



BONUS

Projet 9 – Le Baluchon

RAPPORT

Bonus intégrés à l'application LeBaluchon

Mickaël HORN

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

1 Table des matières

2	Contexte.....	2
3	Bonus 1 : Inversion des langues.....	3
3.1	Comment changer ?.....	3
3.2	Dans le code.....	4
3.2.1	exchangeSourceAndDestination()	4
3.2.2	displayExchangedLanguages()	4
3.2.2.1	languageConfiguration.exchangeLanguages()	5
3.2.2.2	languageConfig()	5
3.2.2.3	resetTextViews()	6
3.2.2.4	textToTranslate.resignFirstResponder()	6
4	Bonus 2 : Langue du téléphone.....	7
4.1	Comment ça marche ?	7
4.1.1	setupLanguages().....	7
4.1.1.1	languageConfiguration.englishChangingLanguage().....	8

2 Contexte

Une liste de bonus à ajouter sur l'application nous était proposée, avec comme condition d'en choisir un parmi la liste.

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 ».

J'ai également réalisé un autre bonus qui permet à l'interface de s'adapter en fonction de la langue du téléphone.

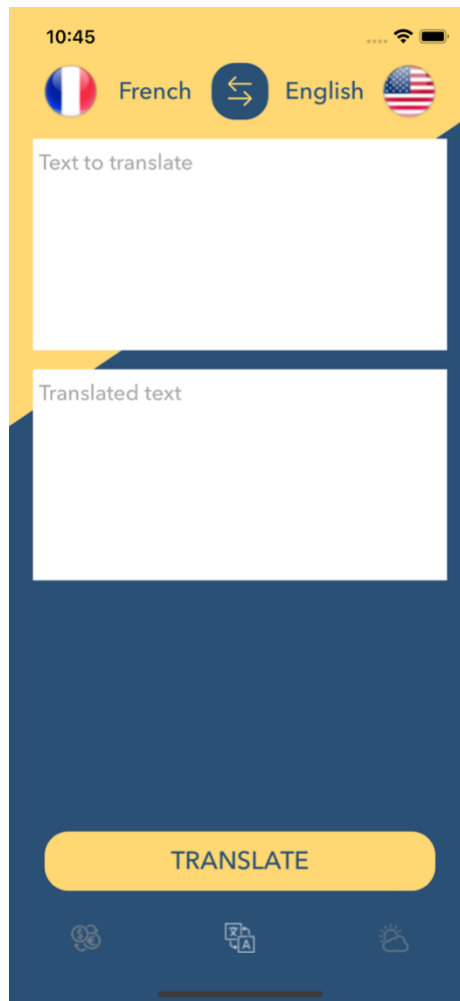
3 Bonus 1 : Inversion des langues

3.1 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 :



3.2 Dans le code

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

3.2.1 exchangeSourceAndDestination()

```
@IBAction func exchangeSourceAndDestination(_ sender: Any) {  
    displayExchangedLanguages()  
}
```

Il s'agit de l'Action du bouton des flèches, dont le rôle sera d'inverser les langues. La fonction `displayExchangedLanguages()` est appelée.

3.2.2 displayExchangedLanguages()

```
private func displayExchangedLanguages() {  
    languageConfiguration.exchangeLanguages()  
    languageConfig()  
    resetTextViews()  
    textToTranslate.resignFirstResponder()  
}
```

Un ensemble de fonction est appelé, nous allons les détailler une à une.

3.2.2.1 *languageConfiguration.exchangeLanguages()*

```
func exchangeLanguages() {  
    // Language changes  
    let tempLanguage = sourceLanguage  
    sourceLanguage = destinationLanguage  
    destinationLanguage = tempLanguage  
  
    // Image changes  
    let tempImage = sourceImageName  
    sourceImageName = destinationImageName  
    destinationImageName = tempImage  
}
```

Comme nous pouvons le constater, il s'agit simplement d'une fonction du Modèle permettant d'inverser les langues source et destination.

Mais pas seulement, on inverse également les images, qui contiennent les drapeaux des deux langues.

3.2.2.2 *languageConfig()*

```
private func languageConfig() {  
    // Outlet's Configuration  
    sourceLabel.text = languageConfiguration.sourceLanguage  
    destinationLabel.text = languageConfiguration.destinationLanguage  
    sourceFlag.image = UIImage(named: languageConfiguration.sourceImageName)  
    destinationFlag.image = UIImage(named: languageConfiguration.destinationImageName)  
}
```

Après l'inversion, on informe nos Outlets de la modification.

3.2.2.3 *resetTextViews()*

```
private func resetTextViews() {  
    // When I want to switch languages, i'm resetting 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.

Les zones de texte retrouvent alors leur configuration de départ.

3.2.2.4 *textToTranslate.resignFirstResponder()*

Quand on switch de langue, j'enlève le focus sur la zone de texte. La raison derrière cette décision est simple.

Si l'utilisateur décide de changer de langue alors que le focus est toujours sur la zone de texte, les langues s'inversent, 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 !

4 Bonus 2 : Langue du téléphone

Ce bonus consiste à détecter la langue du téléphone et ainsi adapter la page traduction de l'application.

Si la langue du téléphone est le français, alors la configuration reste par défaut avec français en langue source, et anglais en langue de destination.

Dans le cas où le téléphone est en anglais, on inverse alors les langues de telle sorte que l'anglais se retrouve en langue source, et le français en destination.

En conclusion, on adapte l'interface pour permettre à une personne française ou anglaise de saisir du texte dans sa langue pour ensuite le traduire.

4.1 Comment ça marche ?

Pour réaliser ce bonus, j'ai créé plusieurs fonctions.

4.1.1 setupLanguages()

```
private func setupLanguages() {  
    languageConfiguration.englishChangingLanguage()  
    languageConfig()  
}
```

La première d'entre-elles, setupLanguages(), présente dans le Contrôleur de traduction, appelle deux autres fonctions.

Elle est elle-même appelée dans le viewDidLoad() du Contrôleur pour faire en sorte que dès qu'on ouvre cette page, tout le processus se lance automatiquement.

4.1.1.1 *languageConfiguration.englishChangingLanguage()*

```
func englishChangingLanguage() {  
    // If the user's language is english, we put english configuration  
    guard let preferredLanguage = preferredLanguage else { return }  
  
    if preferredLanguage == "en" {  
        exchangeLanguages()  
    }  
}
```

Fonction rédigée dans le Modèle, elle permet de vérifier si la variable `preferredLanguage` contient bien la langue préférée de l'utilisateur.

On regarde ensuite si la langue est en anglais et si c'est le cas, on échange les langues source et destination avec la fonction `exchangeLanguages()` présentée dans le Bonus 1.

Une fois que le changement est terminé, la fonction `languageConfig()`, que nous avons également vu au Bonus 1 se lancera afin d'update la Vue avec la bonne configuration de langues.