolare, si dovrà implementare un diagramma di flusso più generale possibile nel Bot Framework, cercando di integrare tutte le funzionalità di EVA già disponibili tramite API.

Successivamente si dovrà integrare la componente open source di Microsoft "Bot Framework Web Chat" che permette di visualizzare e gestire l'interfaccia utente del bot nel front-end. È rilevante che il front-end sia customizzabile attraverso il passaggio di opportuni parametri. La progettazione dell'integrazione dovrà tener conto anche di una UX fluida.

## **Definizioni e Acronimi**

|  |  |
| --- | --- |
| **Acronimo-Nome** | **Definizione** |
| EZ | Elisa Zilich |
| MS | Marco Scanu |
| KB | Knowledge Based |
| UX | User Experience |
| EVA | È una soluzione di Virtual Assistant/Chatbot progettata per dare supporto ai Customer Care, offrire un'esperienza di servizio intuitiva e senza problemi rispondere istantaneamente ed automaticamente alle richieste dei clienti. |

## **Overview del Documento**

Il documento è organizzato in sette sezioni principali:

* Introduzione
* Descrizione del progetto
* Interfacce
* Data Stores
* Use Cases
* Structural Design
* Dynamic Model

## **Bibliografia**

Presentazione PDF “EVA\_UserGuide-v.2.5” fornita da App2Check

Presentazione PDF “EVA API Documentation 28-08-2019 EN” fornita da App2Check

Documento URS

# **2 Descrizione del progetto**

## **2.1 Introduzione**

In questo progetto stiamo utilizzando un approccio object oriented tramite l’utilizzo del linguaggio C# con l’IDE Microsoft Visual Studio.

Per implementare i Class Diagram e gli Use Case Diagram verrà utilizzato il tool DIA.

## **2.2 Architettura del sistema**

L’architettura del sistema è la seguente:

Immagine che contiene screenshot

Descrizione generata automaticamente

Come si può notare, nell’architettura del sistema è presente il chatbot creato attraverso il Microsoft Bot Framework SDK che per ogni messaggio mandato in input dall’utente, tramite l’utilizzo delle apposite API, esegue una query in formato JSON che interroga il KB di un bot preesistente nella piattaforma EVA. Successivamente restituisce a MyBot una risposta, sempre in formato JSON.

Per poter utilizzare il framework web chat è stato creato un account Azure, tramite il quale vengono generati i campi ***MicrosoftAppID*** e ***MicrosoftAppPassword***. Tali valori, devono essere specificati nel file appsetting.json all’interno del bot in modo tale da permettere la pubblicazione del chatbot all’interno di Azure.

Infine, creando una pagina html che incorpora lo script di webchat.js è possibile visualizzare anche la parte grafica del bot.

## **2.3 Vincoli e assunzioni**

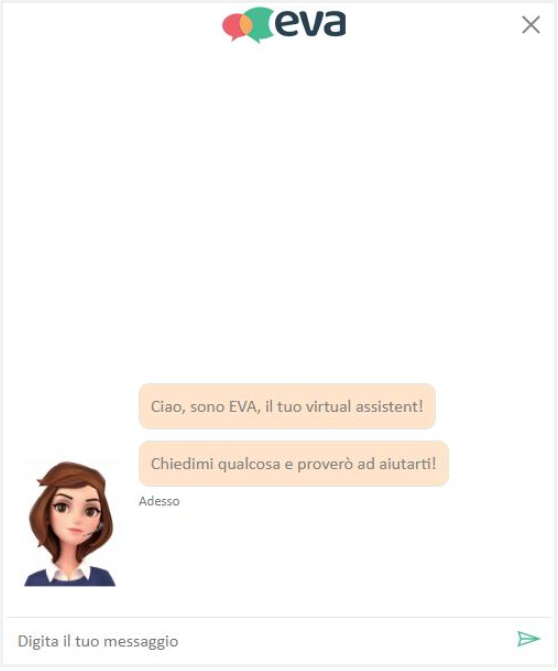
Non ci sono vincoli.

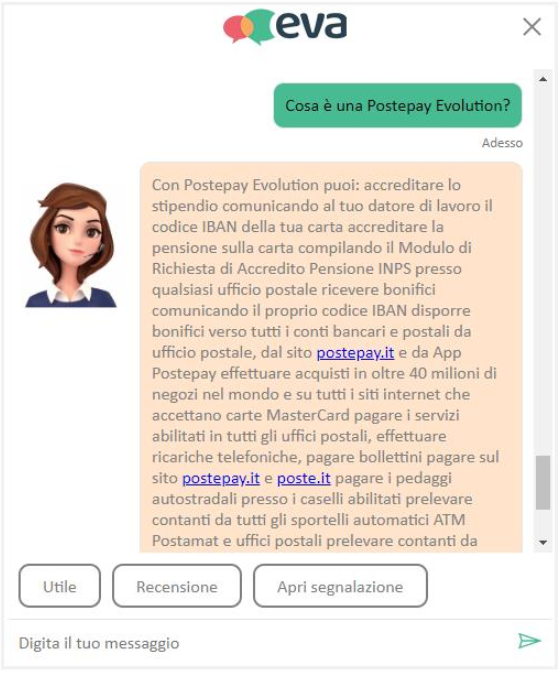
# **3 Interfacce**

## http://localhost/WebChat/immagini/eva.png3.1 Modalità d’uso dell’interfaccia

L’interfaccia si presenta con un pulsante con l’immagine di EVA che se cliccato apre la chat con il bot.

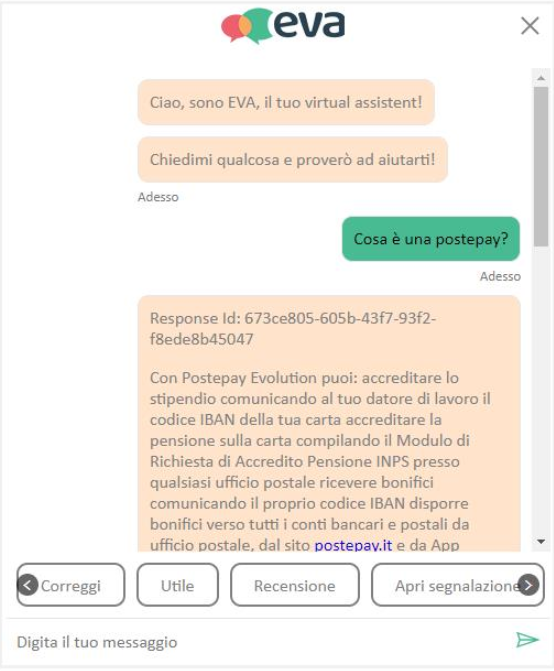
La struttura della chat è fornita da Microsoft Web Chat.

All’avvio del bot verrà visualizzato un messaggio di benvenuto:

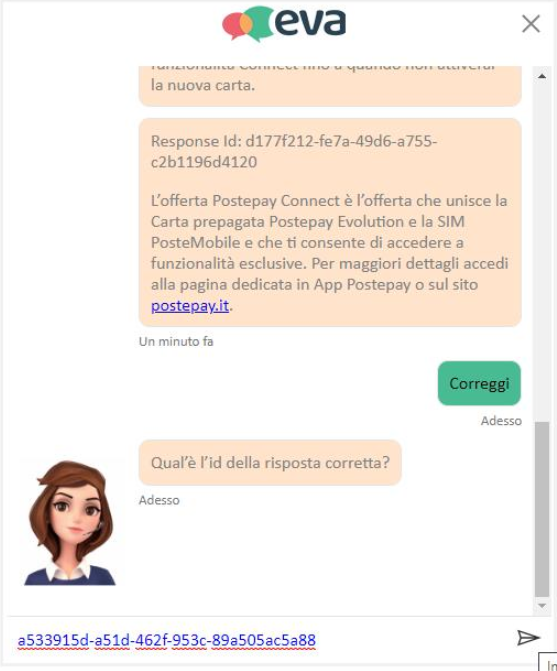
Ora il bot è in attesa che l’utente invii un messaggio:

Poiché il bot è settato in modalità produzione (isDev = false), ha risposto alla nostra domanda con un messaggio e tre action button:

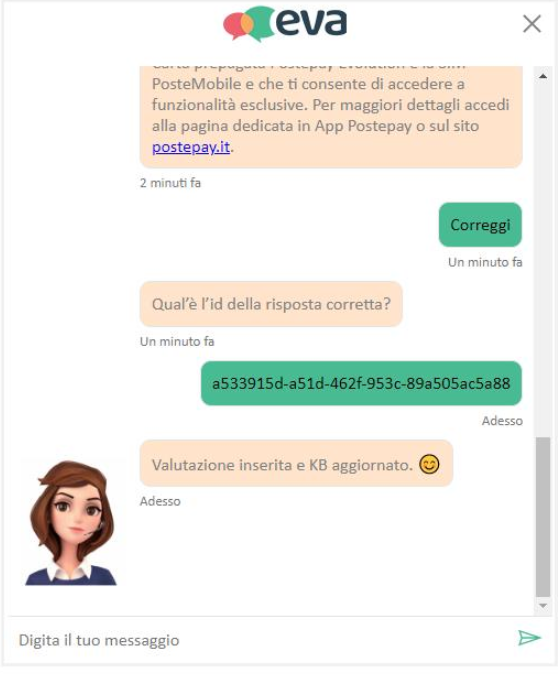
* Utile;
* Lasciami una recensione;
* Apri una segnalazione.

Se invece il bot è settato in modalità sviluppo (isDev = true), verrà visualizzato anche il pulsante “Correggi”:

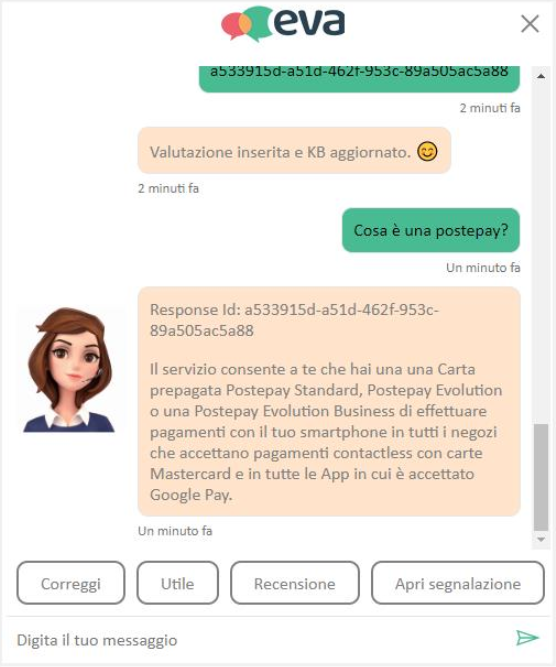
Facendo click su “Correggi”, si avrà modo di correggere la risposta data dal chatbot. Quest’ultimo chiederà di inserire l’id della risposta che si ritiene corretta all’interno del KB:

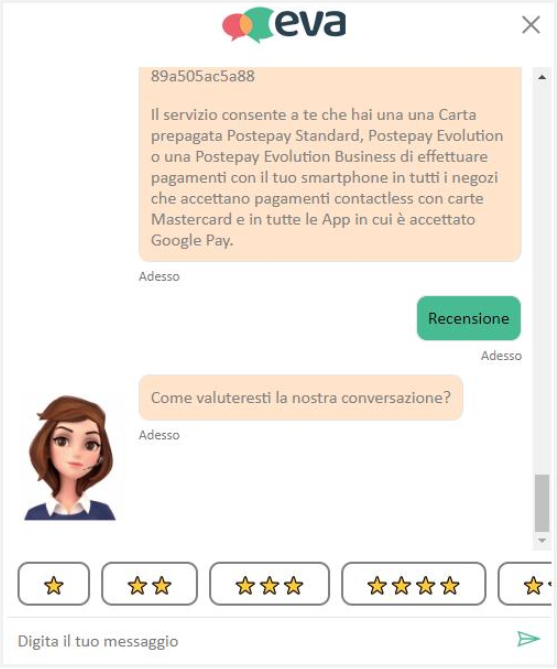


Una volta inserito l’id, il bot risponderà con “Valutazione inserita e KB aggiornato” e la domanda verrà associata a quella risposta:

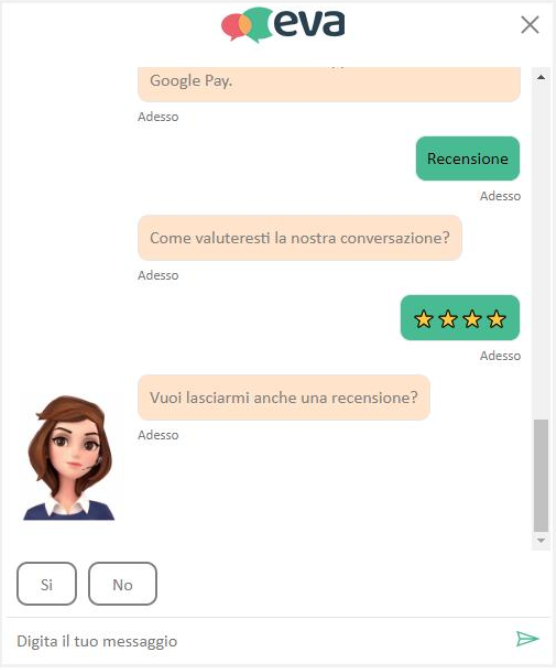


Infatti, riproponendo la stessa domanda, questa volta il bot risponderà correttamente:

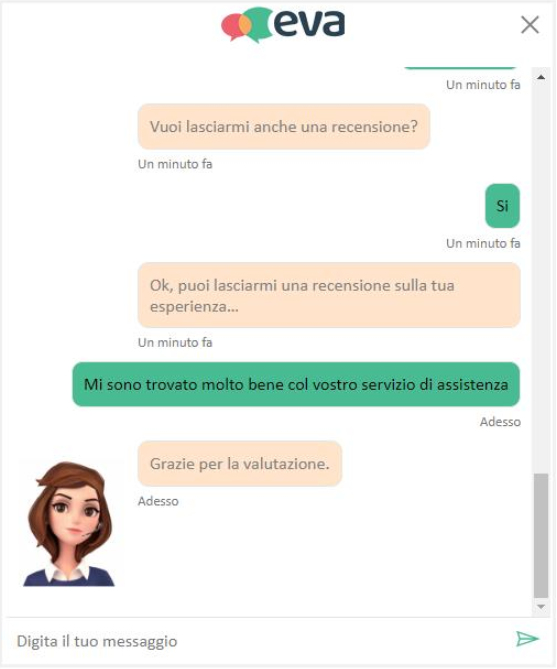


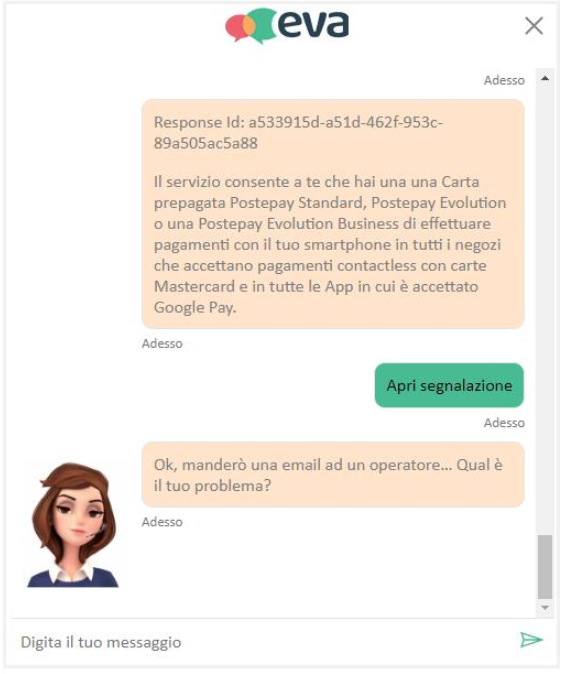
Facendo click sul pulsante “Recensione” oppure sul pulsante “Utile” è possibile valutare la risposta data dal bot:

La differenza tra le due funzionalità è che cliccando il pulsante “Utile” verrà considerato a fine statistico che quella risposta è effettivamente utile.

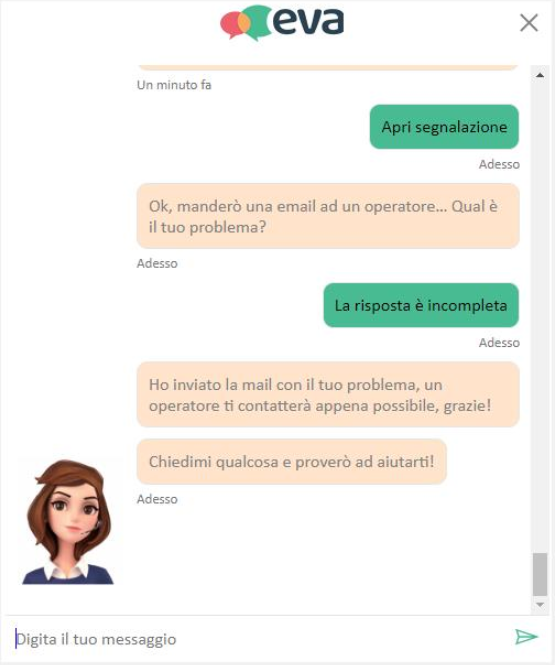
Cliccando su uno dei pulsanti per il voto, verrà valutata la risposta e il bot invierà un messaggio chiedendo se si vuole anche scrivere una recensione:

Una volta che viene inviata la recensione, il bot ringrazierà per la valutazione:



Facendo click sul pulsante “Apri segnalazione” è possibile segnalare un malfunzionamento agli sviluppatori:

Una volta che si scrive la segnalazione, il bot risponderà dicendo che invierà una mail con il problema ad un operatore e successivamente ringrazia.



## 3.2 Customizzazione dell’interfaccia della chat

Attraverso il passaggio di alcuni parametri nell’URL è possibile personalizzare a proprio piacimento la chat.

Elenco dei parametri:

Parametri che modificano la grafica:

***userBaloonColor***: colore delle vignette dell’utente.

***botBaloonColor***: colore delle vignette del bot.

***title***: indirizzo del logo del chatbot.

***sendBtnColor***: colore del bottone per l’invio del messaggio.

***serverUserImage*:** immagine usata come avatar se il parametro “video” è settato a “false”.

***botTextColor***: colore del testo nelle vignette del bot.

***botBtnColor***: colore delle suggested action.

***botBtnTextColor***: colore del testo delle suggested action.

***userTextColor***: colore del testo nelle vignette dell’utente.

***sendTextColor***: colore del testo nella barra di scrittura del messaggio.

***sendBackgrColor***: colore del background della barra di scrittura del messaggio.

Parametri funzionali:

**video**: valore booleano.

* Se è ***true***, utilizza come avatar la gif di eva per simulare il movimento del parlare;
* Se è ***false***, utilizza come avatar l’immagine passata nel campo dell’URL serverUserImage.

Se non viene settato, il suo valore di default è “true” perciò il bot utilizza la gif.

***speech***: valore booleano.

* Se è ***true***, c’è la possibilità di inviare un messaggio vocale al bot e in quel caso il bot risponderà leggendo il testo del messaggio di risposta;
* Se è ***false***, i messaggi tra il bot e l’utente saranno solo testuali.

Se non viene settato, il suo valore di default è “false.

***botId***: id del bot che si interroga.

***isDev***: valore booleano.

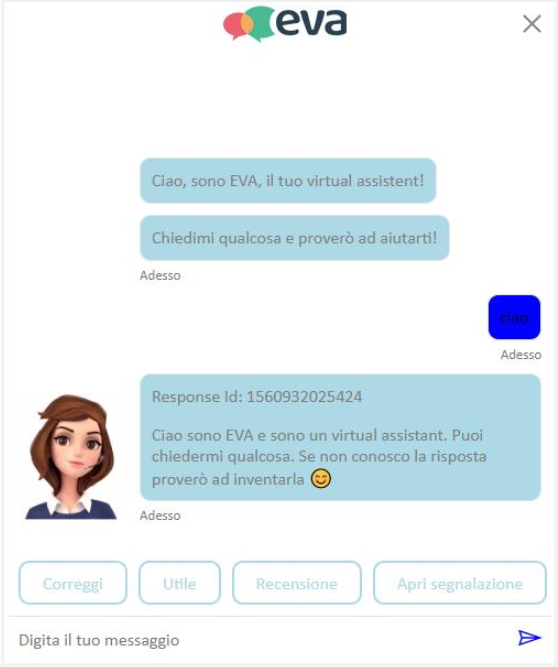
* Se è ***true****,* il bot viene avviato in modalità sviluppo;
* Se è ***false***, il bot viene avviato in modalità produzione.

Se non viene settato, il valore di default p “false” perciò il bot parte di default in modalità produzione.

***supportEmail***: specifica l’email associata al bot.

***testWebSite***: visualizza il chatbot come anteprima sul sito che viene passato come parametro.

In questo caso è stato cambiato il colore della vignetta dell’utente e del bot. Il colore si può specificare utilizzando il suo valore in esadecimale oppure il nome del colore stesso.

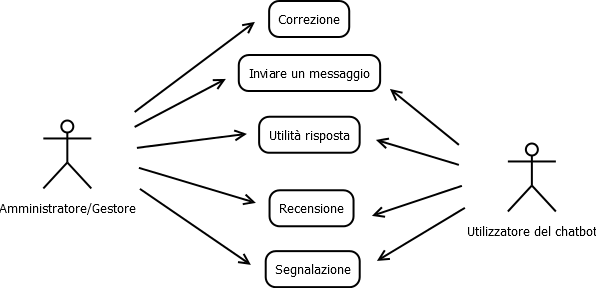


# **4 Data Stores**

Non utilizziamo un Data Base implementato direttamente da noi. La piattaforma EVA gestisce le interazioni con un suo DB.

# **5 Use Cases**

Le modalità di utilizzo del chatbot sono due:

1. La ***modalità sviluppo***: utilizzata solamente dall’utente amministratore/gestore;
2. La ***modalità produzione***: utilizzata da tutti gli stakeholder che interrogano il chatbot per ricevere delle risposte;

|  |  |
| --- | --- |
| **ID** | **USE CASE** |
| 1 | Correzione |
| 2 | Inviare un messaggio |
| 3 | Utilità risposta |
| 4 | Recensione |
| 5 | Segnalazione |

# **6 Structural Design**

## **6.1 Class Diagram**

### **6.1.1 Descrizione delle classi**

L’elenco delle classi utilizzate è il seguente:

#### **Program**

È la classe principale del bot.

Contiene due metodi:

***Main***: metodo che viene chiamato all’avvio del bot, che richiama al suo interno il metodo “*CreateWebHostBuilder”****.***

***CreateWebHostBuilder:*** Inizializza una nuova istanza della classe WebHostBuilder con pre-configurazone di default.

#### **Startup**

La classe ***Startup*** configura i servizi e la pipeline delle richieste dell'app.

Contiene due metodi:

***ConfigureServices***: metodo di tipo *void* che configura i servizi dell'app. Un servizio è un componente riutilizzabile che fornisce la funzionalità delle app.

***Configure***: metodo di tipo *void* che crea la pipeline di elaborazione delle richieste dell'app.

#### **Bot**

Classe che fa l’override di due metodi della classe ***ActivityHandler***.

I metodi contenuti al suo interno sono:

***OnMessageActivityAsync***: metodo che viene chiamato ogni volta che l’utente invia un messaggio al bot. Al suo interno viene chiamato il metodo ***EvaFunctions*** che gestisce la logica di conversazione del bot.

***OnEventActivityAsync:*** metodo che viene chiamato quando si verifica un evento asincrono. Il suo scopo è quello di mandare i *messaggi di benvenuto all’utente* su *web chat* al primo avvio del bot.

#### **UtilityEva**

È una classe astratta che contiene tutti gli attributi e metodi che riguardano la logica di comportamento del chatbot spiegata nel diagramma delle macchine a stati.

Elenco degli attributi:

***API\_KEY***: attributo di tipo *string*, che specifica la chiave necessaria per utilizzare le API di EVA fornite da app2check.

***BOT\_ID***: attributo di tipo *string*, che specifica l’id del bot EVA che viene interrogato tramite le API.

***EMAIL\_OP***: attributo di tipo *string*, che specifica l’email dell’account EVA associato al bot.

***isDev***: attributo di tipo *boolean,* che indica la modalità (Produzione o Sviluppo) di esecuzione del bot.

***client:*** attributo di tipo *HttpClient*utilizzato per l'invio di richieste HTTP e la ricezione di risposte HTTP dal seguente URL: <https://api.chat2check.com/eva>.

***faqIdList***: attributo di tipo *List<string>,* utilizzato per memorizzare gli id delle risposte in modalità sviluppo necessari per l’utilizzo del metodo ***EvaluateAndCorrect***.

***lastQuery***: attributo di tipo *string*, che memorizza l’ultimo messaggio inviato dall’utente.

***isToBeCorrect***: attributo di tipo *boolean*, se è settato a *true* viene eseguito ***EvaluateAndCorrect.***

***rate:*** attributo di tipo *boolean*, se è settato a *true* viene eseguito il metodo ***MakeRate***.

***feedback:*** attributo di tipo *boolean*, se è settato a *true* viene eseguito il metodo ***MakeFeedback.***

***problem:*** attributo di tipo *boolean*, se è settato a *true* viene eseguito il metodo ***MakeReport.***

***count***: attributo di tipo *int,* utilizzato come contatore per le recensioni all’interno del metodo ***MakeFeedback***.

Elenco dei metodi:

***InitializeAPI***: metodo che inizializza tutti gli attributi elencati all’avvio del bot.

***EvaFunctions:*** metodo *asincrono* che restituisce un tipo *Task*, che richiama al suo interno i seguenti metodi:

* *AskCorrectResponseId;*
* *SelectRate;*
* *AskProblem;*
* *EvaluateAndCorrect;*
* *MakeRate;*
* *MaleReport;*
* *RetriveAnswer;*

***WelcomeMessages****.* metodo asincrono che restituisce un tipo *Task,* che genera i messaggi di benvenuto*.*

***RetriveAnswer****:* metodo asincrono di tipo *void,* che richiama l’omonima API EVA*.*

***AskCorrectResponseId****:* metodo asincrono che restituisce un oggetto di tipo *Task,* che chiede di inserire l’id della risposta corretta.

***EvaluateAndCorrect****:* metodo asincrono di tipo *void,* che richiama l’omonima API EVA.

***SelectRate***: *metodo asincrono di tipo void,* che genera le possibili valutazioni di una risposta del bot.

***MakeRate****:* metodo asincrono di tipo *void,* che permette valutare una risposta del bot. Al suo interno chiama il metodo ***MakeFeedback***.

***MakeFeedback:*** metodo asincrono di tipo *void*, che permette di lasciare una recensione.

***AskProblem:*** metodo asincrono che restituisce un oggetto di tipo *Task,* che chiede qual è il problema che si vuole segnalare.

***MakeReport***: metodo asincrono di tipo *void*, che permette di specificare il problema riscontrato.

***SendSuggestedActions:*** metodo asincrono che restituisce un oggetto di tipo *Task*, che visualizza le seguenti *“card actions”:*

* *Correggi;*
* *Utile;*
* *Recensione;*
* *Segnalazione.*

***AnimatedResponse****:* metodo asincrono che restituisce un oggetto di tipo *Task*, fa apparire i “*dot typing*” del bot mentre sta scrivendo la risposta.

***RefreshToken****:* metodo asincrono di tipo *void*, il cui scopo è inizializzare l’attributo “turnContext” e “cancellationToken”.

***RemoveHtmlTag****:* metodo che restituisce un oggetto di tipo *string*, prende come parametro un oggetto di tipo string e si occupa di rimuovere alla stringa specificata in input i tag html.

# **7 Dynamic Model**

## **7.1 Diagramma delle macchine a stati per la modalità sviluppo:**

## **7.2 Diagramma delle macchine a stati per la modalità produzione:**