2122_INFO2_ServicesWeb_Projet

Service d'administration de ChatBot

(English version below)

Dans ce projet, il vous est demandé d'implémenter un service **d'administration** de **ChatBot**. Un ChatBot utilise la technologie Rivescript comme **cerveau**. Le ChatBot peut opérer sur plusieurs **interfaces** : une interface web dédiée, sur Mastodon, Discord, Slack, ...

ChatBot
Un ChatBot n'est pas un pas un poisson d'eau douce (ça, c'est un chabot). C'est un robot conversationnel. En d'autres termes, c'est un logiciel qui propose une interface en langage naturel.

Cerveau (brain rivescript)
Rivescript (https://www.rivescript.com/) est d'abord un langage de description d'interactions « intelligentes » pour des ChatBot basé sur la reconnaissance de schéma (Si tu me dis « Bonjour » alors je te réponds « Salut »). Ces interactions sont décrites dans des fichiers (extension .rive) appelés « brain » (cerveau) car l'ensemble des fichiers .rive donnés à un ChatBot constitue sa « personnalité » (voir https://www.rivescript.com/docs/tutorial#tutorial). Ainsi, vous aller pouvoir créer plusieurs ChaBot ayant chacun sa spécialité/fonction/personnalité, selon les fichiers .rive chargés.

Interface
Rivescript propose plusieurs implémentations (Javascript et Java entre autres). Ceci permet de connecter le cerveau à diverses interfaces textuelles comme le terminal, un dispatcher de sms, ... ou tout simplement le protocole HTTP (protocole textuel question/réponse) et au-delà, toute API basée sur HTTP. Si les fichiers .rive constituent le cerveau de votre ChatBot, ces interfaces sont autant «d'oreilles » et de « bouches ». Dans un premier temps, votre ChatBot sera présent sur une interface Web créée par vous : une simple interface de type SMS.

Administration
Dans ce projet, l'administration des ChatBots se fait à travers une API REST. Administrer un ChatBot, s'est le créer, et le détruire... c'est aussi lui modifier sa personnalité et consulter son état. On peut aussi lui donner une nouvelle interface (lui donner la possibilité d'aller sur Discord par exemple), ou lui enlever.

Use cases

- à travers une interface d'administration, je crée un nouveau ChatBot appelé « Steeve » ayant des fonctionnalités minimales (fichier « standard.rive » de https://www.rivescript.com/try). Je peux alors communiquer avec Steeve à travers une seconde interface (interface de communication, la « bouche »), sur une adresse (et port) propre à Steeve, après m'être identifier avec un login.
- à travers une interface d'administration, je sélectionne un fichier « cerveau » dans une base et je l'ajoute à Steeve. En relançant l'interface de communication, je permets à Steeve de prendre en compte le nouveau fichier. (BONUS : Steeve charge son nouveau cerveau au cours de la conversation).

- Grace à son nouveau cerveau, Steeve peut maintenant garder en mémoire des informations personnelles sur moi en reliant, par exemple, mon login avec ma couleur favorite. (BONUS : stocker le profil collecté sur une base MongoDb en ligne).
- à travers l'interface d'administration, je sélectionne un fichier de paramètres de connexion à Discord (cf https://discordjs.guide) ou Mastodon (utiliser https://botsin.space/api/v1/) et ordonne à Steeve d'y être présent. Je peux alors parler avec une instance de Steeve. (BONUS : L'instance de Steeve qui parle sur botsinspace et celle qui parle sur l'interface de communication partagent les mêmes données collectées.) (BONUS : Je peux aussi choisir une connexion à Slack).
- à travers l'interface d'administration, je supprime l'accès de Steeve à Discord.
- À travers l'interface d'administration, je peux voir l'état (les interfaces actives, a minima) des différents ChatBots (Eude, Hubert, Yann et Adam... tous de la famille Lefrigo). (BONUS : l'interface diffuse les conversations).
- BONUS : Je peux recouper des informations d'un utilisateur recueillies par différents ChatBot.

Cadre du projet

- Ce projet s'effectue en binôme (à constituer : trouvez votre binôme et informez en le responsable de module lors de la première séance).
- Votre code doit être documenté.
- Vous utiliserez GIT pour la gestion de projet (donner le lien vers le repo GIT à votre responsable de module dès la première séance de projet pour qu'il accède à votre travail).
- Le repo GIT doit contenir une procédure d'installation et une procédure de test.
- L'API est en NodeJS.
- L'architecture de l'API est REST (Elle répond aux bonnes pratiques décrites dans OpenAPI)
- Aucun développement ne sera autorisé après le 08/06/2022 20h00 (Je fais un download de tous vos repo à ce moment.).
- Chaque binôme fera la démonstration de son projet lors des séances du 9/06 et 10/06 (Indiquer les fonctionnalités implémentées, jouer un use case et montrer le code lié à un choix technologique).

---

*In this project you are asked to implement a **ChatBot administration service**. A ChatBot uses Rivescript technology as its **brain**. The ChatBot can operate on several **interfaces**: a dedicated web interface, on Mastodon, Discord, Slack, ...*

ChatBot

A ChatBot is not a freshwater fish (that's a chabot). It is a conversational robot. In other words, it is a software that provides a natural language interface.

Brain (brain rivescript)

Rivescript (https://www.rivescript.com/) is first of all a language for describing "intelligent" interactions for ChatBots based on pattern recognition (If you say "Hello" to me then I answer "Hi"). These interactions are described in files (extension .rive) called "brain" because the set of .rive files given to a ChatBot constitutes its "personality" (see https://www.rivescript.com/docs/tutorial#tutorial). Thus, you will be able to create several ChaBots, each with its own speciality/function/personality, depending on the .rive files loaded.

Interface

Rivescript offers several implementations (Javascript and Java among others). This allows to connect the brain to various textual interfaces like the terminal, a sms dispatcher, ... or simply the HTTP protocol (textual question/answer protocol) and beyond, any HTTP based API. If the .rive files are the brain of your ChatBot, these interfaces are as many "ears" and "mouths". At first, your ChatBot will be present on a web interface created by you: a simple SMS-like interface.

Administration

In this project, the administration of ChatBots is done through a REST API. Managing a ChatBot consists in creating it, and destroying it... it's also modifying its personality and consulting its state. We can also give it a new interface (give it the possibility to go on Discord for example), or remove it.

Use cases

- *Through an administration interface, I create a new ChatBot called "Steeve" with minimal functionalities (file "standard.rive" from https://www.rivescript.com/try). I can then communicate with Steeve through a second interface (communication interface, the "mouth"), on an address (and port) specific to Steeve, after having identified myself with a login.*
- *Through an administration interface, I select a "brain" file in a database and I add it to Steeve. By restarting the communication interface, I allow Steeve to take into account the new file. (BONUS: Steeve loads his new brain during the conversation).*
- *Thanks to his new brain, Steeve can now store personal information about me by linking, for example, my login with my favorite color. (BONUS: store the collected profile on an online MongoDb database).*
- *Through the administration interface, I select a Discord (cf https://discordjs.guide) or Mastodon (use https://botsin.space/api/v1/) connection parameters file and order Steeve to be present there. I can then talk with an instance of Steeve (BONUS: The instance of Steeve that talks on botsinspace and the one that talks on the communication interface share the same collected data). (BONUS: I can also choose a connection to Slack).*
- *Through the administration interface, I remove Steeve's access to Discord.*
- *Through the administration interface, I can see the status (active interfaces, at least) of the different ChatBots (Eude, Hubert, Yann and Adam... all from the Lefrigo family). (BONUS: the interface broadcasts the conversations).*
- *BONUS: I can cross-check information about a user collected by different ChatBots.*

Framework of the project

- This project is done in pairs (to be set up : create a pair and inform your teacher during the first session of the project).
- All your code should be documented
- You will use GIT to manage your project (You will give the link to your repo during the first session so that I can access it).
- The Git repo should contain an install procedure and a test set.
- The technology is NodeJS.
- The APIs will be RESTful (see OpenAPI and follow best practices)
- I will download all your code from the GIT on the 8th of June 8 o’clock PM.
- ***Each pair will present its work during the sessions 9 and 10th of June. (What are the implemented fonctionalities, present a use case and a specific technological implementation.)***