BONUS

Projet 9 – Le Baluchon

RAPPORT

Bonus intégré à l'application LeBaluchon

Mickaël HORN

Étudiant sur le parcours « Développeur d'Applications iOS » - OpenClassrooms

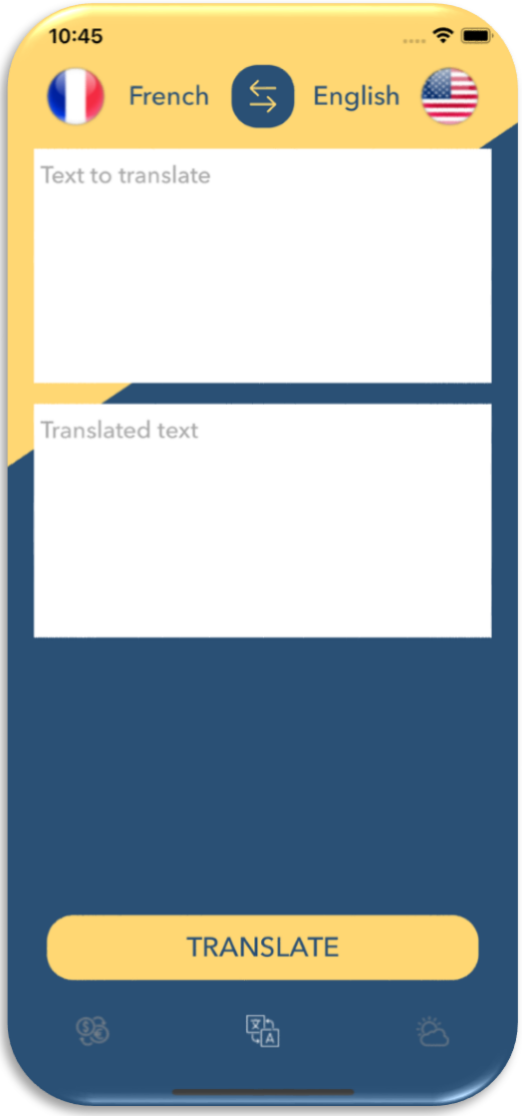
Contexte

Une liste de bonus à ajouter sur l'application nous était proposée, avec comme condition d'en choisir un.
Je me suis dirigé vers l'inversion des langues pour la partie traduction.

C'est-à-dire que mon application qui, initialement proposait une traduction « français vers anglais », propose maintenant une traduction « anglais vers français ».

Comment changer ?

Voici l'interface de la partie Traduction.
Initialement, on propose à l'utilisateur une traduction FR -> ENG.
L'utilisateur rentre son texte dans la première textView, presse le bouton TRANSLATE puis obtient sa traduction dans la seconde textView.



Si l'utilisateur souhaite inverser les langues, il lui suffit d'appuyer sur le bouton avec les deux flèches.

Après l'appui :

Dans le code

Au niveau du code, j'utilise plusieurs éléments afin d'obtenir ce résultat, les voici :

exchangeSourceAndDestination()

```
@IBAction func exchangeSourceAndDestination(_ sender: Any) {
    if sourceLabel.text == "French" {
        sourceFlag.image = UIImage(named: "united_states_of_america_round_icon_64")
        sourceLabel.text = "English"
        destinationFlag.image = UIImage(named: "france_round_icon_64")
        destinationLabel.text = "French"

    } else {
        sourceFlag.image = UIImage(named: "france_round_icon_64")
        sourceLabel.text = "French"
        destinationFlag.image = UIImage(named: "united_states_of_america_round_icon_64")
        destinationLabel.text = "English"
    }

    resetTextViews()
    textToTranslate.resignFirstResponder()
}
```

Il s'agit de l'Action du bouton des flèches, dont le rôle sera d'inverser les langues.
Dans un premier temps, on regarde le contenu du label de la langue source.
Si nous avons « French », nous savons alors que le français est notre langue source et l'anglais, la langue de destination.

Il suffit alors d'inverser les Labels et les Images.
Pour finir, on lance les fonctions resetTextViews() ainsi que textToTranslate.resignFirstResponder(), que nous détaillerons plus bas.

resetTextViews()

```
private func resetTextViews() {
    // When I want to switch languages, i'm reseting the textViews.
    textToTranslate.text = placeHolderTextToTranslate
    textToTranslate.textColor = placeHolderColor
    translatedText.text = placeHolderTranslatedText
    translatedText.textColor = placeHolderColor
}
```

Une fonction privée qui réinitialise les textViews quand on change de langage.
La réinitialisation ne videra pas seulement les zones de textes, mais va également remettre en place les placeHolders.

Étant donné qu'il n'existe pas à ma connaissance de placeHolder pour les textView, je place dans mes textViews mes propres placeHolders.

Je retrouve alors ma configuration de départ.

textToTranslate.resignFirstResponder()

Quand on switch de langue, j'enlève le focus sur la zone de texte.
La raison derrière cette décision est simple, si je ne l'enlève pas, les langues s'inverses, les placeHolders reviennent à leur état initial, mais comme le focus est toujours sur la textView, lorsque j'écris, le placeHolder ne disparaît pas pour laisser place à la saisie de l'utilisateur.

En effet, il reste sur la textView et l'utilisateur écrit à côté, ce qui n'est pas ce qu'on souhaite !

getSourceAndDestinationLanguages()

```
private func getSourceAndDestinationLanguages() -> (String, String) {
    if sourceLabel.text == "French" {
        return ("fr", "en")
    } else {
        return ("en", "fr")
    }
}
```

Une autre fonction privée qui elle, retourne un tuple (source, destination) pour fournir à la fonction chargée de l'appel API, les bons paramètres (dans le bon format).