

---

# channel-handler

*Release 0.1*

Feb 19, 2020



---

## Contents

---

<b>1</b>	<b>APIS</b>	<b>3</b>
<b>2</b>	<b>Services</b>	<b>5</b>
<b>3</b>	<b>Mongo</b>	<b>7</b>
<b>4</b>	<b>Database</b>	<b>9</b>
	<b>Index</b>	<b>11</b>



This microservice is intended to communicate with other services in order to determine intents and entities to retrieve the answer for the channel.

Check also th API description here:



# CHAPTER 1

---

## APIS

---

Here we have the different views of the app.





## CHAPTER 2

---

### Services

---

Here we present all the services used to communicate with the other microservices of the system.

`services.compose` (*data*)

This method connects to the compose engine in order to get the answer for the user.

**Args:**

**param data** A dict containing data to pass to the engine. This dict contains, the user id, the agent, the context and the user input.

**Returns:** A dict containing the context and the answer for the user.



`mongo.create_entry (user, entry, agent)`

This method creates an entry in the conversation thread between the user and the chatbot. The id of the current channel is also stored.

**Parameters:**

**param user** The user object containing all the information of the user.

**param entry** The current entry of the conversation, the input context is also stored.

**param agent** The agent name of the conversation.

**Return:** The id of the current interaction stored.

`mongo.update_entry (entry_id, output)`

This method updates the output of the current conversation thread.

**Parameters:**

**param entry\_id** The id of the message.

**param output** The output context of the interaction.



**class** database.Channel (\*\*kwargs)

A communication channel used to interact with the user and the system.

**Attributes:**

**param @id** Id to populate the database.

**param @name** The name or address of the channel

**param @agent\_id** The id of the agent containing this channel

**param @agent** Tha object agent containing this channel

**param @social\_network** The name of the social network used

**class** database.User (\*\*kwargs)

A class to store the user object.

**Attributes:**

**param @id** Id to populate the database.

**param @user\_name** The user name of the user in the social network.

**param @name** The name of the user

**param @last\_name** The last name of the user

**param @social\_network\_id** The id of the social network if it has it.

**param @channel\_id** The id of the channel where the user is interacting.

**param @channel** The channel object where the user is interacting.

**class** database.Agent (\*\*kwargs)

An Agent refers to the main o domain or purpose of the chatbot.

**Attributes:**

**param @id** Id to populate the database.

**param @name** This name must be unique to identify the Agent.

**param @about** A description of the purpose of the chatbot.

`database.get_channel_id(token)`

This method finds the Channel Object using the authentication token from the channel.

**Parameters:**

**param token** the authentication token used by the channel to communicate with the system.

**Return:** The channel object if found or None.

`database.get_or_create_user(user, channel)`

This method get an user object or creates it if it doesnt exists.

**Parameters:**

**param user** A dict containing information of the user.

**param channel** A channel object used by the user to communicate with the system.

Check also the TODO lis for this project:

This project is part of the architecture described in: Herrera, Andre & Yaguachi, Lady & Piedra, Nelson. (2019). Building Conversational Interface for Customer Support Applied to Open Campus an Open Online Course Provider. 11-13. 10.1109/ICALT.2019.00011.

### A

Agent (*class in database*), 9

### C

Channel (*class in database*), 9

compose() (*in module services*), 5

create\_entry() (*in module mongo*), 7

### G

get\_channel\_id() (*in module database*), 10

get\_or\_create\_user() (*in module database*), 10

### U

update\_entry() (*in module mongo*), 7

User (*class in database*), 9