External API recommendations

It is possible to prepare any number of "integration steps" in the virtual assistant, which are points in the conversation tree where the virtual assistant uses the API to integrate with another system in order to obtain some data necessary for its function or, conversely, to send some data that it has detected from the user to the system.

The virtual assistant can adapt quite a bit to the format and structure of the API, but if the API is created for the sake of the virtual assistant, the following recommendations can be followed to make the implementation as smooth as possible.

Recommended API properties

- Authentication is ideal in one of the following ways
  - username+password as a BASE64 string in the header `Authorization: Basic ...`
  - token in the header `Authorization: Bearer ...`
    - can be predefined for the entire virtual assistant
    - or if the bot e.g. runs inside the system where the user logs in, this token can always be passed to the bot with the scop of the user, so that the bot will have at most his permissions and not more
  - standard OAuth2 `client_credentials` grant (i.e. a separate request to obtain and extend the access token), which is then used in `Authorization: Bearer ...` with each request
- Ideally follow the REST rules for the HTTP methods used
  - **GET** for data collection
  - **POST** to create a new record or start an action (not idempotent)
  - **PUT** to update an existing record (its identifier ideally as a URL parameter)
  - **PATCH** to partially update an existing record (its identifier ideally as a URL parameter)
  - **DELETE** to delete an existing record (its identifier ideally as a URL parameter)
- Request and response body as JSON
- If the method requires it, implement filtering and sorting as URL parameters or as custom `X-...` HTTP headers
- It would be a good idea to keep the maximum response time under 3 seconds, because in some use-cases the API may be called synchronously in real time and the user will wait for the result

Sample

The virtual assistant can use the API to retrieve a list of scheduled events from the dependent system. The user IDs are passed, for example, to a WebChat component inside an internal system where the user is already authenticated. The data is used by the bot to display a "carousel" of scheduled events, where the user can select one of them and continue with some other action.

```
GET https://somedomain.cz/api/v1/user/33/events

Authorization: Bearer ...
Content-type: application/json

→ 200 [{"id":"1", "name":"Onboarding meeting", "date":"2020-05-07T08:22:30.871Z"}]
```

At the end of the communication, the virtual assistant can send information about the newly acquired user to the CRM. The request can include all the data that the virtual assistant has collected about the user.

```
POST https://somedomain.cz/api/v1/user

Authorization: Bearer ...
Content-type: application/json
{"email": "user@example.com", "name":"Jack", "surname":"User", "interestedInProductIds":
[244, 234]}

→ 200 {"id":"433"}
```

Revision #2
Created 18 August 2021 08:46:58
Updated 18 August 2021 10:42:11