Introduction à Storyboard et Autolayout

Faire un calendrier

Antoine Moevus

Table des matières

Références

- [Tutorial-calendar] https://www.youtube.com/watch?v=tFbJVQQ9YwM

Vidéos explicatives

- Présentation de la app de calendrier

Planification des requis

Objectifs

On veut une app qui: * Permets à l'utilisateur de visualiser et de sélectionner une date spécifique à partir d'une interface simple et intuitive. * Affiche la date sélectionnée par l'utilisateur d'une manière claire et compréhensible. * Permets à l'utilisateur de personnaliser l'interface du sélecteur de date en fonction de ses préférences. Les options pourraient inclure différentes présentations visuelles du sélecteur de date. * Fournis une expérience utilisateur locale en affichant les informations de date dans le format approprié en fonction de la localisation de l'utilisateur.

Tâches à effectuer

1. **Tâche 1 - Mise en place de l'application de calendrier :** Commencez par créer le projet dans votre environnement de développement. Ajoutez ensuite les composants visuels nécessaires à votre application, à savoir une étiquette pour afficher la date sélectionnée, un sélecteur de date pour choisir une date, et un contrôleur segmenté pour changer l'apparence du sélecteur de date.
2. **Tâche 2 - Configuration du sélecteur de date :** Une fois que vous avez mis en place les composants visuels de base, vous devez ensuite configurer le sélecteur de date pour qu'il affiche les dates dans le format approprié et qu'il réponde aux interactions de l'utilisateur. Il devrait afficher la date du jour par défaut.
3. **Tâche 3 - Réponse à l'interaction de l'utilisateur :** Enfin, vous devez programmer l'application pour qu'elle réponde correctement lorsque l'utilisateur sélectionne une date ou change l'apparence du sélecteur de date. Lorsqu'une date est sélectionnée, l'application doit afficher cette date dans l'étiquette prévue à cet effet. Lorsque l'utilisateur change l'apparence du sélecteur de date, l'application doit mettre à jour le sélecteur en conséquence.

Création de la app

Création du projet

Vidéo explicative: lien

1. Ouvrir XCode, et créer un nouveau projet `iOS` de type `App`
2. Le nommer `calendrierAEC`
3. Choisir `Swift` et `Storyboard`
4. Décocher "Use Core Data"
5. Décocher "Include Tests"

Préparation du `Launchscreen`

Vidéo explicative: lien

Préparation de la vue principale

Vidéo explicative: lien

Création du contrôleur de vue

Vidéos explicatives : * partie 1: le `segmentedControl` * partie 2: Ajout d'événements

Partie 1

IBOutlet et IBAction

`IBOutlet` et `IBAction` sont des attributs spéciaux en Swift utilisés pour connecter votre code avec votre interface utilisateur dans un storyboard.

`IBOutlet` signifie Interface Builder Outlet. Vous utilisez `IBOutlet` pour créer une référence à un composant de l'interface utilisateur à partir de votre code. Par exemple, vous pourriez avoir un `IBOutlet` pour un bouton ou une étiquette dans votre interface utilisateur. Cela vous permet de changer les propriétés de ce composant, comme le texte d'une étiquette ou l'image d'un bouton, directement à partir de votre code.

```
@IBOutlet weak var dateLabel: UILabel!
```

Dans cet exemple, `dateLabel` est un `IBOutlet`. Cela signifie que vous avez une référence à une étiquette UILabel quelque part dans votre storyboard, et vous pouvez utiliser `dateLabel` dans votre code pour obtenir ou définir des propriétés sur cette étiquette.

`IBAction` signifie Interface Builder Action. Vous utilisez `IBAction` pour indiquer qu'une certaine méthode dans votre code doit être appelée en réponse à une action spécifique dans votre interface utilisateur, comme un bouton étant pressé.

```
@IBAction func segCtrlTapped(_ sender: UISegmentedControl) {
    // code to be executed when the segmented control is tapped
}
```

Dans cet exemple, `segCtrlTapped(_:)` est un `IBAction`. Cela signifie que cette méthode sera appelée chaque fois que l'utilisateur tape sur le contrôle segmenté correspondant dans l'interface utilisateur.

Ces deux attributs permettent à votre code de communiquer avec votre interface utilisateur, créant ainsi une interaction dynamique entre l'utilisateur et l'application.

Partie 2: Ajout d'événements

Dans Swift, `@objc` est un attribut utilisé pour indiquer que le membre de code (par exemple, une méthode, une propriété, etc.) suivant est disponible pour Objective-C. C'est souvent nécessaire lorsque vous interagissez avec des API basées sur Objective-C dans votre code Swift, comme UIKit.

Un `selector` est un type en Objective-C qui représente le nom d'une méthode. Dans Swift, vous utilisez `#selector` pour créer un sélecteur à partir d'une méthode. C'est particulièrement utile lorsque vous configurez les actions des contrôles d'interface utilisateur. Par exemple, lorsque vous voulez qu'une certaine méthode soit appelée en réponse à un événement (comme un tap sur un bouton), vous utiliseriez `#selector` pour faire référence à cette méthode.

Dans votre code, `@objc func dateSelected()` signifie que la méthode `dateSelected` est exposée à Objective-C, ce qui permet de l'utiliser comme une action pour `UIControl` (qui est une classe Objective-C).

Et `datePicker.addTarget(self, action: #selector(dateSelected), for: .valueChanged)` ajoute `self` (le ViewController) comme l'objet cible qui recevra l'action `dateSelected` chaque fois que l'événement `valueChanged` se produit pour `datePicker` (c'est-à-dire chaque fois que l'utilisateur change la valeur du datePicker).

La classe Locale

La classe `Locale` en Swift fait partie de la bibliothèque standard Foundation, et elle est utilisée pour fournir des informations spécifiques à la culture lors de la manipulation de composants sensibles à la culture, comme les dates, les heures, les nombres, la devise, etc.

Un `Locale` peut être utilisé pour formater les dates, les nombres et autres, de manière à ce qu'ils soient affichés de manière appropriée pour les utilisateurs de différentes régions du monde. Par exemple, la date et l'heure sont affichées différemment en fonction de la région, tout comme les nombres et la devise.

Dans le code que vous avez partagé, la ligne suivante définit la locale du datePicker à "fr-CA"

(Français - Canada) :

```
datePicker.locale = Locale(identifier: "fr-CA")
```

Cette ligne de code garantit que la date sera affichée conformément aux conventions françaises utilisées au Canada. Par exemple, au Canada, la norme pour le français est d'afficher la date en utilisant le format "jour mois année", alors qu'aux États-Unis, le format typique est "mois jour, année".

C'est une bonne pratique d'utiliser `Locale.current` (forme abrégée: `.current`) pour obtenir la locale actuelle de l'appareil de l'utilisateur, car cela permet à votre application de s'adapter aux préférences de l'utilisateur.

Vérificaiton de la app

Vidéo: lien

À la fin du projet, il faut vérifier que les requis et les objectifs ont été bien remplis.

Exercice

Rajouter un contrôleur segmenté afin de changer la langue (`datePicker.locale`) du calendrier.