Rapport de projet

Création d'une application avec **Ingescape Circle**

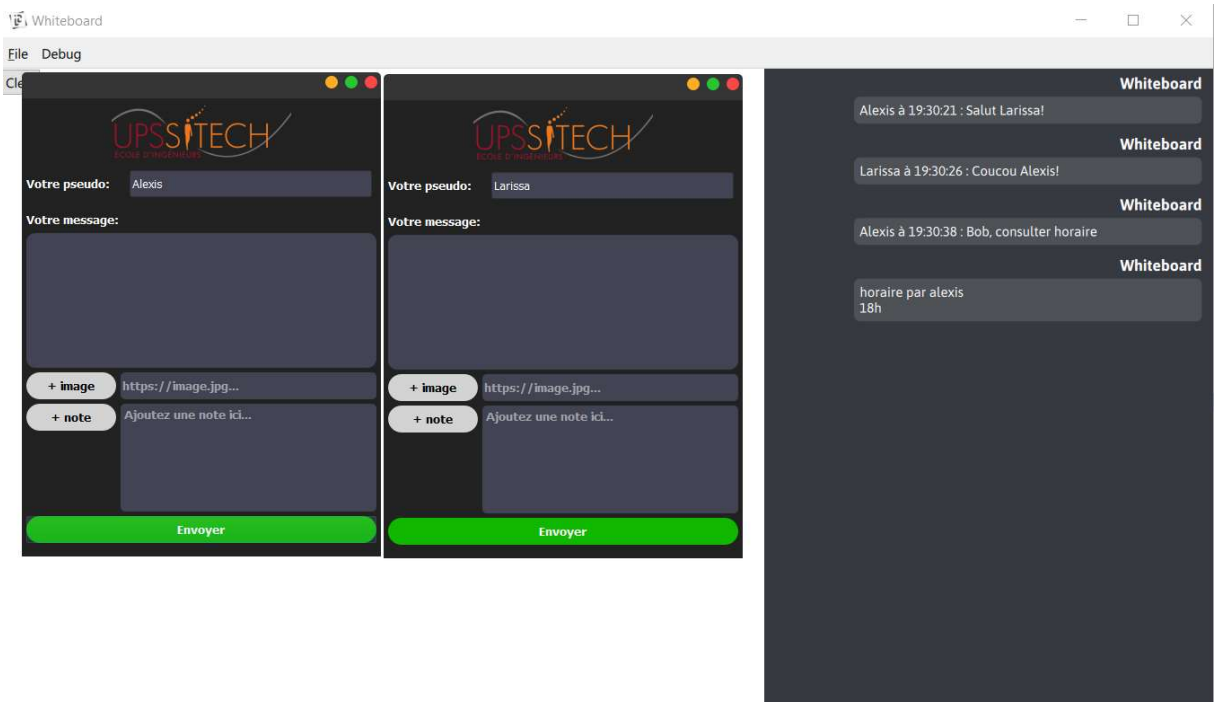
XAVIER VITOR Larissa
DELAUNAY LE DINS Maureen
LEVANEN Antoine
ZEMB Alexis

Année 2023 – 2024

Introduction

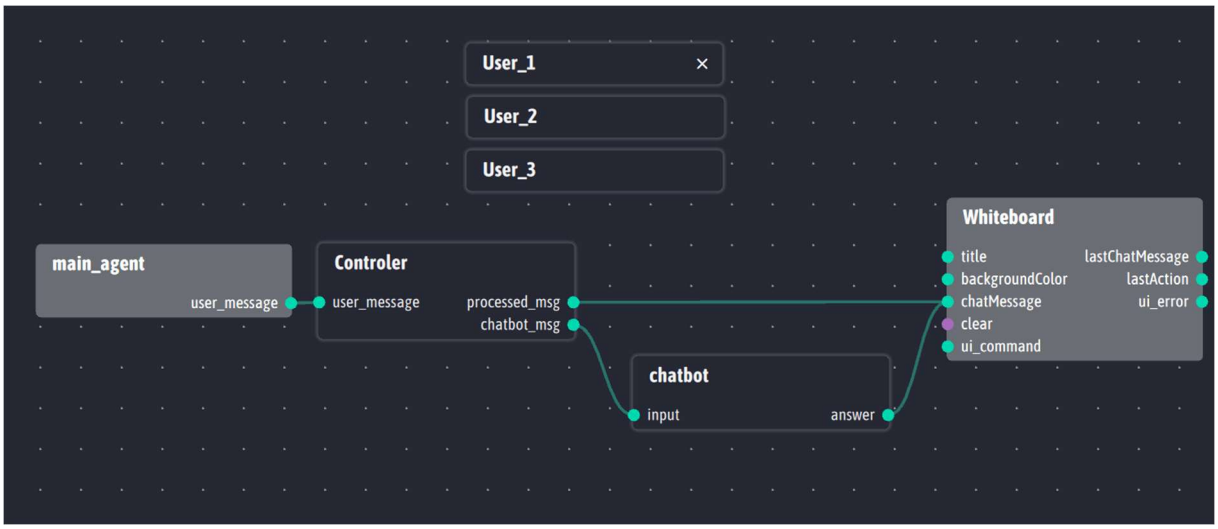
Au cours de notre dernière année d'études d'ingénieurs en systèmes robotiques et interactifs à l'UPSSITECH, nous avons eu l'opportunité de réaliser un projet en collaboration avec l'entreprise **Ingenuity i/o** basée à Toulouse. Ce projet s'est déroulé dans le cadre du cours d'interaction distribuée, et a été réalisé avec le logiciel **Ingescape Circle** développé par l'entreprise. Ce logiciel permet de retranscrire dans une interface graphique l'interaction entre différents agents logiciels, de manipuler leurs entrées/sorties, ou encore de générer un squelette de code à partir de la définition graphique d'un agent.

L'objectif de ce projet est de créer une application constituée d'agents fonctionnant avec Ingescape Circle. Nous avons ainsi choisi de créer une application de chat multi-utilisateurs, intégrant une partie communication (chat) et une partie interaction avec l'intégration d'un chatbot à la discussion, permettant aux utilisateurs de demander des informations. Ce système de chat aura pour but la planification de randonnées entre utilisateurs.



Architecture globale

L'architecture globale du projet comprend au total quatre agents distincts, chacun chargé d'effectuer une tâche spécifique. A cette architecture globale s'ajoute des instances de l'agent User (un User pour chaque utilisateur connecté). Du point de vue d'Ingescape Circle, l'architecture peut être visualisée de la façon suivante :



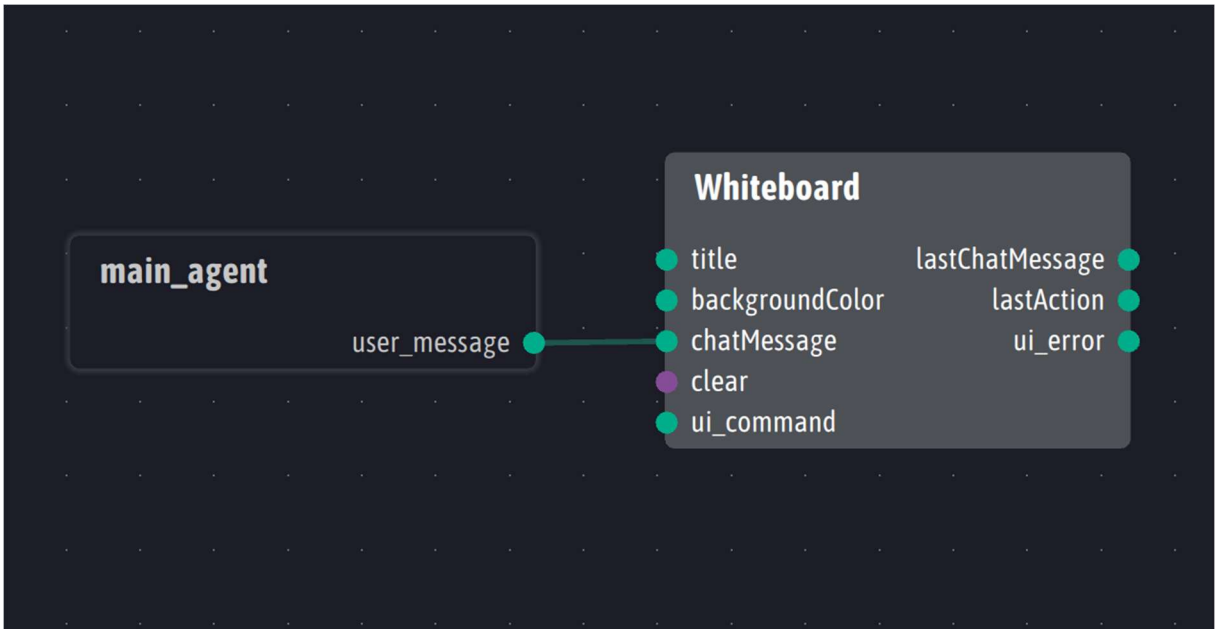
Dans cette architecture, le main agent possède un service « WaitMessage », qui reçoit les messages émis par les différents utilisateurs. Les messages émis sont ensuite transférés au contrôleur, chargé de formater les messages avant de les afficher sur le whiteboard. Les messages émis par les utilisateurs sont en effet de la forme « HH:MM:SS#user_name#str_msg ». Le message contient ainsi l'information d'heure d'émission du message, de l'utilisateur émetteur, et le message en lui-même. Les caractères « # » sont utilisés comme séparateurs entre les différentes sections du message reçu.

Le contrôleur transfère ensuite le message formaté au whiteboard afin de les afficher. Si le mot clé de déclenchement du chat bot est détecté dans le message de l'utilisateur, le message est également transmis au chat bot qui traite la demande et renvoi sa réponse au whiteboard pour l'afficher.

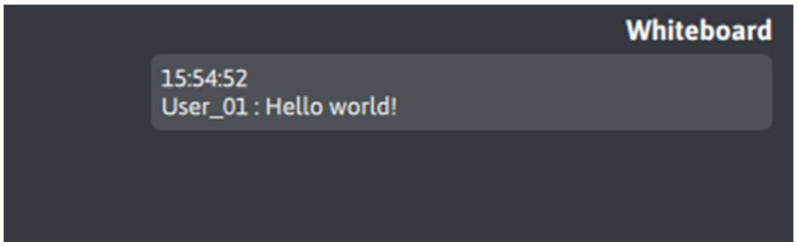
Les utilisateurs ont également la possibilité d'ajouter des images et des notes au whiteboard avec d'autres fonctions détaillées par la suite.

Agent principal

Présentation



L'agent *main_agent* est l'agent qui agit comme le cœur de l'application, lles utilisateurs envoient des messages qui sont récupérés par l'agent *main_agent* . Celui-ci formate les messages pour les afficher avec l'heure d'envois, le nom de l'utilisateur et le contenu du message.



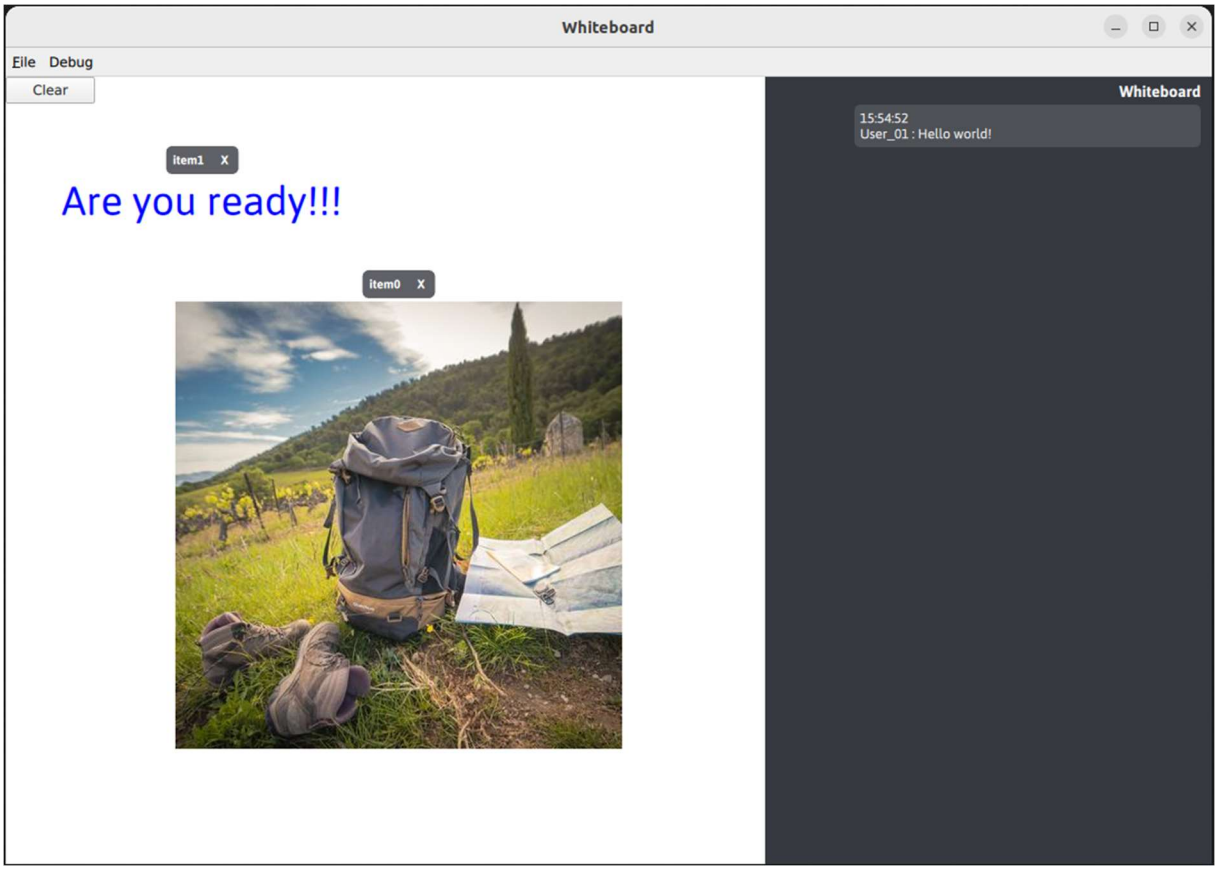
Capture d'écran du message formaté et affiché dans le whiteboard.

L'agent *main_agent* possède seulement une sortie *user_message* qui est une chaîne de caractères formatés pour l'affichage sur le whiteboard.

L'agent *main_agent* dispose d'un service qui écoute et traite les messages envoyés par l'utilisateur.

Fonctionnement

Le service de l'agent main est capable de recevoir plusieurs types de message, des messages utilisateur qui feront partie de la conversation sur le whiteboard, des messages de types images pour placer des images tirées du web dans l'interface du whiteboard, et enfin des messages de types notes pour afficher des messages en couleur dans l'interface du whiteboard comme illustré ci-dessous.



Pour cela le service **waitMessage** attend comme paramètres le type de message, un entier entre 0 et 2, et le contenu du message, la fonction **service_callback()** associée au service **waitMessage** traite donc les messages en fonction de leur type.

Agent user

Présentation

L'agent *User* est l'agent qui offrira à l'utilisateur une interface d'interaction avec l'application de chat. Il est similaire à ce que l'on retrouve dans beaucoup de systèmes de communications. Il est composé d'une partie back-end (services et fonctions de rappel, pour transmettre les informations aux différents agents), et d'une partie front-end, offrant une interface graphique à l'utilisateur.

Capture d'écran de l'interface graphique présentée à l'utilisateur au démarrage de l'agent. Elle est composée de deux champs, un dédié au pseudonyme de l'utilisateur, et le second dédié à l'écriture du message. Le bouton envoyer permet l'envoi du message à l'agent principal si les champs de pseudo et de message ne sont pas vides.

Deux autres champs et boutons sont également accessibles depuis l'interface utilisateur. Le premier champ permet d'ajouter l'URL d'une image, et d'appuyer sur le bouton « + URL image » pour ajouter l'image au whiteboard. Le second champ permet d'entrer le texte d'une note, et de l'ajouter au whiteboard avec le bouton « + note ». Une note est un texte déplaçable sur le whiteboard. Ces deux autres fonctions appellent des services du whiteboard pour y ajouter des images et du texte.

L'agent User ne possède ni entrées ni sorties à proprement parler. Du point de vue Ingescape Circle (capture ci-contre), l'agent User est donc une simple « boîte » qui interagit via des services ou des fonctions de rappel. L'agent ne possède pas d'entrées ou de sorties dans la mesure où les entrées se font via son interface graphique (invisible sur Circle) et que les données sont transmises à l'agent principal en appelant son service d'écoute.

Fonctionnement

L'agent User est relativement simple, et fonctionne de façon scindée en deux scripts python différents. Le premier script est celui de l'interface graphique (UI) présentée plus haut. Cette UI est codée avec les librairies PyQt5 (librairie dédiée à la conception d'interfaces graphiques) ainsi que qtmodern (une autre librairie fonctionnant avec PyQt5 et permettant d'ajouter un style plus moderne à l'interface).

Le second script est le « main », c'est-à-dire le corps principal de l'agent. C'est dans ce script que sont définies les différentes fonctions, fonctions de rappel et services que peut utiliser l'agent. Différentes fonctions sont définies :

- La fonction **input_callback**(**str** : msg), permettant l'envoi d'un message sous forme de String à l'agent principal en appelant son service « Wait_message ».
- La fonction **start_UI**(**function** : **input_callback**), permettant de démarrer l'interface utilisateur définie dans le script UI.py. Cette fonction prend en argument la fonction input_callback, qui est appelée par l'interface utilisateur lorsque ce dernier envoie un message.

L'agent principal est quant à lui constitué de la structure de base créée lors de la génération de code par Ingescape Circle, à l'exception de deux nouvelles lignes permettant d'appeler la fonction **start_UI** dans un thread (c'est-à-dire dans un nouveau processus) afin de ne pas bloquer l'exécution du *main* de l'agent. Ce threading de la fonction **start_UI** est réalisé avec la librairie threading, une librairie native dans python.

Agent chatbot

Le chatbot offre la possibilité d'accéder et de modifier des informations partagées entre les utilisateurs. Ces informations, également appelées "topics", peuvent englober divers types de données telles que l'horaire, le lieu, la température, les participants, etc. Afin d'interagir avec le chatbot, veuillez débuter vos requêtes par "Bob, ".

Commandes possibles pour le chatbot

1. Consulter des informations : Pour obtenir des détails sur une information, utilisez des expressions telles que "je veux consulter [topic]" ou "consulter [topic]". Pour consulter l'historique complet d'un topic, ajoutez l'expression "historique" entre "consulter" et le topic. Exemple : "Je veux consulter l'horaire" ou "Consulter historique du lieu".
2. Modifier des informations : Pour modifier ou ajouter une information, utilisez des expressions telles que "je veux modifier [topic]" ou "modifier [topic]". Si le topic existe déjà, l'information sera ajoutée à l'historique. Dans le cas contraire, le topic sera créé et partagé avec les autres utilisateurs. Exemple : "Je veux modifier l'horaire" ou "Modifier le lieu".
3. Confirmer la participation : Pour confirmer votre participation à une randonnée, utilisez les expressions "confirmer participation" ou "participer". Aide : Pour obtenir de l'aide sur les actions possibles, entrez "aide".
4. Quitter : Pour quitter le chatbot, utilisez les expressions "exit" ou "quitter".

Utilisation du chatbot

1. Lancer le chatbot : Exécutez le script Python avec la commande python3 main.py agent_name network_device port, en remplaçant agent_name, network_device, et port par les valeurs appropriées.
2. Communiquer avec le chatbot : Utilisez la syntaxe "Bob, [votre message]" pour dialoguer avec le chatbot. Exemple : "Bob, Je veux consulter l'horaire".
3. Recevoir les réponses : Les réponses du chatbot seront affichées dans le chat du whiteboard.

Fonctionnement du code

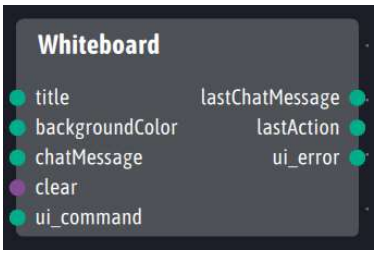
1. Le code utilise la bibliothèque nltk pour le traitement du langage naturel.
2. Des expressions régulières (patterns) sont définies pour détecter différents types de commandes de l'utilisateur.
3. Un chatbot est créé avec des patterns et des réponses prédéfinies. Les données sont stockées dans un fichier JSON (informations.json) pour maintenir un historique des informations.
4. Des threads sont utilisés pour gérer de manière asynchrone les entrées utilisateur et les sorties du chatbot.
5. Les utilisateurs interagissent avec le chatbot en préfixant leurs messages par "Bob, ".
6. Le chatbot analyse les messages de l'utilisateur, prend des actions en conséquence, et renvoie des réponses.

Note : Assurez-vous d'avoir toutes les dépendances nécessaires installées, notamment la bibliothèque nltk. Vous pouvez installer les dépendances manquantes en utilisant la commande pip install nltk.

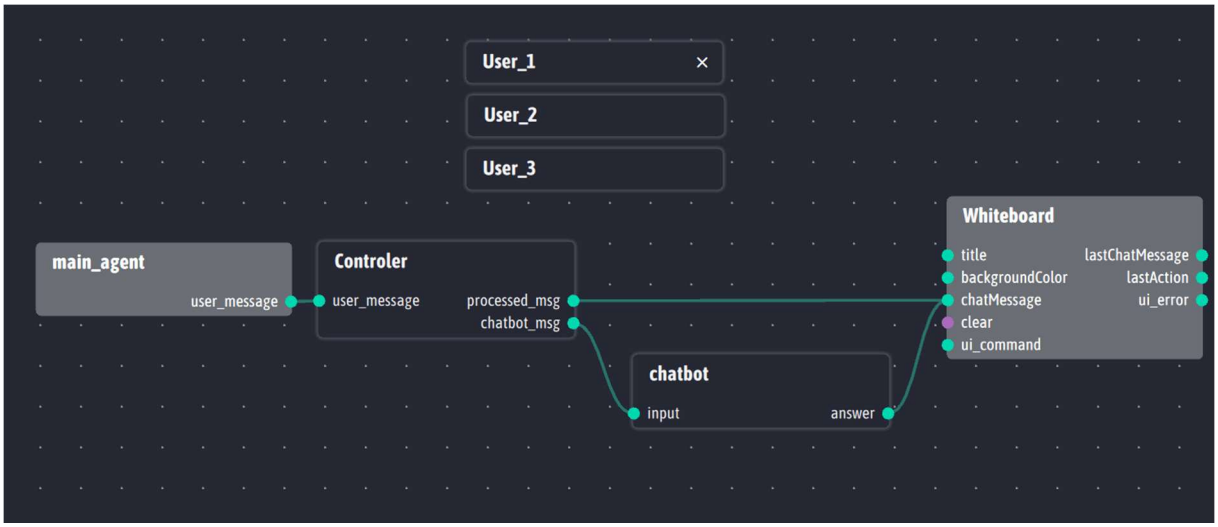
Mise en place de l'environnement du projet

Il est nécessaire de mettre en place l'environnement du projet avant de le lancer. Plusieurs outils doivent être lancés :

- **Ingescape Circle**, nécessaire à l'observation des différents agents. Il est également nécessaire de le démarrer sur un certain réseau/port pour observer sur ces derniers ;
- L'application whiteboard, dont l'agent devrait apparaitre sur Ingescape Circle comme sur la capture ci-dessous :



- Dans le dossier du projet, le Main_agent doit être démarré (python main.py main_agent Wi-Fi 5670) sur le même réseau et le même port qu'Ingescape Circle et le whiteboard, et sa sortie doit être liée à l'input raw_str du contrôleur.
- Dans le dossier du projet, démarrer l'agent chatbot et l'agent contrôleur, puis les lier de la même manière que la capture d'écran suivante :



- Dans le dossier du projet, démarrer un agent User (python main.py user Wi-Fi 5670), cet agent devrait être visible sur Ingescape circle, et son interface graphique devrait également apparaitre.

Il devrait maintenant être possible d'écrire des messages sur le whiteboard via l'interface utilisateur, en entrant d'abord un pseudo, puis un message avant d'appuyer sur envoyer. Il est possible d'instancier plusieurs utilisateurs tant que les agents sont créés sur le même réseau et le même port.

Comme spécifié dans la partie chatbot, il est également possible d'interagir avec l'agent de chat « Bob » (« Bob, consulter horaire », « Bob, consulter participation »).

Scénarios de test

Les scénarios de tests suivants sont à réaliser depuis l'interface graphique utilisateur

Scénario 01 : entrée pseudonyme			
Entrées : pseudo			
État entrée	Action	Résultat attendu	Résultat
Champ de texte pseudo vide	Envoyer	Pas de message	Pas de message
Champ de texte pseudo non vide	Envoyer	Pas de message	Pas de message

Scénario 02 : entrée message			
Entrées : pseudo, message			
État entrée	Action	Résultat attendu	Résultat
Champ de texte pseudo vide et message vide	Envoyer	Pas de message	Pas de message
Champ de texte pseudo non vide et message vide	Envoyer	Pas de message	Pas de message
Champ de texte pseudo vide et message non vide	Envoyer	Pas de message	Pas de message
Champ de texte pseudo non vide et message non vide	Envoyer	Message	Message

Scénario 03 : entrée URL image			
Entrées : URL			
État entrée	Action	Résultat attendu	Résultat
Champ de texte URL vide	Ajout image	Pas d'image	Pas d'image
Champ de texte URL non vide	Ajout image	Image ajouté	Image ajouté

Scénario 04 : entrée Note image			
Entrées : URL			
État entrée	Action	Résultat attendu	Résultat
Champ de texte Note vide	Ajout note	Pas de note	Pas de note
Champ de texte Note non vide	Ajout note	Note ajouté	Note ajouté