Guide pratique

Siri Shortcuts

Principes clés & mise en œuvre



Etude réalisée et diffusée gratuitement par

USERADGENTS



Sommaire

01 **Les Siri Shortcuts, qu’est-ce que c’est ?**
Les principes clés, possibilités & fonctionnement

02 **Les étapes pour créer des Shortcuts**
Definition, donation & handling

03 **Démonstration**
Implémentation de Shortcuts dans une app de test

04 **Raccourci « Ajouter à Siri »**
Où et comment ajouter cette action dans mon app

01

Les Siri Shortcuts, qu’est-ce que c’est ?



La nouveauté iOS 12

Siri Shortcuts

Avec iOS 10, **SiriKit** avait ouvert l'assistant aux applications tierces, pour une sélection des catégories restreintes d'apps : messagerie, paiement mobile, transports...

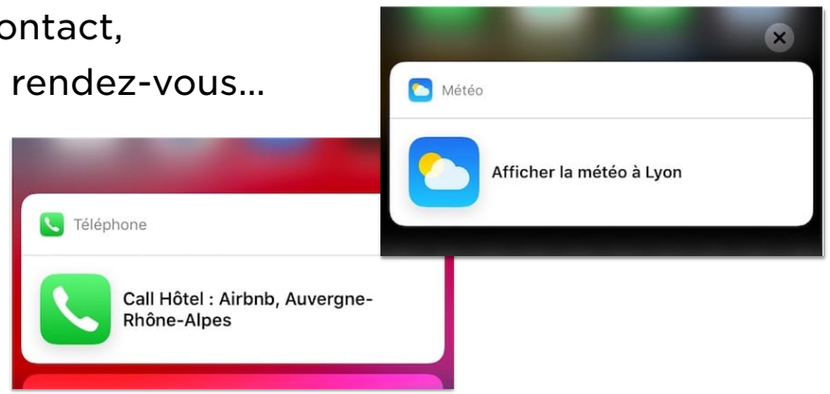
Lors de la WWDC 2018, Apple a présenté Siri Shortcuts sur iOS 12, une manière d'aller plus loin dans l'utilisation de Siri pour **accéder à toutes les fonctionnalités de vos apps**, prédire certains usages et permettre de **créer des routines** à lancer avec Siri, à la voix depuis l'écran verrouillé de son iPhone, iPad ou de son Apple Watch.

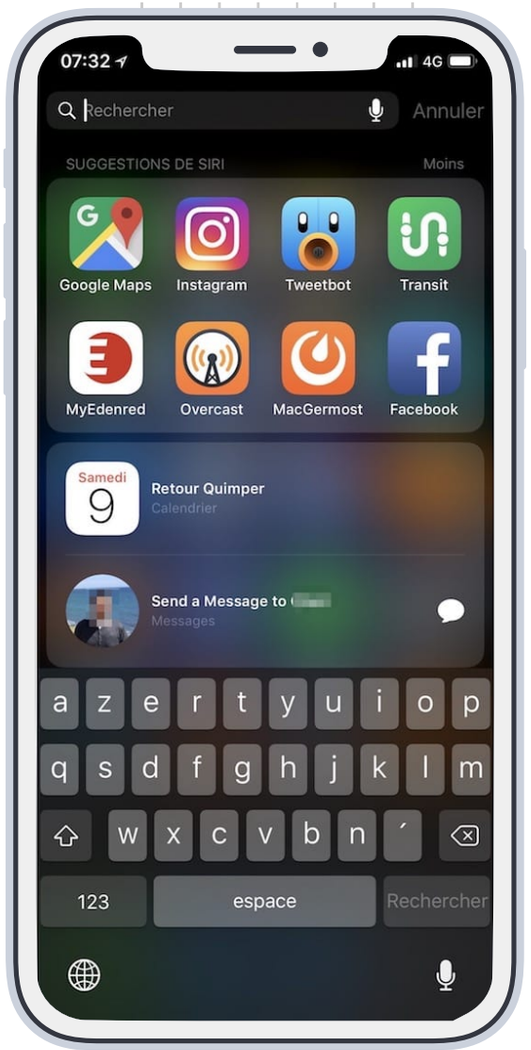
Les notifications proactives

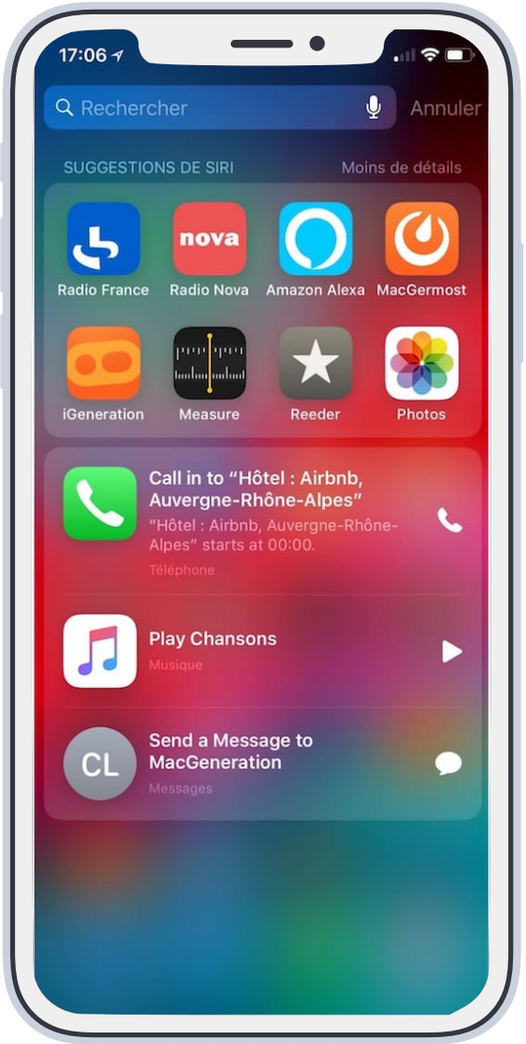
Dans le panneau de recherche Spotlight d'iOS 12

Auparavant Siri ne proposait que des apps dans ses suggestions, il va aujourd'hui plus loin en proposant des intentions d'usage d'apps sous forme d'extension. Ces extensions permettent d'avoir le résultat directement dans Spotlight sans lancer l'app.

+ Envoyer un message à un ami,
+ Connaître la météo,
+ Jouer un morceau,
+ Appeler un contact,
+ Prévenir d'un rendez-vous…





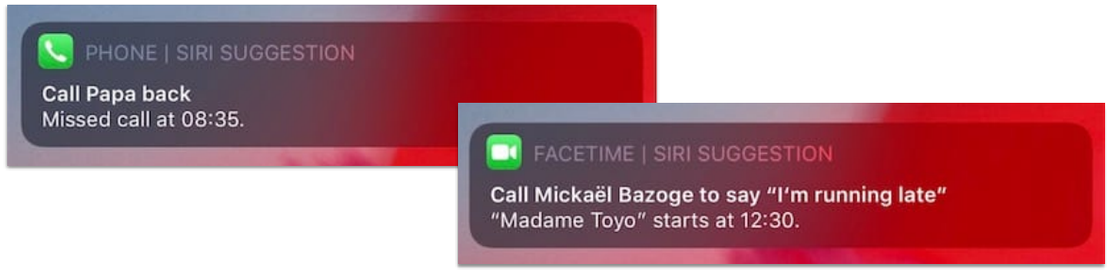


Les notifications proactives

Sur l’écran verrouillé

Quand Siri est absolument certain qu'une suggestion est importante, **une notification apparait sur l'écran verrouillé de l'appareil.**

+ Passer son iPhone en mode « Ne pas déranger » le temps d’une réunion,
+ Rappeler un contact suite à un appel manqué,
+ Prévenir d’un retard à un rendez-vous,
+ Commander son café habituel…





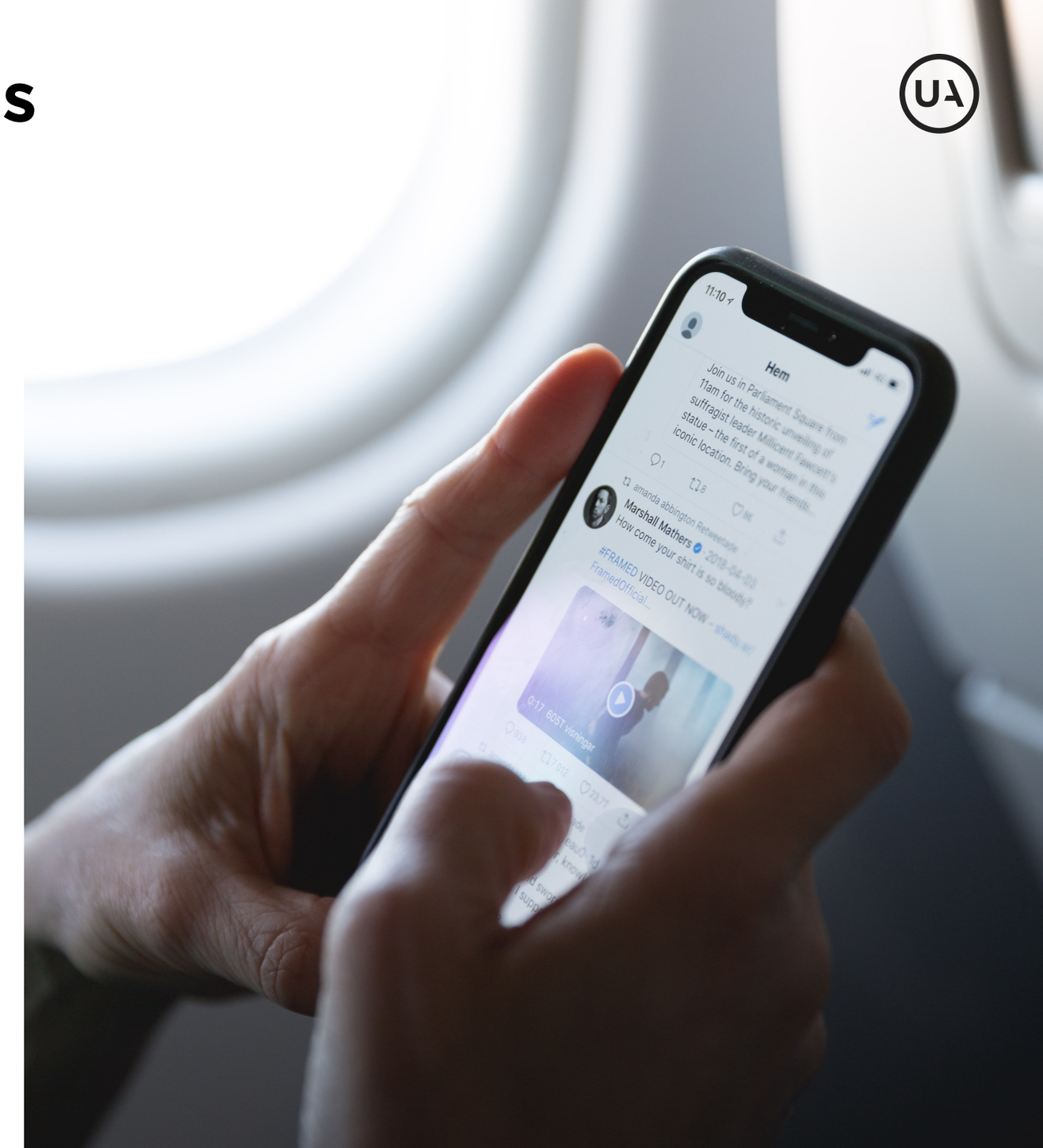
Les notifications proactives

Comment Siri suggère-t-il ?

Pour pousser ces suggestions proactives, Siri exploite des données comme :

+ la **localisation**,
+ **l'heure** de la journée,
+ les **mouvements** de l'utilisateur (marche, course, conduite…),
+ ses **habitudes**.

Cet apprentissage se fait au sein du device, c'est-à-dire de manière **personnalisée** selon les usages des apps de l'utilisateur. Aidé de ses algorithmes d'apprentissage automatique, Siri observe les habitudes et va chercher en profondeur dans les informations données par l'utilisateur pour tenter de trouver un **pattern** et proposer la suggestion la plus pertinente au bon moment.

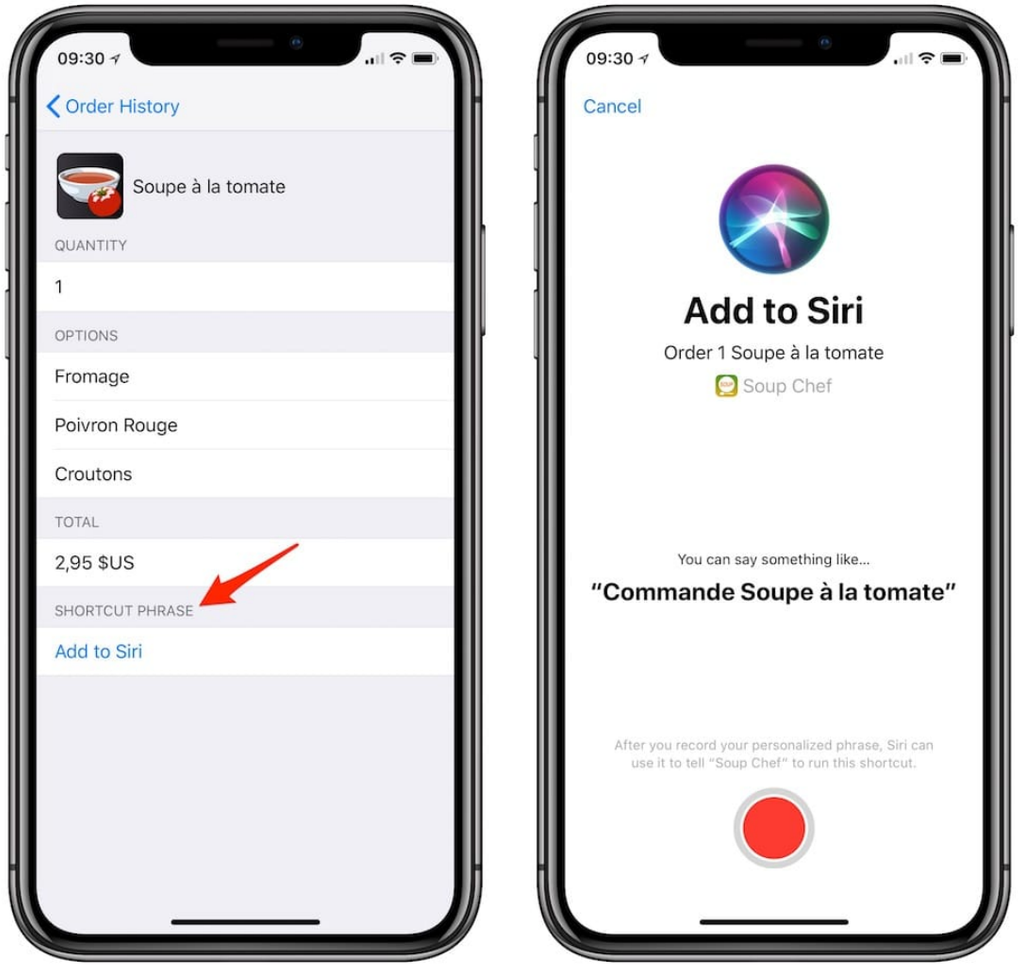


Les notifications proactives

Le rôle des développeurs ?

Avec iOS 12, les développeurs peuvent exploiter un système de **« dons »** à Siri. À chaque fois que l'utilisateur interagit avec une app et effectue une action possible en shortcut, elle peut en fait part à l'assistant pour qu'il puisse par la suite générer une action à l'utilisateur.

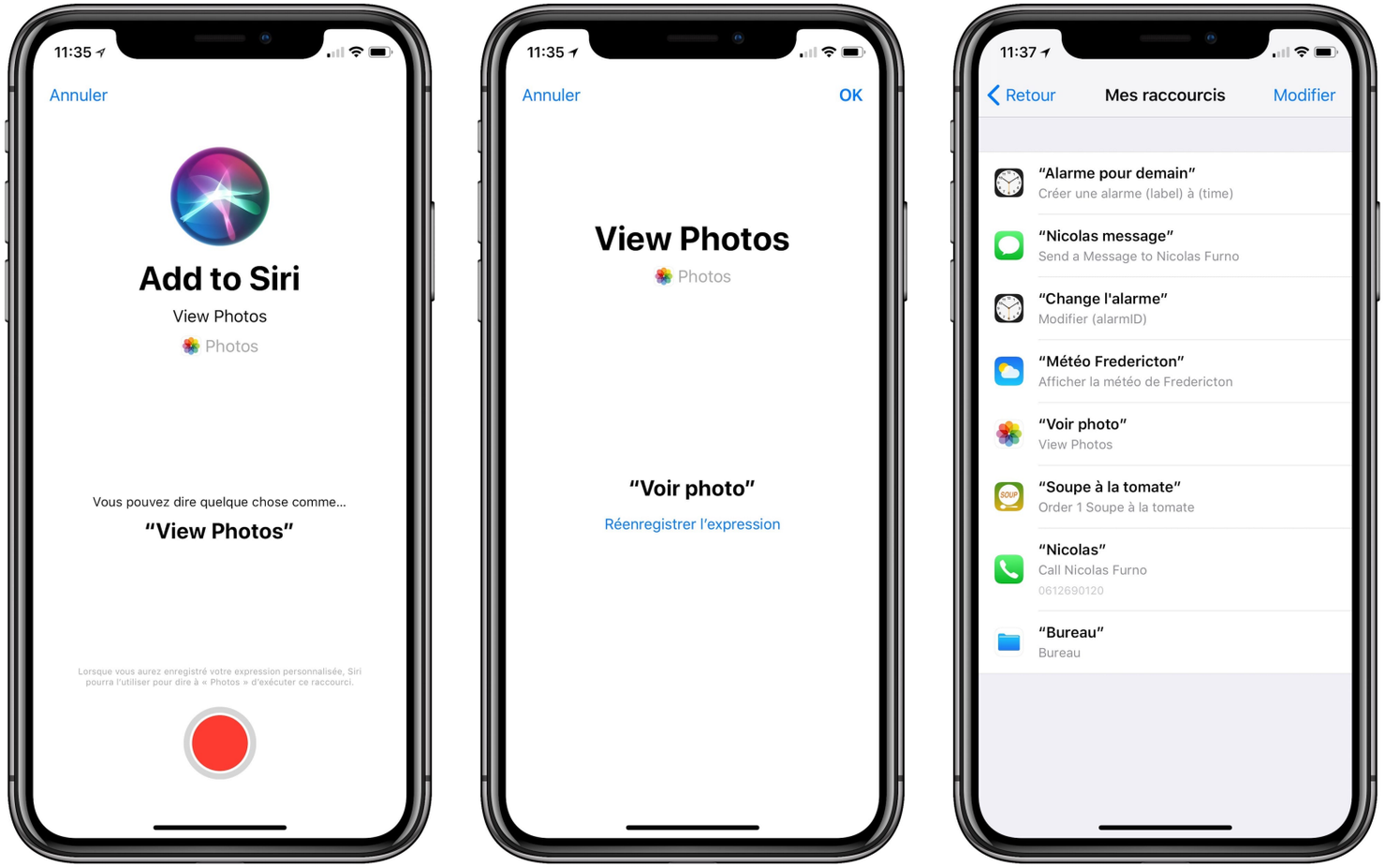
Le développeur peut donner à l'assistant un accès profond à son application pour que Siri puisse faire des suggestions proactives. Il peut également autoriser l'utilisateur à créer des ordres vocaux personnalisés vers des raccourcis de son application comme dans l'exemple de démonstration d'Apple avec l'app test Sous Chef.



Créer ses propres Shortcuts

Via les réglages
Siri & Rechercher

Des raccourcis pré-définis sont proposés dans les réglages « Siri & Rechercher » en fonction des **dernières interactions** de l'utilisateur avec ses apps. Celui-ci n'a plus qu'à **enregistrer un « ordre vocal »** pour activer le shortcut. Siri propose une tournure par défaut mais libre à chacun d'opter pour ce qui lui plait le plus !



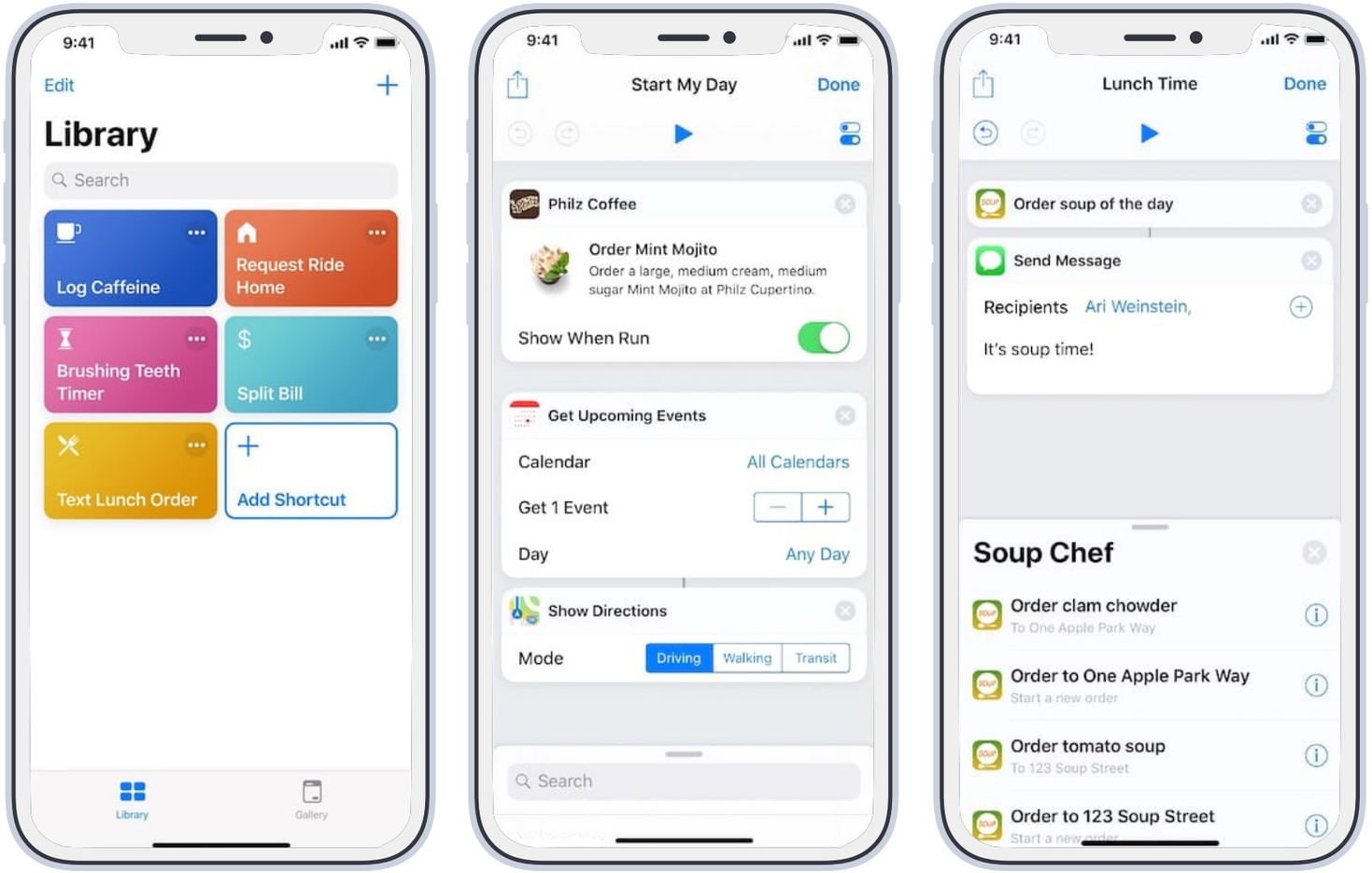
Créer ses propres Shortcuts

Plus tard via l’app Shortcuts

L’app Shortcuts semble être une mise à jour de Workflow rachetée par Apple l’an dernier.

Elle permet aux utilisateurs d’aller plus loin en proposant de concevoir des **scenarii plus complexes**, impliquant plusieurs applications connectées entre elles par des actions.

La version finale de l’app ne sortira qu’à l’automne.



L’UI des Shortcuts

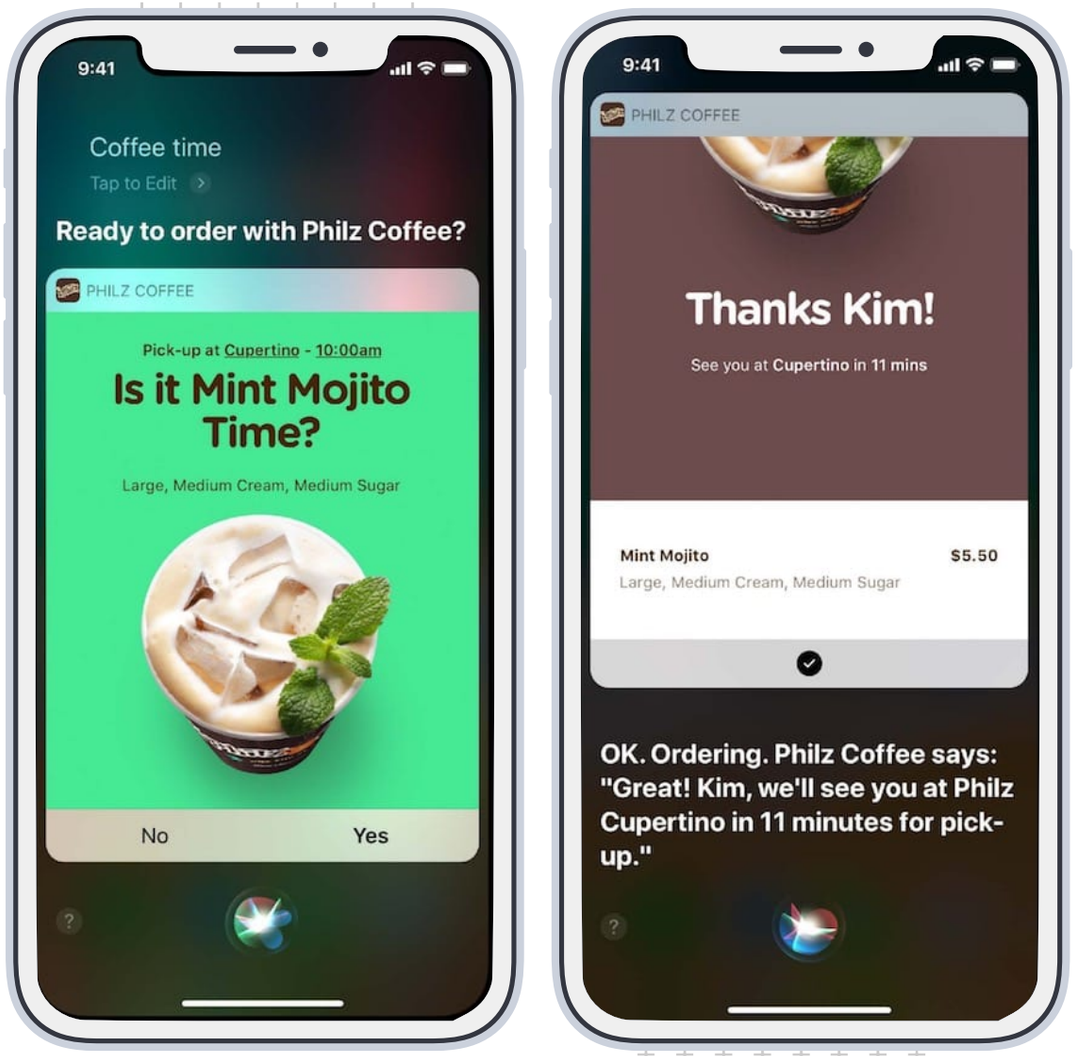
Quelle réponse à l’activation d’un shortcut ?

Lorsque l’utilisateur invoque Siri pour lui demander d’effectuer un shortcut, l’assistant peut afficher **une carte et une réponse personnalisée**…

Le développeur doit ainsi prendre soin de préciser la phrase que Siri devra énoncer en réponse, et designer la carte qui s’affichera en simultané sur l’écran.

Par exemple, dans le cadre d’une commande de café, après confirmation de l’utilisateur, Siri pourra dire dans combien de temps celui-ci sera prêt.

Dans le cas contraire, il sera possible de simplement ouvrir l’app pour payer sa commande par exemple et choisir la carte bancaire à utiliser.



Concevoir utile

Pour un shortcut utile :

+ Réfléchissez aux **fonctionnalités clés** de votre app, les plus utilisées / appréciées par vos utilisateurs

+ Imaginez un moyen de leur procurer ce qu'ils souhaitent **plus rapidement**

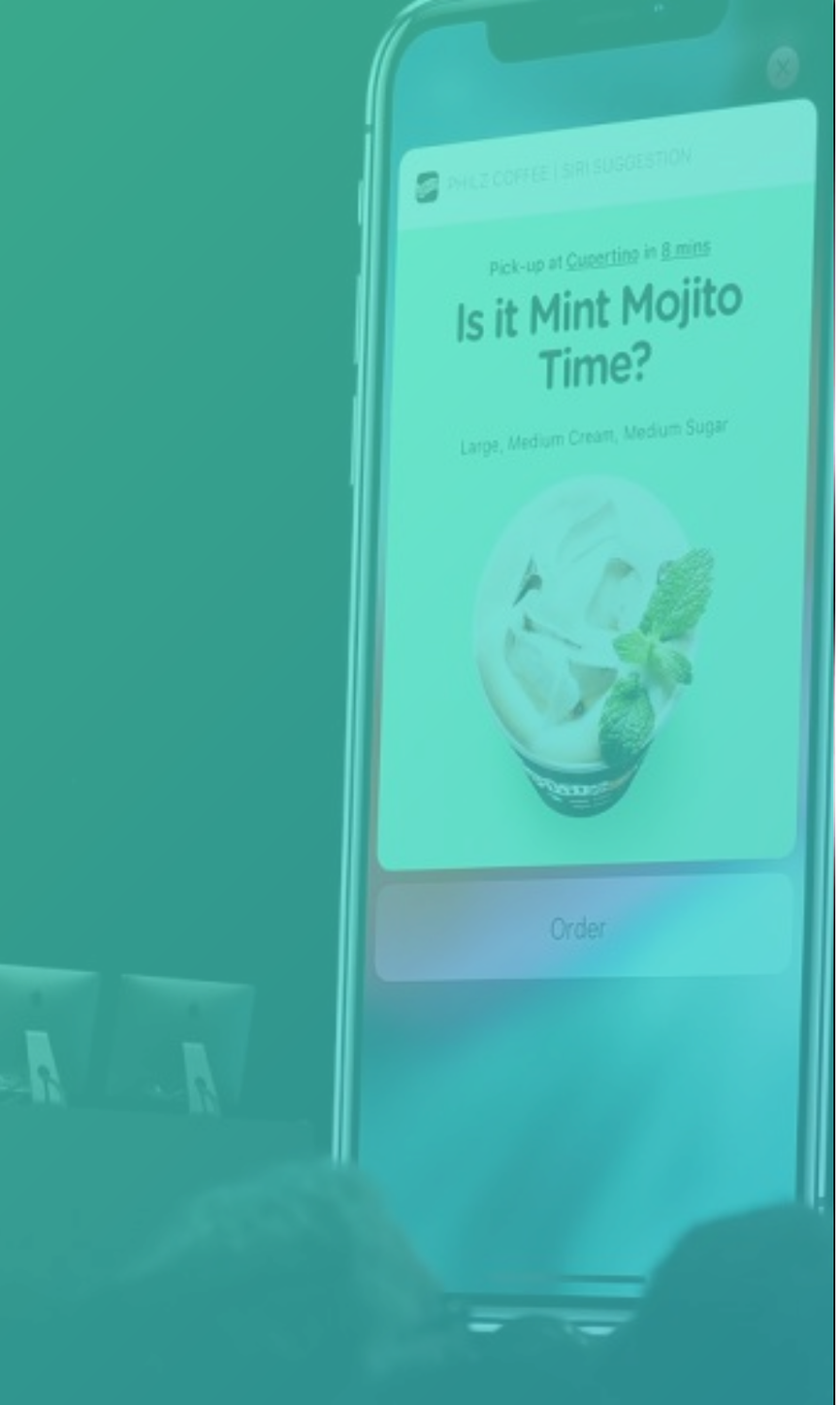
+ Proposez des shortcuts qui seront **susceptibles d'être utilisées régulièrement** et non pas une ou deux fois de temps en temps

+ L'ajout de l'icône « Add to Siri » au sein de votre application **n'est pas obligatoire**. Faites des « dons » ciblés et laissez iOS se charger du reste !



02

Les étapes pour concevoir un shortcut



Shortcuts APIs

Il existe 2 APIs pour concevoir des shortcuts :

+ NSUserActivity

Solution simple qui s'intègre à d'autres comme Spotlight ou Handoff, mais qui ne permet pas de customiser le Shortcut.

Permet uniquement d'ouvrir l'app lors de l'affichage du Shortcut.

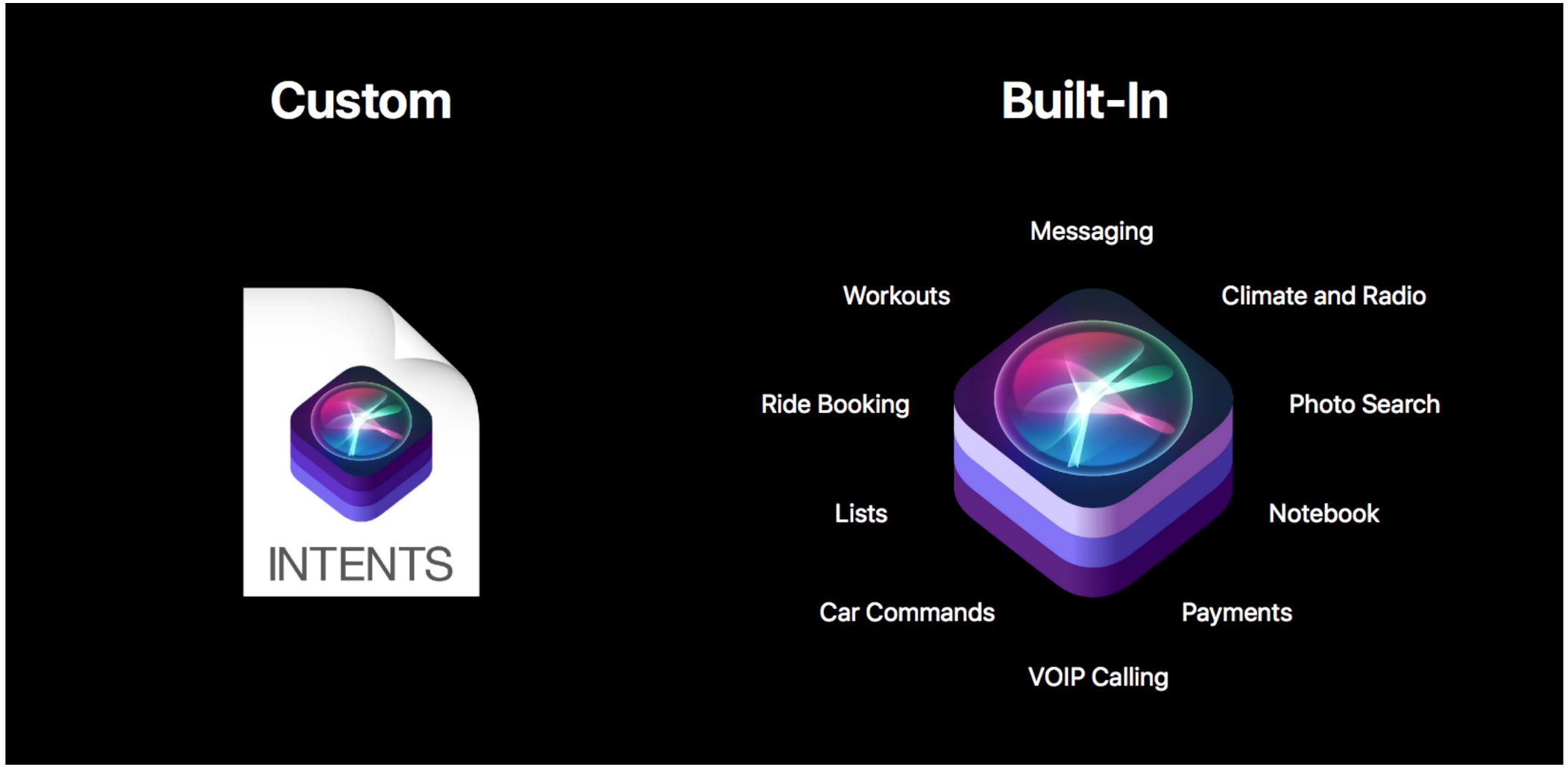
+ Intents

Solution la plus complète qui permet un contrôle total sur le rendu, l'affichage et la catégorisation des Shortcuts.

Permet notamment l'exécution d'une extension de votre app sans devoir la lancer.

Dans la suite de la présentation, nous présenterons l'utilisation des **Intents**.

Types d’intentions (Intents)



En fonction de votre implémentation, vous pouvez choisir entre utiliser des catégories intégrées à iOS, ou développer votre propre type d’intention.

Adopter l’API Shortcuts

La création d’une shortcut s’effectue en **3 étapes** :

1



Définir ce qu’il faudra présenter comme un Shortcut et l’étendue des fonctionnalités pour que Siri soit notifié de l’intégralité de ce que l’app peut faire.

2



Informer le système lorsque l’utilisateur utilise une fonctionnalité de l’app afin que Siri sache quand et comment la suggérer par la suite.

3



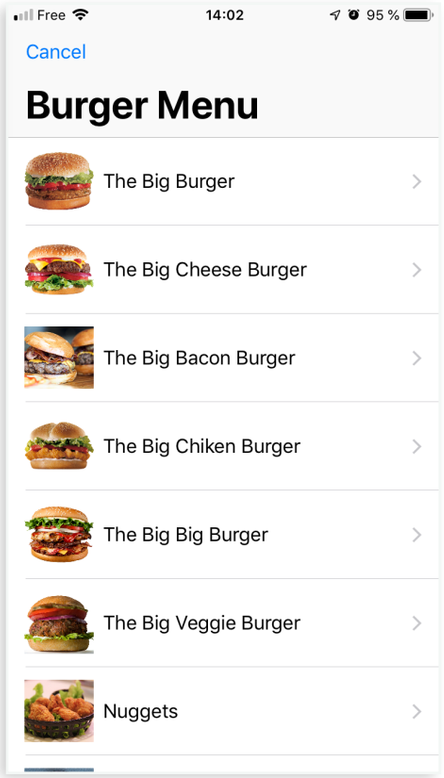
Développer la logique qui sera appelée lorsque l’utilisateur utilisera le ShortCut, notamment si une extension est utilisée sans lancer l’app.

03

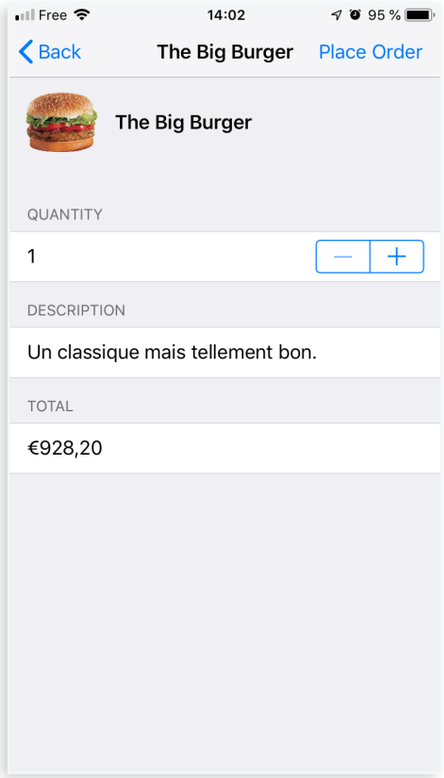
Démonstration

Pré-requis

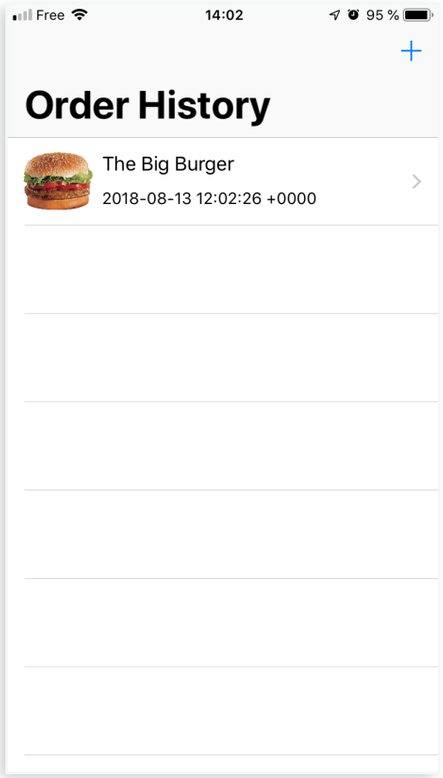
Pour vous présenter l'implémentation de Siri Shortcuts, nous avons développé une application très simple qui permet de commander des burgers. L'application contient 3 écrans ci-dessous. L'objectif est d'intégrer Siri Shortcuts dans l'app pour simplifier la commande d'un burger, et promouvoir son utilisation.



Liste des burgers



Page produit



Historique de commandes

Rappel : Adopter l’API Shortcuts

La création d’une shortcut s’effectue en **3 étapes** :

1



Définir ce qu’il faudra présenter comme un Shortcut et l’étendue des fonctionnalités, pour que Siri soit notifié de l’intégralité de ce que l’app peut faire.

2



Informer le système lorsque l’utilisateur utilise une fonctionnalité de l’app afin que Siri sache quand et comment la suggérer par la suite.

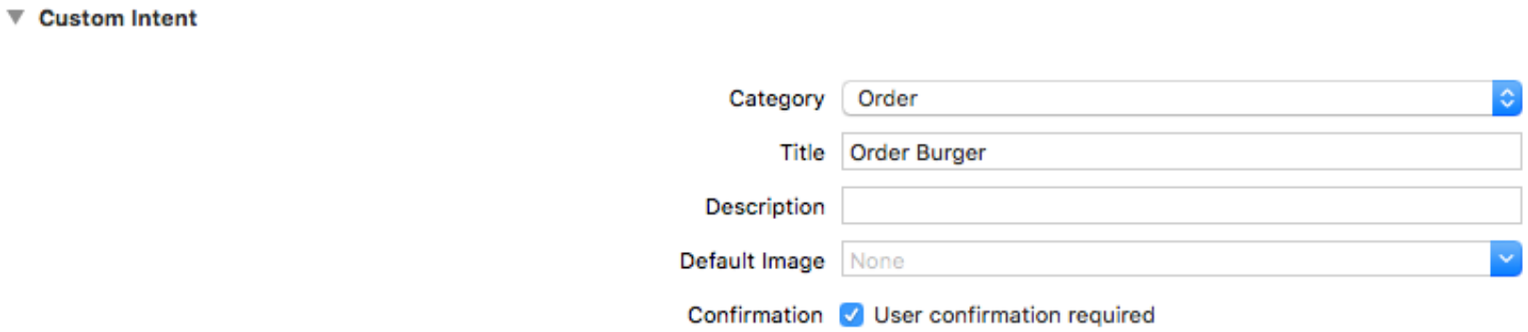
3

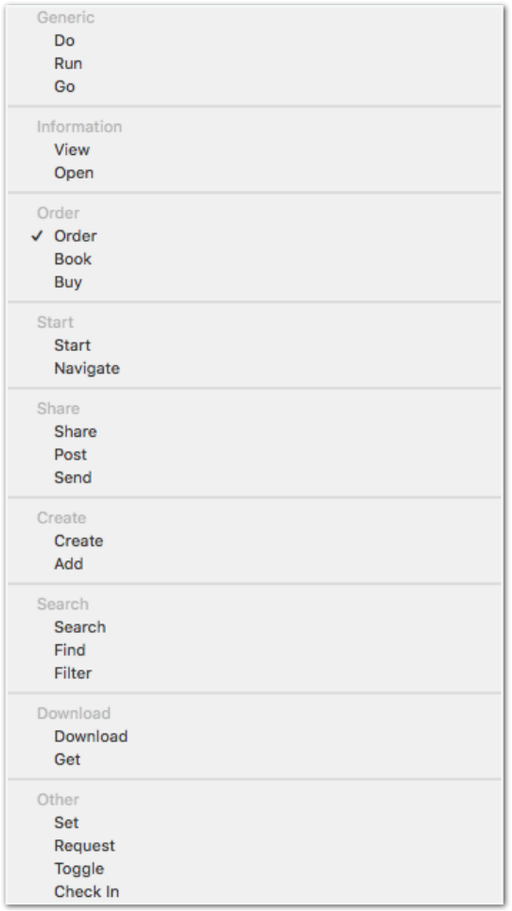


Développer la logique qui sera appelée lorsque l’utilisateur utilisera le ShortCut, notamment si une extension est utilisée.

1. Define ShortCut

Dans **Xcode** création d'un nouveau fichier « **SiriKit Intent Definition File** »



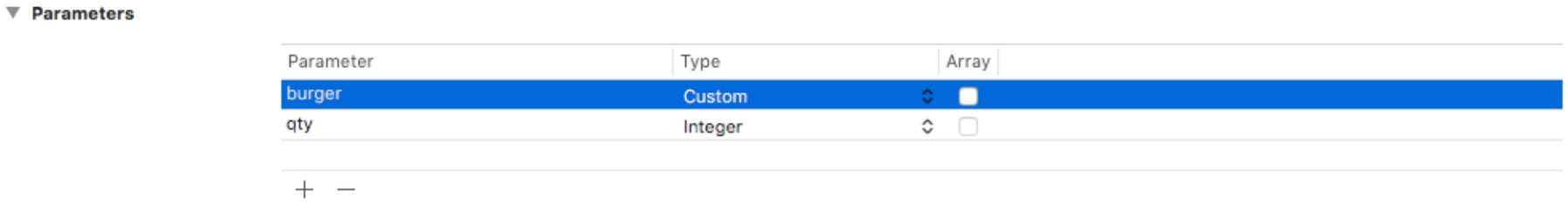


La première section contient les metadatas de l'intent, dont la **catégorie**. La liste des catégories possibles est disponible à droite. Celle-ci permet d'aider Siri à configurer l'intent, et à afficher des éléments UI contextuels à celle-ci.

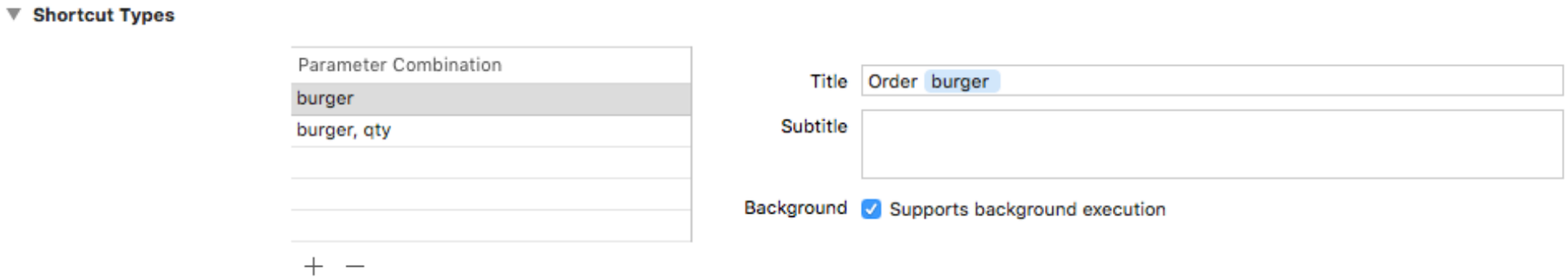
Par exemple, dans la catégorie « Order », Siri affichera un bouton « acheter » sur l'écran verrouillé, et confirmera l'intention par « Ok je l'ai acheté ».

1. Define ShortCut

Le ShortCut peut aussi prendre des paramètres en entrée :



Les paramètres de notre démo sont ici **burger** comme un élément du menu qui est de type *custom* et **qty** avec comme type un *int*.



Chaque type de Shortcut inclue des combinaisons de paramètres qui seront prédits et permettront de formater l'affichage. Dans ce cas, Siri peut prédire à la fois 3 combinaisons :

- acheter sans le type de burger par défaut
- acheter un burger précisé sans quantité, et l'ext. de l'app se chargera de la mettre à 1 par exemple
- acheter un burger précisé avec une quantité précisée aussi

1. Define ShortCut

Xcode va ensuite générer automatiquement le code suivant :

```
public class OrderBurgerIntent: INIntent {

    @NSManaged public var burger: INObject?
    @NSManaged public var qty: NSNumber?

}

public protocol OrderBurgerIntentHandling: NSObjectProtocol {
    func handle(intent: OrderBurgerIntent, completion: @escaping (OrderBurgerIntentResponse) -> Swift.Void)

    optional func confirm(intent: OrderBurgerIntent, completion: @escaping (OrderBurgerIntentResponse) -> Swift.Void)
}
```

2. Donate shortcut



À chaque fois qu'un utilisateur va effectuer une action équivalent au Shortcut dans l'application, il est nécessaire de l'indiquer à Siri. Ce process s'appelle « donate » le shortcut. Dans notre cas, nous allons le faire à chaque fois que l'utilisateur passe une commande :

```
extension Order {
    public var intent: OrderBurgerIntent {
        let intent = OrderBurgerIntent()
        intent.burger = INObject(identifier: burger.ref, display: burger.title)
        intent.qty = NSNumber(value: qty)
        return intent
    }
}
```

Création de l'interaction entre intent et donate pour aider Siri dans ses prédictions :

```
// Donate an interaction to the system
let interaction = INInteraction(intent: order.intent, response: nil)
interaction.donate { (error) in
    if let error = error {
        print("Did Fail: \(error.localizedDescription)")
    }
}
```

3. Handle shortcut



Convertissez l'intent en un order object (si **qty** n'est pas fourni, alors on ajoute 1 par défaut) :

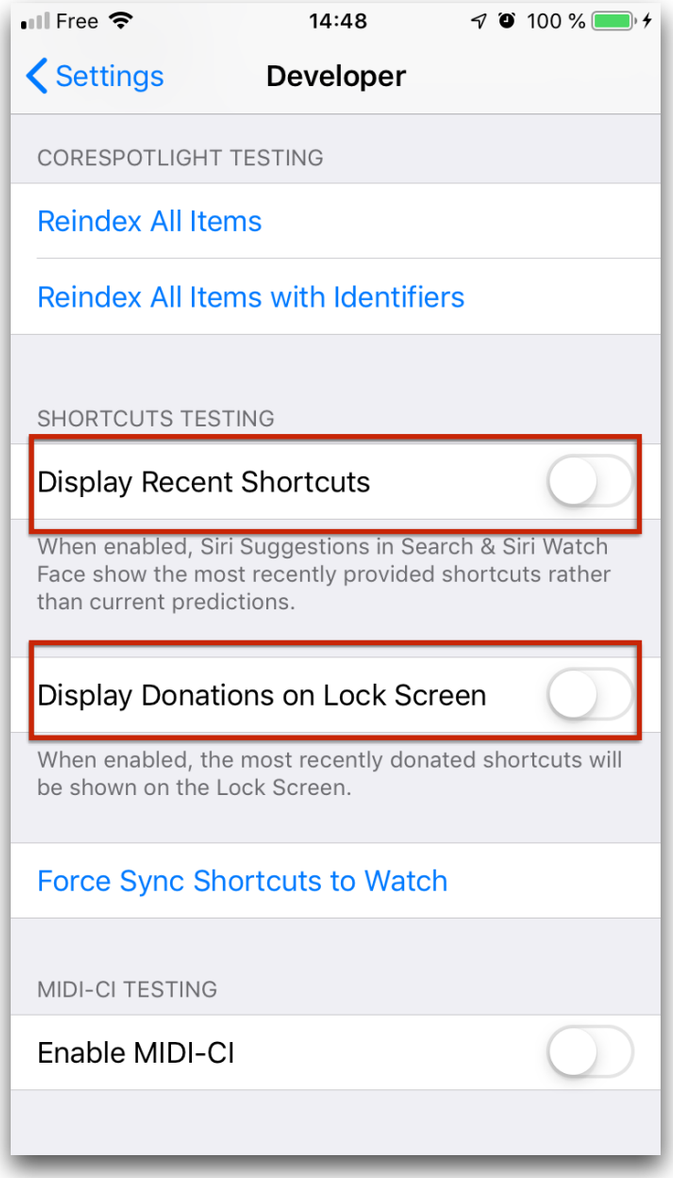
```
extension Order {
    public init?(from intent: OrderBurgerIntent) {
        let menuService = MenuService()
        guard let burgerRef = intent.burger?.identifier,
            let burger = menuService.burger(with: burgerRef)
            else { return nil}
        self.init(burger: burger, qty: intent.qty?.intValue ?? 1, date: Date())
    }
}
```

L'interaction entre l'utilisateur et le ShortCut se trouve dans la méthode suivante de l'**AppDelegate :**

```
    func application(_ application: UIApplication, continue userActivity: NSUserActivity,
restorationHandler: @escaping ([UIUserActivityRestoring]?) -> Void) -> Bool {
        if userActivity.activityType == "OrderBurgerIntent" {
            guard let intent =  userActivity.interaction?.intent as? OrderBurgerIntent,
                let order = Order(from: intent) else { return false }
            presentOrder(order)
        }
        return true
    }
```

Vérifiez le type d'activité, convertissez l'intent et redirigez l'utilisateur vers l'écran de commande.

Démonstration

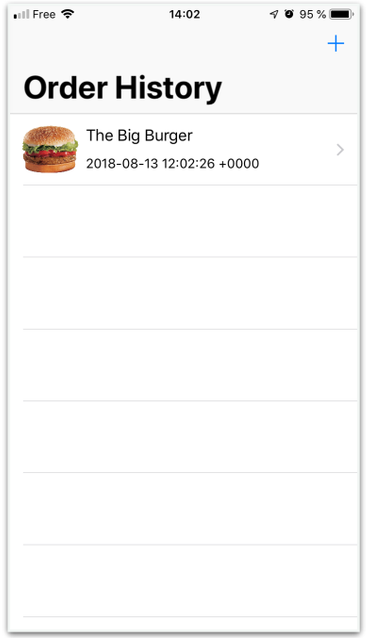


Pour le debug uniquement, il est possible de forcer Siri à afficher systématiquement les Shortcuts dans le panneau de notifications afin de les tester. Il est aussi possible d'afficher les « Donations » sur l'écran verrouillé.

Cet écran se trouve dans : **Réglages** —> **Développeur**

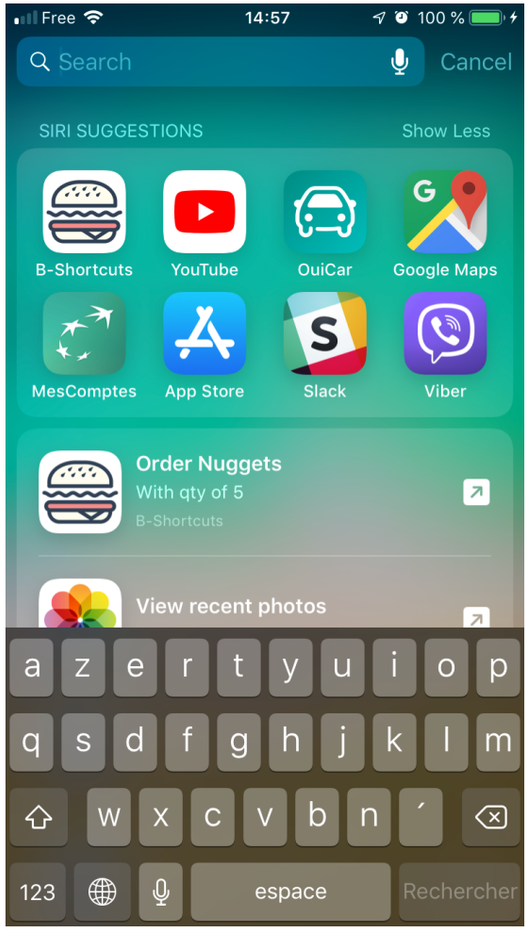
Démonstration

Une fois l'application installée, le comportement est le suivant après avoir déjà réalisé une commande :



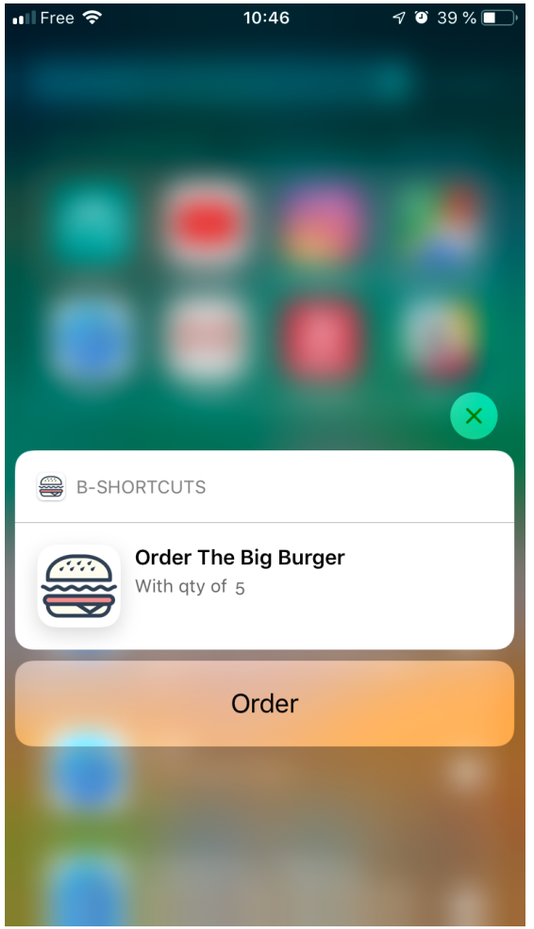
Pré-requis : passer une première commande
(cf. historique des commandes)

plus tard...



Siri détecte qu'il est l'heure de manger un burger

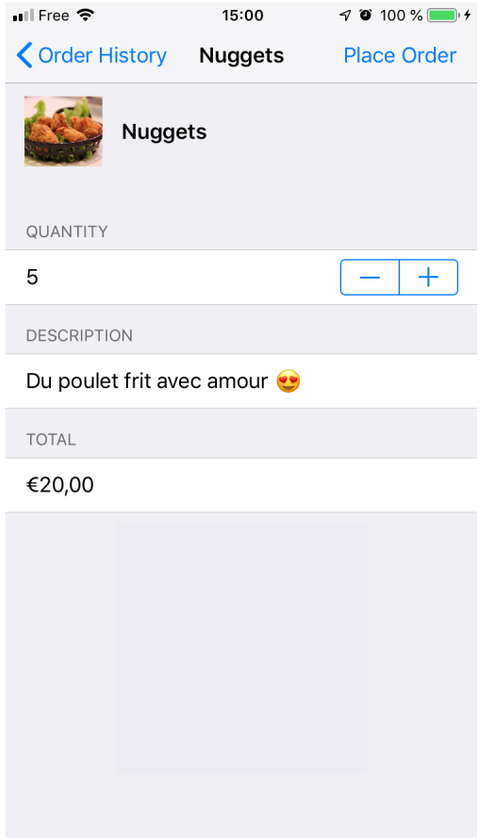
au click



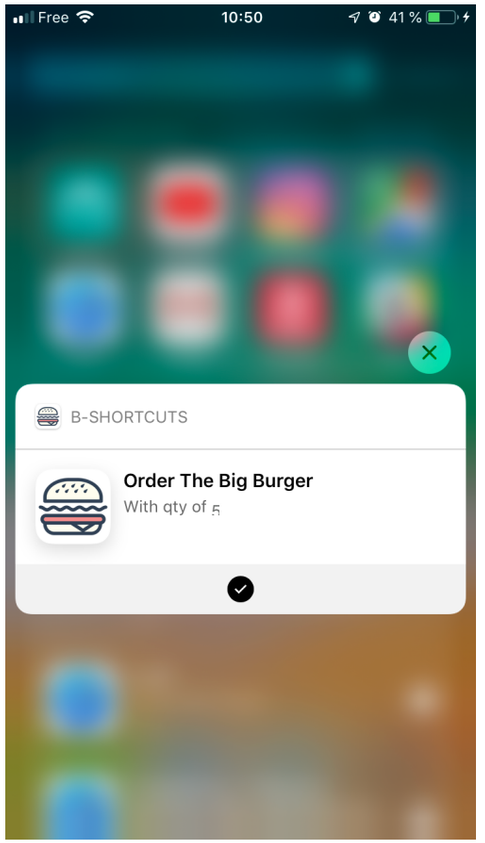
Affichage de la fiche standard d'un type « Order »

Démonstration

2 solutions s’offrent ensuite à l’utilisateur :



Tap sur la fiche Shortcut pour ouvrir l’application. L’utilisateur peut ensuite changer la quantité par exemple, ou rajouter des produits.



Tap directement sur le bouton « Commander » pour lancer l’exécution du code associé. Ici la commande avec la carte enregistrée dans le compte (cf. slide suivant).

Optionnel : Customiser le Shortcut

En utilisant un « intent extension » (cf. précédent), le développeur peut définir ce qu'il affichera

1) Dans **Xcode** créez une nouvelle target « **Intents Extension** »
2) Assurez vous que tous les fichiers nécessaires sont inclus dans la nouvelle target
3) Ajoutez cette target dans le même groupe que le reste de votre code pour lui permettre d'accéder à l'ensemble des données de l'application (Target -> Capabilities -> App Group)
4) Ajoutez ensuite ce type de code d'extension :

```
class IntentHandler: INExtension, OrderBurgerIntentHandling {

    func handle(intent: OrderBurgerIntent, completion: @escaping (OrderBurgerIntentResponse) -> Void) {
        if let order = Order(from: intent) {
            let orderService = OrderService()
            orderService.save(order: order)
            completion(OrderBurgerIntentResponse(code: .success, userActivity: nil))
        } else {
            completion(OrderBurgerIntentResponse(code: .failure, userActivity: nil))
        }
    }
}
```

IntentHandler doit se confirmer au **OrderBurgerIntentHandling** *protocol* (généré automatiquement par Xcode). Dans la *handler* méthode, essayer de créer une commande depuis l'*intent*, et si ok, appelez la *completion* avec le code *success*.

04

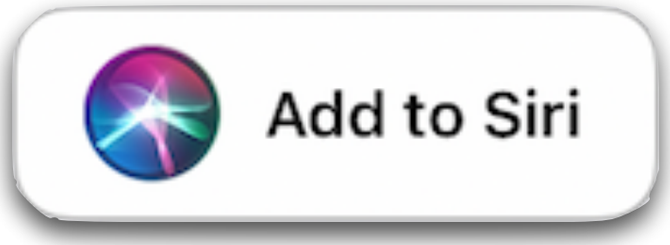
Bouton « Ajouter à Siri »

Optionnel : Rajouter un bouton « Add to Siri »

Nous l'indiquions précédemment, le bouton « Add to Siri » n'est pas obligatoire pour exploiter les Shortcuts. Apple se chargera de définir les récurrences d'usage grâce aux « dons ».

Néanmoins Apple a prévu de laisser la possibilité aux développeurs d'ajouter un bouton « Add to Siri » pour que les utilisateurs déclenchent une action eux-mêmes sans attendre que Siri détecte que l'action doit être lancée.

Ce bouton permet d'associer une phrase dictée par l'utilisateur pour évoquer le shortcut directement depuis Siri.



NOTA : Pour tous les Shortcuts existants, un utilisateur peut associer une phrase custom à Siri en allant dans **Réglages —> Siri & recherche** (section Shortcuts)

Optionnel : Rajouter un bouton « Add to Siri »

Pour ajouter un bouton « Ajouter à Siri », Apple met à disposition un kit graphique embarqué dans iOS qui permet de respecter l'UI Apple. Le bouton peut ainsi être sur fond blanc, fond noir et avec/sans bordure.

Pour commencer, ajouter une vue **INUIAddVoiceShortcutButton** depuis le framework **IntentsUI** à votre *layout*.

Pour afficher le controller de configuration Siri, il faut utiliser **INUIAddVoiceShortcutViewController.**

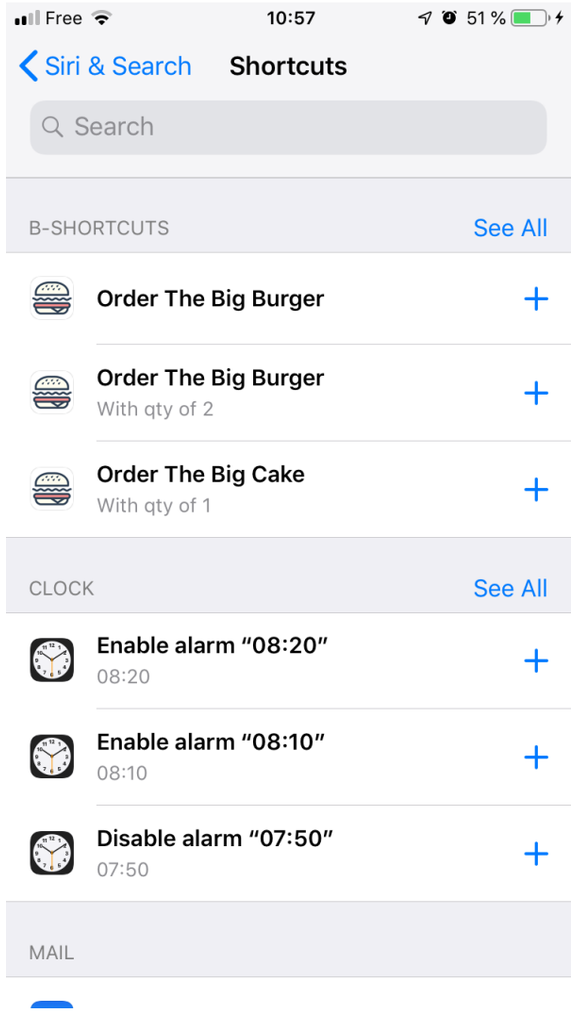
```
let addVoiceShortcutVC = INUIAddVoiceShortcutViewController(shortcut: shortcut)
addVoiceShortcutVC.delegate = self
present(addVoiceShortcutVC, animated: true, completion: nil)
```

Et se conformer aux 2 méthodes *delegate* suivantes :

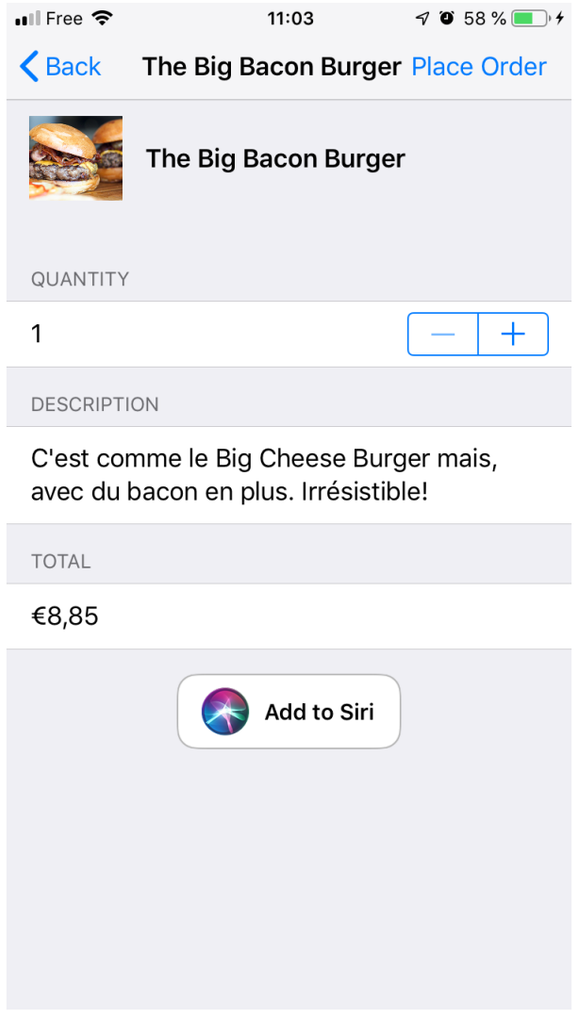
```
    func addVoiceShortcutViewController(_ controller: INUIAddVoiceShortcutViewController, didFinishWith voiceShortcut:
INVoiceShortcut?, error: Error?)

    func addVoiceShortcutViewControllerDidCancel(_ controller: INUIAddVoiceShortcutViewController)
```

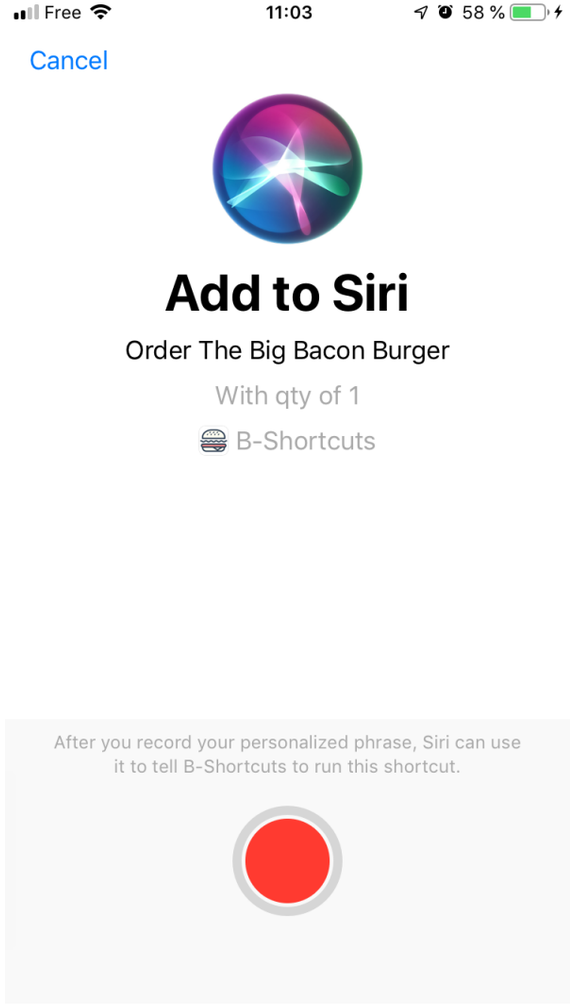
Optionnel : Rajouter un bouton « Add to Siri »



Association d’une phrase à un shortcut existant.

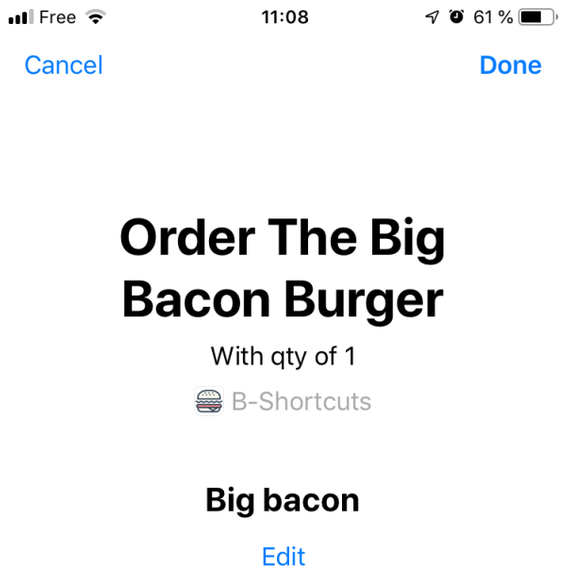


Exemple d’intégration du bouton dans une page.



Ecran d’enregistrement de la phrase.

Optionnel : Rajouter un bouton « Add to Siri »



La phrase est enregistrée

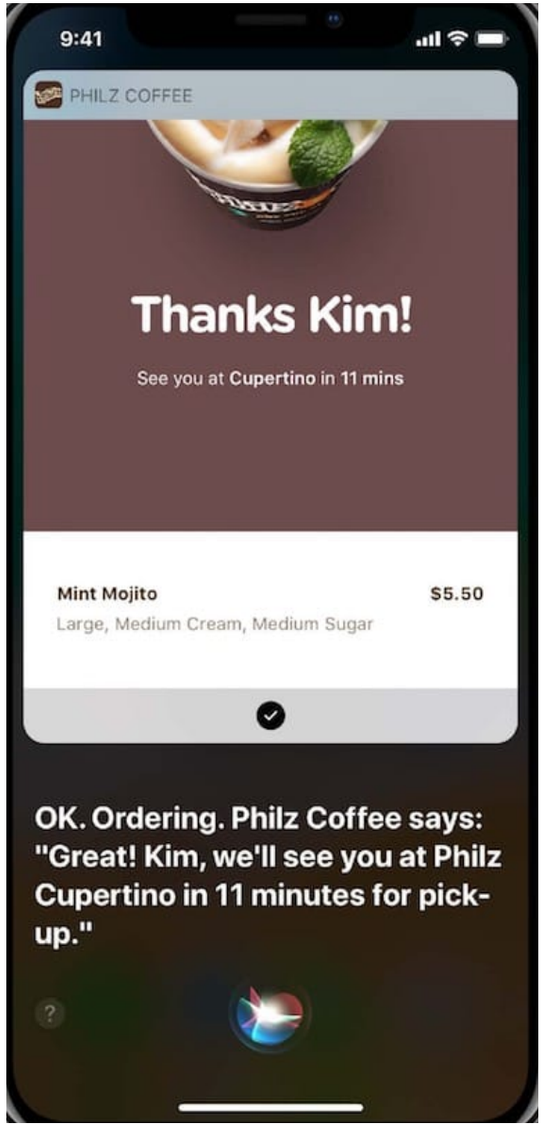


Invocation de Siri avec la phrase « Big Bacon »



La commande est passée depuis Siri.

Optionnel : Rajouter un bouton « Add to Siri »



Lorsque l'application n'est pas invoquée, l'écran de confirmation de Siri, peut aussi être personnalisé, et afficher des informations à l'utilisateur ou même des images.

Exemple ci-dessous avec un écran de confirmation spécifique à l'application Phil.z Coffee, ainsi qu'une réponse de Siri contenant des informations additionnelles comme l'heure de livraison par exemple.

Merci !

Nicolas BENOIST
CTO
n.benoist@useradgents.com

Oleg VOITENKO
iOS Lead Developer
o.voitenko@useradgents.com

Marie BILLON
Chargée d'Études Digitales
m.billon@useradgents.com



USERADGENTS

8 rue de la Rochefoucauld - 75009 Paris
01 77 75 65 90
www.useradgents.com