

Documentation interne

Application de compagnon virtuel sur smartphone



Étudiants : Mathis Ruffieux et Néréïs Dugaleix

Encadrante : Mme. Lydie du Bousquet

Tuteur : Mr. Damien Pellier

Informations d'identification du document

| | |
|-----------------------|------------------------------------|
| Référence du document | D4 |
| Version du document | 1.01 |
| Date du document | 13/06/2022 |
| Auteurs | Mathis Ruffieux et Néréïs Dugaleix |

Éléments de vérification du document

| | |
|-------------------|-----------------------------|
| Validé par | Mme. Lydie du Bousquet |
| Validé le | |
| Soumis le | 13/06/2022 |
| Type de diffusion | Document électronique (pdf) |
| Confidentialité | Non confidentiel |

| | |
|---|-----------|
| Introduction | 4 |
| Module 1 : Message | 4 |
| Objectifs et relation du module | 4 |
| Définitions de types | 4 |
| Procédure et variables | 5 |
| Module 2 : Question | 6 |
| Objectifs et relation du module | 6 |
| Définitions de types | 6 |
| Procédure et variables | 8 |
| Module 3 : Reply | 9 |
| Objectifs et relation du module | 9 |
| Définitions de types | 9 |
| Procédure et variables | 10 |
| Module 4 : RelationQR | 11 |
| Objectifs et relation du module | 11 |
| Définitions de types | 11 |
| Procédure et variables | 12 |
| Module 5 : Scénario | 12 |
| Objectifs et relation du module | 12 |
| Définitions de types | 13 |
| Procédure et variables | 14 |
| Module 6 : Note | 18 |
| Objectifs et relation du module | 18 |
| Définitions de types | 18 |
| Procédure et variables | 19 |
| Module 7 : Variable | 19 |
| Objectifs et relation du module | 19 |
| Définitions de types | 20 |
| Procédure et variables | 20 |
| Remerciements | 21 |
| Annexes | 22 |
| Annexe 1 : DDD (Dictionnaires de données) | 22 |
| Annexe 2 : Diagramme de données | 24 |

Introduction

Cette documentation vise à faciliter la maintenance du code source de l'application par un développeur. Elle regroupe les différents modules sous forme de classes pour les objets de type Message, Question, Reply, Note ainsi que le fonctionnement de la classe Scenario qui utilise ces différents objets.

La classe Note est utilisée dans le journal et les classes Message, Question et Reply dans les scénarios.

Pour compléter ce document, les fonctions sont aussi commentées et expliquées directement dans le code source disponible sur le répertoire github du projet.

Module 1 : Message

Objectifs et relation du module

Le module message a été créé afin de pouvoir manipuler, afficher les messages à l'écran dans l'application. Un message peut être créé par l'utilisateur à travers l'interface ou importer d'une base de données. Il peut aussi être modifié ou supprimé. L'utilisateur peut aussi interagir avec lui en lui mettant un émoticône "j'aime" ou "secret".

Le module message est appelé dans différents modules externes comme le module question ou reply . En effet les messages sont souvent soit des questions soit des réponses. Ils sont donc transformés en messages afin d'être affichés.

Définitions de types

- **id : int**

Valeur numérique identifiant de manière unique les différents message de l'historique

- **text : String**

Chaine de caractère correspondant au contenu du message

- **date : DateTime**

Date de création du message

- **isSentByMe : bool**

Booléen permettant de savoir si le message vient du robot ou de l'utilisateur

- **isLiked : bool**

Booléen permettant de savoir si l'utilisateur à aimé ce message

- **isSecret : bool**

Booléen permettant de savoir si l'utilisateur à mis en secret ce message

Procédure et variables

Type : public

Nom : **Message**

Entrée : id, date, text , isSentByMe, isLiked, isSecret

Sortie : message

Description : Constructeur de la classe message

Type : public

Nom : **copy**

Entrée : id, date, text , isSentByMe, isLiked, isSecret

Sortie : message

Description : permet de créer une copie d'un message on admettant les null

Type : static

Nom : **fromJson**

Entrée : Map<String, Object>

Sortie : message

Description : Convertit une map (correspondant à un Json dans notre cas) en un objet message

Type : public
Nom : **toJson**
Entrée : message
Sortie : Map<String, Object>
Description : Convertit un objet message en un json

Type : public
Nom : **setIsLiked**
Entrée : bool
Sortie : Ø
Description : setter de l'attribut isLiked

Type : public
Nom : **setIsSecret**
Entrée : bool
Sortie : Ø
Description : setter de l'attribut isSecret

Type : public
Nom : **displayContent**
Entrée : message
Sortie : Ø
Description : fonction toString de l'objet message, permet d'afficher dans la console tous les attribut d'un message

Module 2 : Question

Objectifs et relation du module

Le module question a été créé afin de pouvoir manipuler, créer des questions dans l'application. Une question peut uniquement être importée de la base de données. Elle peut être modifiée ou supprimée à travers la base de données

Le module question est appelé dans différents modules externes comme le module relation ou réponse. En effet les questions sont souvent associées à des réponses, elles apparaissent donc dans une relation avec une réponse.

Définitions de types

- **id : int**

Valeur numérique identifiant de manière unique les différents message de l'historique

- **idScenario : int**

Valeur numérique identifiant le scénarios de la questions

- **createdTime : DateTime**

date de création de la question

- **TextEN : String**

texte en Anglais de la question

- **TextFR : String**

Texte en Français de la question

- **TextJP : String**

Texte en japonais de la question

- **idNextQuestion : id**

Valeur numérique identifiant la prochaine question qui doit être posé par le robot

- **isOpenQuestion : bool**

Booléen permettant de savoir si la question est une question ouverte

- **isFirst : bool**

Booléen permettant de savoir si la question est la première du scénario

- **isEnd: bool**

Booléen permettant de savoir si la question est la dernière du scénario

- **nameVariable : String**

Nom de la variable qui doit être enregistré avec la réponse de la question

Procédure et variables

Type : public

Nom : **Question**

Entrée : id, idscenario, createdTime, textEN, textFR, textJP, idNextQuestion, isOpenQuestion, isFirst, isEnd, nameVariable

Sortie : Question

Description : Constructeur de la classe question

Type : public

Nom : **copy**

Entrée : id, idscenario, createdTime, textEN, textFR, textJP, idNextQuestion, isOpenQuestion, isFirst, isEnd, nameVariable

Sortie : Question

Description : permet de créer une copie d'une question en admettant les null

Type : static

Nom : **fromJson**

Entrée : Map<String, Object>

Sortie : question

Description : Convertit une map (correspondant à un Json dans notre cas) en un objet question

Type : public

Nom : **toJson**

Entrée : question

Sortie : Map<String, Object>

Description : Convertit un objet question en un json

Type : public

Nom : **displayContent**

Entrée : question

Sortie : Ø

Description : fonction toString de l'objet question, permet d'afficher dans la console tous les attribut d'une question

Module 3 : Reply

Objectifs et relation du module

Le module réponse a été créé afin de pouvoir manipuler, créer des réponses dans l'application. Une réponse peut uniquement être importée de la base de données. Elle peut être modifiée ou supprimée à travers la base de données

Le module réponse est appelé dans différents modules externes comme le module relation ou question . En effet les réponses sont souvent associées à des questions, elles apparaissent donc dans une relation avec une question.

Définitions de types

- **id : int**

Valeur numérique identifiant de manière unique la réponse

- **idScenario : int**

Valeur numérique identifiant le scénarios de la réponse

- **createdTime : DateTime**

Date de création de la réponse

- **TextEN : String**

texte en Anglais de la réponse

- **TextFR : String**

Texte en Français de la réponse

- **TextJP : String**

Texte en japonais de la réponse

- **idQuestion : id**

Valeur numérique identifiant une question précise par son id

- **nameVariable : String**

Nom de la variable qui doit être enregistré avec le text de la réponse

Procédure et variables

Type : public

Nom : **Reply**

Entrée : id, idscenario, createTime, textEN, textFR, textJP, idQuestion, nameVariable

Sortie : reply

Description : Constructeur de la classe reply

Type : public

Nom : **copy**

Entrée : id, idscenario, createTime, textEN, textFR, textJP, idNextQuestion, isOpenQuestion, isFirst, isEnd, nameVariable

Sortie : reply

Description : permet de créer une copie d'une reply en admettant les null

Type : static

Nom : **fromJson**

Entrée : Map<String, Object>

Sortie : reply

Description : Convertit une map (correspondant à un Json dans notre cas) en un objet reply

Type : public

Nom : **toJson**

Entrée : reply

Sortie : Map<String, Object>

Description : Convertit un objet reply en un json

Module 4 : RelationQR

Objectifs et relation du module

Le module relationQR a été créé afin de pouvoir lier les questions et les réponses dans l'application. Une relation indique quelle question est liée avec quelle réponse pour chaque objet relation.

Le module relationQR est appelé dans différents modules externes comme le module question ou réponse.

Définitions de types

- **id : int**

Valeur numérique identifiant de manière unique la relation entre les questions et les réponses

- **idScenario : int**

Valeur numérique identifiant le scénario de la relation

- **idReply : int**

Valeur numérique identifiant une réponse précise par son id

- **idQuestion: int**

Valeur numérique identifiant une question précise par son id

- **createdTime : DateTime**

Date de création de la relation

Procédure et variables

Type : public

Nom : **RelationQR**

Entrée : id, idscenario, idReply, idQuestion, createdTime

Sortie : relationQR

Description : Constructeur de la classe relationQR

Type : public

Nom : **copy**

Entrée : id, idscenario, idReply, idQuestion, createdTime

Sortie : relationQR

Description : permet de créer une copie d'une relationQR en admettant les null

Type : static

Nom : **fromJson**

Entrée : Map<String, Object>

Sortie : relationQR

Description : Convertit une map (correspondant à un Json dans notre cas) en un objet relationQR

Type : public

Nom : **toJson**

Entrée : relationQR

Sortie : Map<String, Object>

Description : Convertit un objet relationQR en un json

Module 5 : Scénario

Objectifs et relation du module

Le module scénario a été créé afin de pouvoir lier plusieurs questions, réponse dans des groupes pour pouvoir les appeler uniquement quand on en a besoin.

Le module scénario est le module central des discussions, il permet d'appeler les modules relation, question et réponses afin de les grouper en modules scénarios.

Définitions de types

- **List<Question> : questions**

Tableau de question d'un scénario

- **List<RelationQR> : relationQR**

Tableau de relation d'un scénario

- **List<Reply> : Replies**

Tableau de reply d'un scénario

- **List<Variable> : variables**

Tableau de variable d'un scénario

- **List<Reply> : currentReply**

Tableau de reply d'un scénario

- **isClosedQuestion : bool**

Variable permettant de connaître si la dernière question est posée et une question fermée.

- **isOpenQuestion : bool**

Variable permettant de connaître si la dernière question est posée et une question ouverte.

- **isResumeScenario : bool**

Variable permettant de connaître le scénario est le scénario de départ.

Procédure et variables

Type : public

Nom : **getScenarioQuestions**

Entrée : idScenario

Sortie : Future<List<Question>>

Description : renvoie la liste de questions correspondantes à un id de scénario souhaité

Type : public

Nom : **getScenarioReplies**

Entrée : idScenario

Sortie : Future<List<Reply>>

Description : renvoie la liste de réponses correspondantes à un id de scénario souhaité

Type : public

Nom : **getScenarioRelationsQR**

Entrée : idScenario

Sortie : Future<List<RelationQR>>

Description : renvoie la liste de relations questions/réponses correspondantes à un id de scénario souhaité

Type : public

Nom : **initScenarioVariables**

Entrée : Ø

Sortie : Future<int>

Description : initialise les variables de la base de données

Type : public

Nom : **getQuestionsByReply**

Entrée : reply

Sortie : Quesiton

Description : retourne la question qui suit une réponse fermée de l'utilisateur.

Type : public

Nom : **getQuestionsByQuestion**

Entrée : question

Sortie : list<Reply>

Description : retourne la liste de réponses en fonction d'une réponse fermée, en passant par la table relationQR.

Type : public

Nom : **typeOfQuestion**

Entrée : question

Sortie : int

Description : retourne le type de réponse attendue après une question :

0 = dernière réplique

1 = question ouverte

2 = question fermée

3 = robot continue

Type : public

Nom : **replyIsEnd**

Entrée : reply

Sortie : boolean

Description : retourne un bool pour savoir si la réponse conclue le scénario

Type : public

Nom : **SetVariable**

Entrée : string name , string value

Sortie : Ø

Description : modifie la valeur d'une variable en fonction de son nom, si la variable n'existe pas alors elle est créée .

Type : public
Nom : **getFirstQuestion**
Entrée : Ø
Sortie : question
Description : Retourne la première question d'un scénario.

Type : public
Nom : **getCurrentQuestion**
Entrée : Ø
Sortie : question
Description : Retourne la dernière question posée

Type : public
Nom : **getVariablebyName**
Entrée : String name
Sortie : string
Description : retourne la valeur d'une variable en fonction de son nom

Type : public
Nom : **getQuestionById**
Entrée : int id
Sortie : question
Description : Retourne un objet question en fonction de son id.

Type : public
Nom : **getReplyById**
Entrée : int id
Sortie : reply
Description : Retourne un objet Reply en fonction de son id.

Type : public
Nom : **endScenario**
Entrée : Ø
Sortie : Ø
Description : Termine un scénario en passant les idCurrent à null

Type : public
Nom : **resumeOnGoingScenario**
Entrée : Ø
Sortie : Ø
Description : permet de reprendre un scénario en cours.

Type : public

Nom : **initScénario**

Entrée : int id

Sortie : Ø

Description : Démarre un scénario en fonction de son id

Type : public

Nom : **addMessage**

Entrée : string TextMessage, bool isSentByMeMessage

Sortie : Ø

Description : Crée et ajoute un objet message en base de données

Type : public

Nom : **removePercent**

Entrée : string text

Sortie : Ø

Description : Permet de retirer les caractères %% d'un string

Type : public

Nom : **replaceStringToVariable**

Entrée : string text

Sortie : string

Description : Permet de remplacer le nom de variable dans une phrase par sa valeur en base de données.

Type : public

Nom : **displayQuestion**

Entrée : question

Sortie : Ø

Description : Affiche la question actuelle

Type : public

Nom : **displayClosedReply**

Entrée : reply

Sortie : Ø

Description : Affiche une réponse fermée et passe à la question suivante

Type : public

Nom : **displayOpenReply**

Entrée : string text, Question question

Sortie : Ø

Description : Affiche une réponse ouverte et passe à la question suivante

Type : public

Nom : **initRandomScénario**

Entrée : question

Sortie : Future<Int>

Description : Permet d'initialiser un scénario aléatoire parmi les scénarios ayant une fistQuestion.

Module 6 : Note

Objectifs et relation du module

Le module note a été créé afin de pouvoir manipuler, modifier, créer et supprimer des notes dans le journal de l'application. Une note peut être créée par l'utilisateur à travers l'interfaces. Il peut aussi la modifier ou la supprimer.

Définitions de types

- **id : int**

Valeur numérique identifiant de manière unique la note

- **title : String**

Titre de la note

- **createdTime : DateTime**

Date de création de la note

- **number : bool**

Numéro de la note

- **description : bool**

Contenu de la note

- **isSecret : bool**

Booléen permettant de savoir si l'utilisateur à mis en secret ce message

Procédure et variables

Type : public

Nom : **Todo**

Entrée : id, createdTime, title, number, description, isSecret

Sortie : todo

Description : Constructeur de la classe message

Type : public

Nom : **copy**

Entrée : id, createdTime, title, number, description, isSecret

Sortie : todo

Description : permet de créer une copie d'une note en admettant les null

Type : static

Nom : **fromJson**

Entrée : Map<String, Object>

Sortie : todo

Description : Convertit une map (correspondant à un Json dans notre cas) en un objet note

Type : public

Nom : **toJson**

Entrée : todo

Sortie : Map<String, Object>

Description : Convertit un objet note en un json

Module 7 : Variable

Objectifs et relation du module

Le module variable a été créé afin de pouvoir manipuler des variables que l'on cherche à garder à propos de l'utilisateur. Comme par exemple son nom, prénom, date de naissance afin de rendre les questions plus personnelles.

Le module variable est appelé dans tous les autres modules vu précédemment. En effet les variables peuvent intervenir dans une réponse ou dans une question. Par exemple : "Bonjour %%Prénom", ici la variable prénom sera remplacée par la valeur de la variable prénom.

Définitions de types

- **id : int**

Valeur numérique identifiant de manière unique la variable

- **date : DateTime**

Valeur alphabétique identifiant de manière unique la variable

- **name : String**

Valeur de la variable

- **value : String**

Date de création de la variable

Procédure et variables

Type : public

Nom : **Variable**

Entrée : id, createTime, name, value

Sortie : variable

Description : Constructeur de la classe variable

Type : public

Nom : **copy**

Entrée : id, createTime, name, value

Sortie : variable

Description : permet de créer une copie d'un variable en admettant les null

Type : static

Nom : **fromJson**

Entrée : Map<String, Object>

Sortie : variable

Description : Convertit une map (correspondant à un Json dans notre cas) en un objet variable

Type : public

Nom : **toJson**

Entrée : message

Sortie : Map<String, Object>

Description : Convertit un objet variable en un json

Type : public

Nom : **displayContent**

Entrée : variable

Sortie : Ø

Description : fonction toString de l'objet variable, permet d'afficher dans la console tous les attribut d'une variable

Remerciements

Nous tenons tout d'abord à remercier nos professeurs du master MIASHS parcours Informatique et cognition, en particulier notre tuteur de stage, Mr Damien Pellier de nous avoir offert l'opportunité de réaliser ce projet. Le suivi hebdomadaire et l'aide qu'il nous a fourni tout au long de l'année scolaire nous ont permis d'avancer efficacement tout en gardant un rythme de travail soutenu.

Nous souhaitons également remercier notre encadrante de stage, Mme Lydie Du Bousquet, qui nous a fait confiance pour la réalisation de ce projet et nous a indiqué le bon chemin à suivre pour son développement, en nous donnant toutes les informations nécessaires au bon développement de celui-ci lors de suivis réguliers.

Nous tenons surtout à remercier la région Auvergne-Rhône-Alpes, qui au travers de l'UGA à financer notre déplacement au Japon dans la région du Kansai, lieu de naissance du projet SCUSI Kouno Tori de compagnon virtuel sur smartphone.

De plus, avoir une expérience de stage à l'international est pour nous un atout primordial dans le secteur de l'informatique et sans leur financement et leur aide nous n'aurions pas pu effectuer ce voyage pour aller au bout de ce projet.

A ce titre, nous souhaitons tout particulièrement remercier Mr. Masahide Nakamura, professeur à l'Université de Kobe, de nous accueillir au sein de son établissement afin de poursuivre et de nous aider au développement d'un agent intelligent de compagnon virtuel, qui fait l'objet de certaines de ses recherches.

Nous n'oublions pas aussi toutes les personnes de l'administration des différentes parties, de l'Université Grenoble Alpes et de l'Université de Kobe qui nous ont aidé et conseillé sur de nombreux points.

Enfin, nous remercions toutes les personnes qui se sont portées volontaires pour tester et valider notre application mobile.

Annexes

Annexe 1 : DDD (Dictionnaires de données)

| Table | Description | type | clé | règle de calcul (format) |
|------------|---|------|----------|----------------------------|
| Message | | | | |
| id | Valeur numérique identifiant de manière unique les différents message de l'historique | N | PRIMAIRE | |
| text | Chaine de caractère correspondant au contenu du message | A | | |
| date | Date de création du message | AN | | DD-MM-YYYY |
| isSendByMe | Booléen permettant de savoir si le message viens du robot ou de l'utilisateur | N | | |
| isLiked | Booléen permettant de savoir si l'utilisateur à aimé ce message | N | | |
| isSecret | Booléen permettant de savoir si l'utilisateur à mis en secret ce message | N | | |
| Question | | | | |
| id | Valeur numérique identifiant de | N | PRIMAIRE | |

| | | | | |
|-------------------|--|----|--------------------------------------|------------|
| | manière unique les différentes questions | | | |
| idScenario | Valeur umérique identifiant le scénarios de la questions | N | | |
| creatiedTime | date de création de la question | AN | | DD-MM-YYYY |
| textEn | texte en Anglais de la question | A | | |
| textFR | Texte en Français de la question | A | | |
| textJP | Texte en japonais de la question | A | | |
| idNextQuestion | Valeur numérique idéntifiant la prochaine question qui doit etre posé par le robot | N | | |
| isOpenQuestion | Booléen permettant de savoir si la question est une question ouverte | N | | |
| isFirst | Booléen permettant de savoir si la quesiton est la première du scénario | N | | |
| isEnd | Booléen permettant de savoir si la question est la dernière du scénario | N | | |
| nameVariable | Nom de la variable qui doit etre enregistrer avec la réponse de la question | A | ETRANGER E (variable.n ame) | |
| RQuestionResponse | | | | |
| id | Valeur numérique identifiant de manière unique la relation entre les questions et les réponses | N | PRIIMARE | |
| idScenario | Valeur umérique identifiant le scénarios de la relation | N | | |
| idQuestion | Valeur numérique identifiant une question précise par son id | N | ETRANGER E (question.i d) | |
| idReply | Valeur numérique identifiant une réponse précise par son id | N | ETRANGER E (reply.id) | |
| createdTime | Date de création de la relation | AN | | DD-MM-YYYY |
| Reply | | | | |
| id | Valeur numérique identifiant de manière unique la réponse | N | PRIMAIRE | |
| idScenario | Valeur umérique identifiant le scénarios de la réponse | N | | |
| textEN | texte en Anglais de la réponse | A | | |
| textFR | Texte en Français de la réponse | A | | |
| textJP | Texte en japonais de la réponse | A | | |

| | | | | |
|--------------|--|----|----------------------------------|------------|
| idQuestion | Valeur numérique identifiant une question précise par son id | N | ETRANGER E (question.id) | |
| nameVariable | Nom de la variable qui doit etre enregistré avec le text de la réponse | A | ETRANGER E (variable.name) | |
| Notes | | | | |
| id | Valeur numérique identifiant de manière unique la note | N | PRIMAIRE | |
| isSecret | Booléen permettant de savoir si l'utilisateur à mis en secret ce message | N | | |
| number | Numéro de la note | N | | |
| title | Titre de la note | A | | |
| description | Contenu de la note | A | | |
| createdTime | Date de création de la note | AN | | DD-MM-YYYY |
| Variable | | | | |
| id | Valeur numérique identifiant de manière unique la variable | N | PRIMAIRE | |
| name | Valeur alphabétique identifiant de manière unique la variable | A | PRIMAIRE | |
| value | Valeur de la variable | A | | |
| createdTime | Date de création de la variable | AN | | DD-MM-YYYY |

Annexe 2 : Diagramme de données

