# Applications mobiles

## *Dictionary App*

Damien Vanhove, Gauthier Linard, Guillaume Verfaille

et Lancelot Lhoest

# 1. Présentation de l'application

**DictionaryApp** est une application permettant de traduire et d'enregistrer des mots dans diverses langues. La traduction est pour l'instant unidirectionnelle de Français vers Allemand, Anglais, Russe et/ou Espagnol. Il est aisé d'ajouter des langues supplémentaires, mais pour cette démonstration, nous nous limiterons à ces langues.

L'application permet à l'utilisateur de créer facilement un quiz afin de réviser pour ses interrogations. Il lui suffit d'ajouter les mots à la base de données. Les mots seront ensuite affichés dans une liste avec les traductions. L'utilisateur peut ensuite générer un quiz afin de tester ces connaissances. Les 10 mots présents dans le quiz sont récupérés de manière complètement aléatoire. Chaque quiz est ainsi différent.

La traduction des mots est réalisée en envoyant une requête à l'API libre d'utilisation fournit par le site internet http://transltr.org.

# 2. Fonctionnalités

Voici les objectifs que nous nous sommes fixés en début de projet :

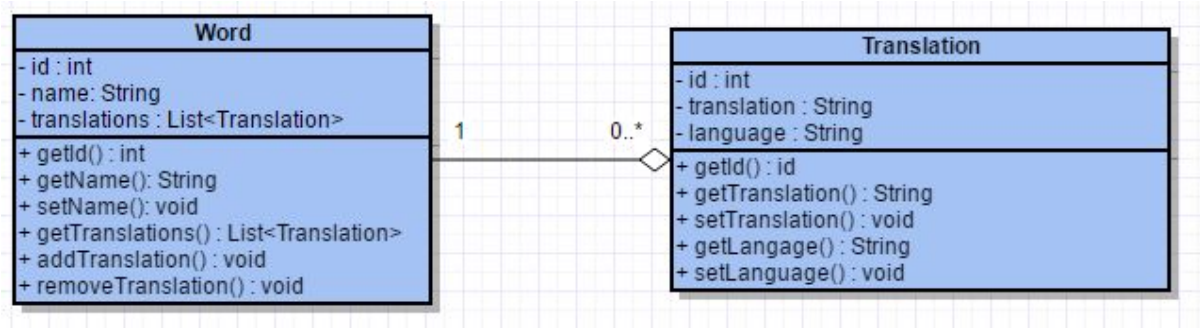
| Fonctionnalités | Priorité |
|---|---|
| activité:quizz | faible |
| activité: préférence pour choisir la langue (français vers autres) | moyen |
| activité: lister les mot et bouton pour ajouter des mots | très haute |
| activité: ajouter un mot | haute |

À la fin du projet, l'application possède les fonctionnalités suivantes:

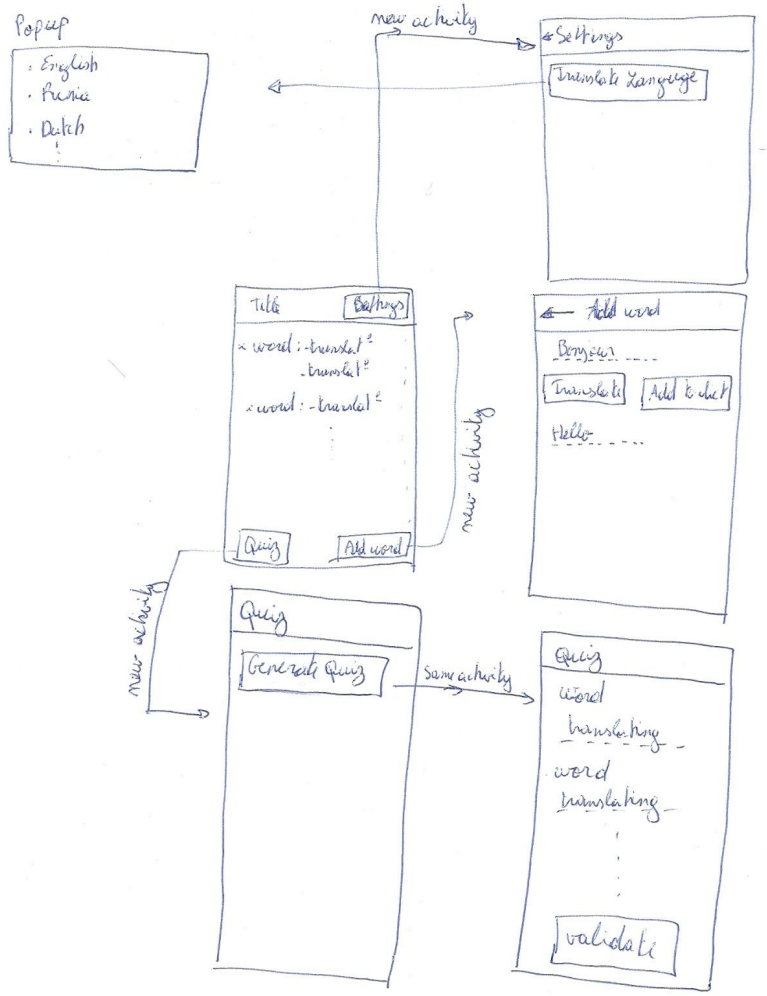
- Affichage de la liste des mots avec sa traduction
- Affichage de toutes les traductions pour un mot spécifique lorsque le mot est cliqué dans la liste
- Possibilité d'ajouter de nouvelles traductions
- Possibilité de supprimer des mots ainsi que leurs traductions
- Possibilité de générer un quiz
- Gestion de la rotation de l'appareil

# 3. Diagramme de classe

Nos données sont structurées suivant le diagramme de classe montré ci-dessous. Comme on peut le voir, un mot contient plusieurs traductions comme prévu. Le diagramme de classe de toute l'application est montré en annexe.




# 4. Croquis de l'application



# 5. Structure de la base de données

Nous avons décidé de sauvegarder les mots et leurs traductions dans une base de données. De ce fait, les données sont gardées même si l'utilisateur ferme l'application. À l'ouverture, l'utilisateur retrouvera sa liste de mots telle quelle.

Nous utilisons une base de données SQLite afin que les données soient sauvegardées sur le téléphone. Cela permet, entre autres d'utiliser l'application même sans connexion internet.

La base de données a été structurée en deux tables. Une table **Word** qui contient les mots à traduire. Une autre table **Translate** contenant les traductions. Un mot peut contenir plusieurs traductions. La structure se résume au tableau ci-dessous :

| **Table** | **Colonne 1** | **Colonne 2** | **Colonne 3** | **Colonne 4** |
|---|---|---|---|---|
| word | id : *int* | name : *String* | N/A | N/A |
| translation | id : *int* | word_id : *int* | translation : *String* | language : *String* |

La base de données étant maintenant créée, nous avons décidé de créer une classe qui permettrait la gestion totale de la base de données. De cette façon, les autres développeurs ne doivent pas se soucier de la gestion de la base de données. Il leur suffit d'appeler les méthodes du manager en passant en paramètre leur objet. Le manager s'occupera du reste: sauvegarde, modification et suppression dans la base de données.

Pour ce faire, une classe *DictionaryDBHelper* a été créée. Cette classe contient toutes les méthodes nécessaires à la sauvegarde, mise à jour et suppression de données. Parmi les méthodes publiques de cette classe, nous pouvons retrouver :

- **save(Word)** : permet de sauver un mot et les traductions liées au mot. Si le mot n'est pas encore présent dans la base de données, une nouvelle entrée sera automatiquement créée. Si le mot existe déjà, le mot sera simplement mis à jour. Cette méthode sauvegarde également chaque traduction liée à ce mot.
- **save(Translation)** : permet de sauvegarder une traduction. Comme pour le mot, si la traduction n'existe pas encore, une nouvelle entrée sera automatiquement créée. Si en contrepartie, la traduction existe déjà, la traduction sera simplement mise à jour.
- **getWords()**: récupère tous les mots et les traductions présents dans la base de données. Renvoie donc une liste de mots.
- **getWords(String)**: récupère tous les mots appartenant à une langue spécifique.
- **getWord(String)**: permet de rechercher un mot spécifique en fonction du paramètre.
- **delete(Word)**: supprime le mot ainsi que toutes ces traductions.
- **delete(Translation)**: supprime la traduction.

À chaque fois que ces méthodes récupèrent un mot, elles recherchent également les traductions liées au mot. Il est ainsi facile de parcourir toutes les traductions d'un mot certain mot pour ensuite, par exemple, pouvoir facilement l'afficher

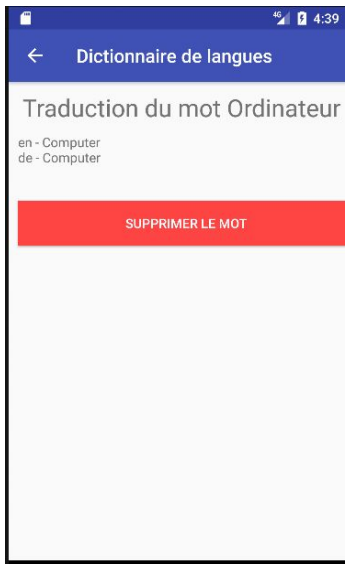
Grâce à ce manager, la programmation est ainsi plus aisée pour la suite du développement. Il ne faut en effet plus se soucier de la gestion de la base de données.

# 6. Page d'accueil


Nous avons décidé que la première chose que devrait voir l'utilisateur en ouvrant l'application serait la liste des mots que ce dernier a gardé en mémoire. Pour cela, nous avons utilisé une *RecyclerView* qui permet à l'utilisateur de faire dérouler l'ensemble des traductions.

La *RecyclerView* est structurée de la même manière que nous avons vu au cours. Les composants *ItemAdapter* et *viewHolder* sont complétés grâce à l'utilisation aisée de la base de données. Un simple appel à la méthode **DB.getWords()** permet de récupérer la liste des mots traduits. À chaque mot est alors associé un *viewHolder.* Dans ce dernier, nous appelons **getTranslations()** sur la classe *Word* pour récupérer toutes les traductions stockées dans l'objet, pour ensuite les ajouter dans la view.

Lorsque l'utilisateur clique sur un des éléments de la liste, il est renvoyé à la page suivante qui affiche toutes les traductions du mot. L'envoi du mot et de son contenu vers cette nouvelle vue se fait au moyen d'un *Intent*.





Cette activité permet également à l'utilisateur de supprimer le mot qu'il vient de sélectionner.

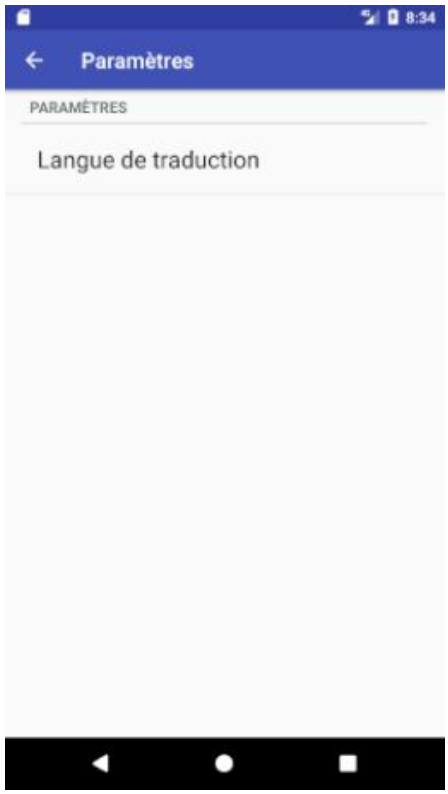
En cliquant sur **Supprimer le mot**, le texte change et demande à l'utilisateur de confirmer. S'il ne re-clique pas sur le bouton endéans un certain délai, l'opération est annulée et le texte redevient **Supprimer le mot**.

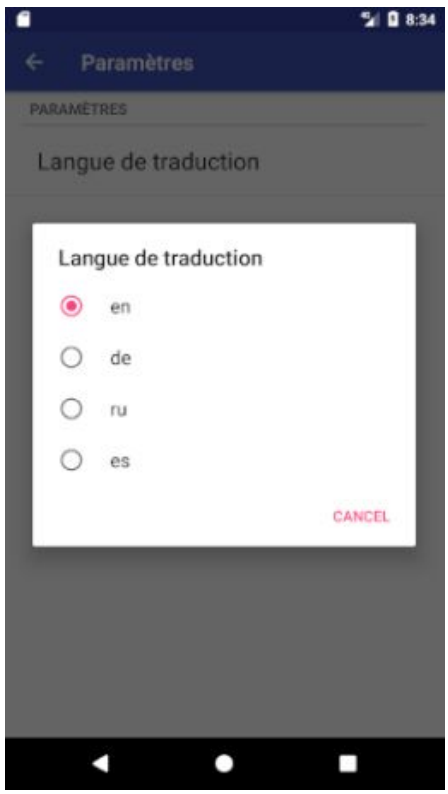
# 7. Préférences

Dans la conception de notre application, il était très important de pouvoir choisir la langue de traduction. Ce choix devait être sauvé sur l'appareil jusqu'à la désinstallation de l'application.


C'est pourquoi nous avons décidé d'intégrer un menu de préférence qui permet de choisir la langue de traduction parmi une liste. Le bouton menu en haut à droite sur l'écran principal de notre application donne accès au choix des préférences en cliquant sur l'option **Paramètres** (voir la figure ci-contre).

Quand l'utilisateur clique sur cette option, l'application lance une nouvelle tâche qui affiche et enregistre le menu de préférences. Dans ce menu, une option **Langue de traduction** permet alors de faire son choix de langue parmi une liste de langues préétablies (voir les deux figures ci-dessous).



Afin de faciliter la navigation de notre application, nous avons également ajouté un bouton dans le menu des paramètres. Ce bouton représenté par une flèche vers la gauche, permet de revenir à l'activité principale sans devoir utiliser le bouton retour par défaut dans android.

# 8. Traduction


Pour gérer la traduction, une nouvelle activité a été créée: la *NewWordActivity*. Un bouton a été ajouté à la *MainActivity* accompagné d'un **onClickListener**, afin que la *NewWordActivity* soit ouverte quand on clique dessus. Ci-contre on peut voir à quoi ressemble la *NewWordActivity*.

Si on entre un mot dans le champ prévu à cet effet et qu'on clique sur le bouton "Traduire", alors le contenu du champ est récupéré et stocké dans une variable. L'app vérifie ensuite que le téléphone est bien connecté à internet (pour pouvoir traduire le mot). Si ce n'est pas le cas, un toast est affiché le signalant à l'utilisateur.
Si par contre il est connecté et que le champ n'est pas vide, alors l'*AsynchTask* est appelée. Cette *AsynchTask* récupère la langue choisie dans les préférences et la variable contenant le mot à traduire. Elle fait appel aux classes *Networking* et *translation*. La méthode getApiTranslation permet d'effectuer une requête auprès du serveur web. Pour l'envoi d'un objet JSON avec l'ensemble des paramètres nécessaires, on reçoit en réponse un nouvel objet JSON contenant la traduction.
Grâce à la fonction **get()** de la classe *translation*, la traduction est récupérée, et la fonction **onPostExecute** est alors appelée afin d'insérer la traduction dans la *TextView* se trouvant sous les boutons.

Le deuxième bouton de la *NewWordActivity* permet de stocker le mot à traduire ainsi que sa traduction dans la base de données. Quand on clique dessus, tout d'abord le mot à traduire et sa traduction sont récupérés depuis leur champ respectif et la langue de traduction est récupérée depuis la mémoire du téléphone. Ensuite un **getTranslations()** est effectué sur le mot à traduire pour vérifier que la traduction du mot n'a pas déjà été enregistrée dans la langue dans laquelle on souhaite le stocker au moment où on clique sur le bouton. S'il a déjà été stocké dans cette langue, alors il ne l'enregistre pas à nouveau et notifie l'utilisateur à l'aide d'un toast pour lui faire savoir que la traduction a déjà été enregistrée. Sinon, la traduction est enregistrée avec la langue de sa traduction et elle est liée au mot qui était à traduire (voir chapitre concernant la structure de la base de données). Ce dernier est aussi enregistré s'il ne l'avait pas encore été, dans la table **Word**.

Dans cette activité, un aspect très important que nous avons mis en place permet de restaurer la traduction du mot précédent lorsque l'appareil passe du mode portrait au mode

paysage. Pour ce faire, nous avons utilisé le système de *savedInstanceState*. Ce système permet de distinguer la création initiale de l'activité ou la recréation de celle-ci. Ce qui nous permet de restaurer la traduction s'il s'agit de la recréation de l'activité. De plus, si l'utilisateur n'a plus de connexion à internet, l'application l'avertira avec un message *Toast*.

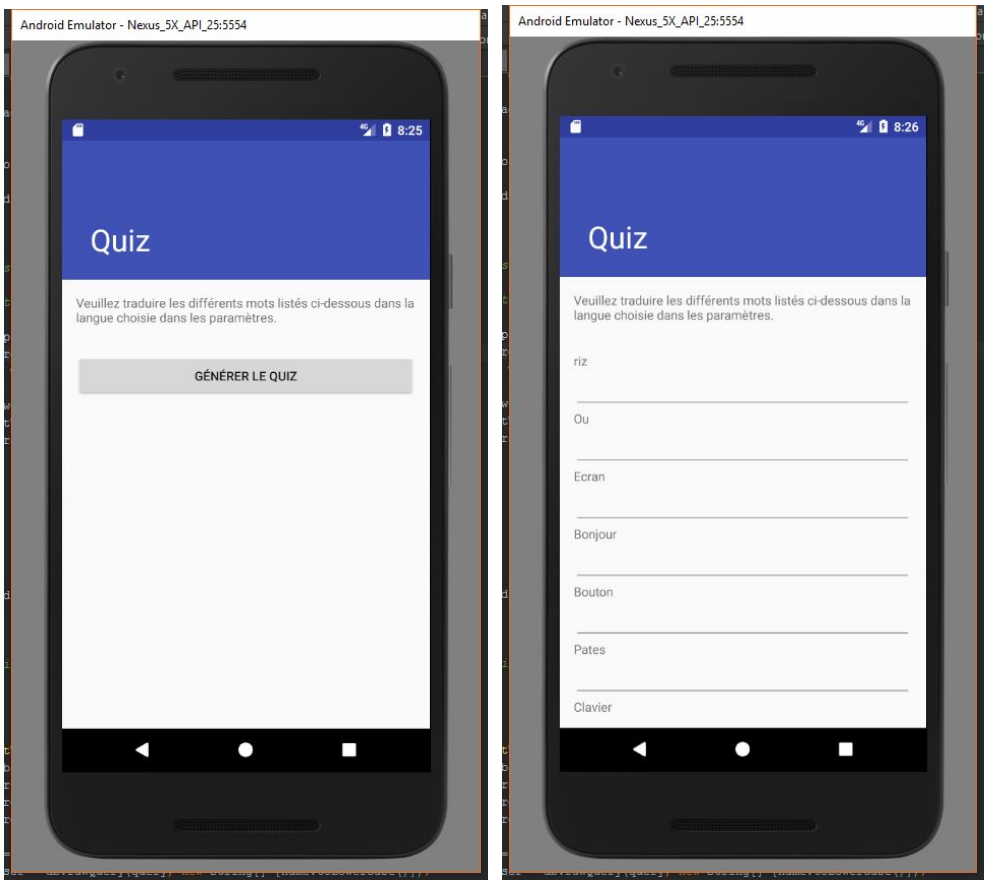
# 9. Quiz

L'application peut générer un quiz. Chaque quiz contient 10 mots maximum à traduire dans la langue de préférence de l'application. Pour afficher le quiz, il suffit de cliquer sur le bouton **Quiz** sur la page d'accueil. L'activité comprenant le quiz va donc s'ouvrir dans une nouvelle fenêtre.

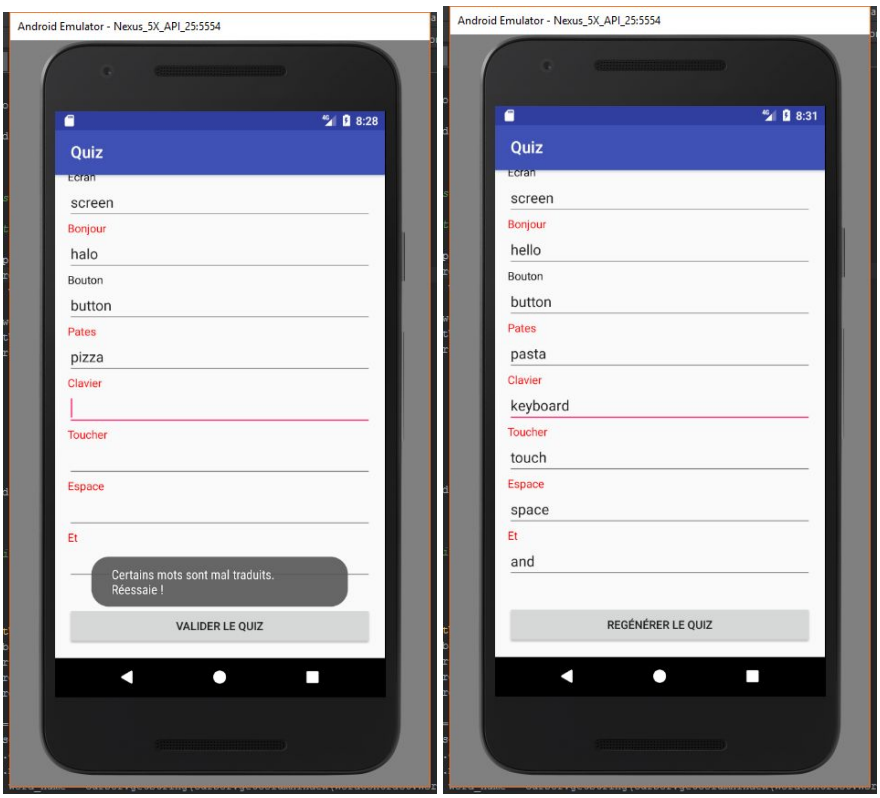
L'utilisateur est invité à cliquer sur le bouton **Générer le quiz** pour commencer. Lorsque le bouton est cliqué, plusieurs possibilités :

1) S'il n'y a aucun mot dans la langue de préférence, un pop-up avertira l'utilisateur que le quiz ne peut être créé.
2) S'il existe moins de 10 mots dans la langue de préférence, tous les mots sont affichés.
3) S'il existe plus de 10 mots dans la langue de préférence, 10 mots sont affichés à l'écran

Imaginons que la base de données contient plus de 10 mots. Les figures ci-dessous montrent les 2 premières étapes :



L'utilisateur est maintenant invité à traduire les mots. Une fois les champs remplis, nous pouvons cliquer sur le bouton **Valider le quiz**. Essayons… (cf. figure ci-dessous). Nous pouvons apercevoir que chaque mot mal traduit est surligné en rouge. Un nouveau bouton vient d'apparaître permettant d'afficher les réponses. Cliquons dessus. Nous pouvons voir que tous les mots sont maintenant traduits. Un nouveau bouton vient d'apparaître **Régénérer le quiz**. Ce bouton permet de régénérer le quiz. Si nous cliquons dessus de nouveaux mots seront affichés.




# 10. Conclusion

L'application fonctionne correctement. Nous avons développé toutes les fonctionnalités que nous espérons. Il y a cependant certaines fonctionnalités à ajouter/améliorer pour obtenir une application plus conviviale et plus facile à utiliser :

- Ajout d'un bouton dans le menu quiz qui permettrait de changer de langue de traduction. Pour l'instant, il faut aller dans les paramètres pour changer la langue du quiz
- Ajout d'un champ permettant de modifier le nombre de mots à traduire pour le quiz.
- Affichage d'un toast indiquant que la langue a bien été modifiée dans le menu des paramètres.
- Affichage des mots traduits uniquement dans une certaine langue. Par exemple afficher uniquement tous les mots qui sont traduits en anglais.

# Annexe

Diagramme de classe de toute l'application :