Persistance des données

Longue vie aux données !

Persistance des données

- Property Lists
- Les préférences
- L’archivage
- Accès au système de fichiers
- CloudKit
- Core Data

Property Lists

Property Lists

- Fichiers .plist
- Stocke des objets sérialisés
- Permet de stocker des tableaux ou des dictionnaires d’objets “standards”
  - (NS)Data, (NS)String, NSNumber (Int/Double/Float), (NS)Date, (NS)Array, (NS)Dictionary
- Il faut savoir lire et écrire sur le disque !

Property Lists

- NSArray ou NSDictionary
- `convenience init?(contentsOf url: URL)`
- `func write(url: URL, atomically: Bool) -> Bool`

Les préférences

Les préférences

- (NS)UserDefaults
  - Utilisé pour stocker des petites quantités d’informations
  - Basé sur un fichier Property List
  - Stocké dans le dossier Library/Preferences de l’app

- Permet de stocker :
  - des types courants (Int / Float / Double / Bool / (NS)URL)
  - des objets compatibles plist ((NS)Data, (NS)Date, (NS)Array, (NS)Dictionary
- Garde un cache en mémoire pour les réglages
  - Cache synchronisé automatiquement avec le système de fichier

- `class func standard() -> NSUserDefaults`
- `func set(_ value: Bool, forKey defaultName: String)`
- `func set(_ value: Float, forKey defaultName: String)`
- `...`
- `func object(forKey: String) -> Any?`
- `func bool(forKey: String) -> Bool`
- `...`

```swift
let prefs = UserDefaults.standard
prefs.set(true, forKey: "UserAgreed")
prefs.set("Ludovic", forKey: "firstName")

let firstName = prefs.string(forKey: "firstName")
print(firstName)
```

```
Optional("Ludovic")
```

L’archivage

L’archivage

- Permet d’écrire des objets sur le disque
- Méthode utilisée par Xcode pour écrire vos UI/Storyboards
- Permet d’enregistrer n’importe quel type de graphe d’objet
  - Du moment que les objets se conforment au protocole NSCoding
    - Ajout de méthodes dans la classe.
    - Classe qui hérite de NSObject pour le bon fonctionnement

NSCoding

- Méthodes obligatoires :
- `func encode(with aCoder: NSCoder)`
- `init(coder aDecoder: NSCoder)`

NSCoding

```
@objc func encode(with aCoder: NSCoder) {

    aCoder.encode(age, forKey: "age")
    aCoder.encode(name, forKey: "name")
    aCoder.encode(size, forKey: "size")
    aCoder.encode(gender, forKey: "gender")
}

@objc required init(coder aDecoder: NSCoder) {

    age = aDecoder.decodeInt(forKey: "age")
    name = aDecoder.decodeString(forKey: "name")
    size = aDecoder.decodeFloat(forKey: "size")
    gender = aDecoder.decodeString(forKey: "gender")
}
```

L’archivage

- NSKeyedArchiver
  - Archive un graphe d’objet dans du NSData
  - `class func archivedData(withRootObject: Any) -> NSData`
- NSKeyedUnarchiver
  - Désarchive un graphe d’objet d’un objet NSData
  - `class func unarchiveObject(withData: NSData) -> Any?`
- Il faut savoir lire et écrire du NSData sur le disque !

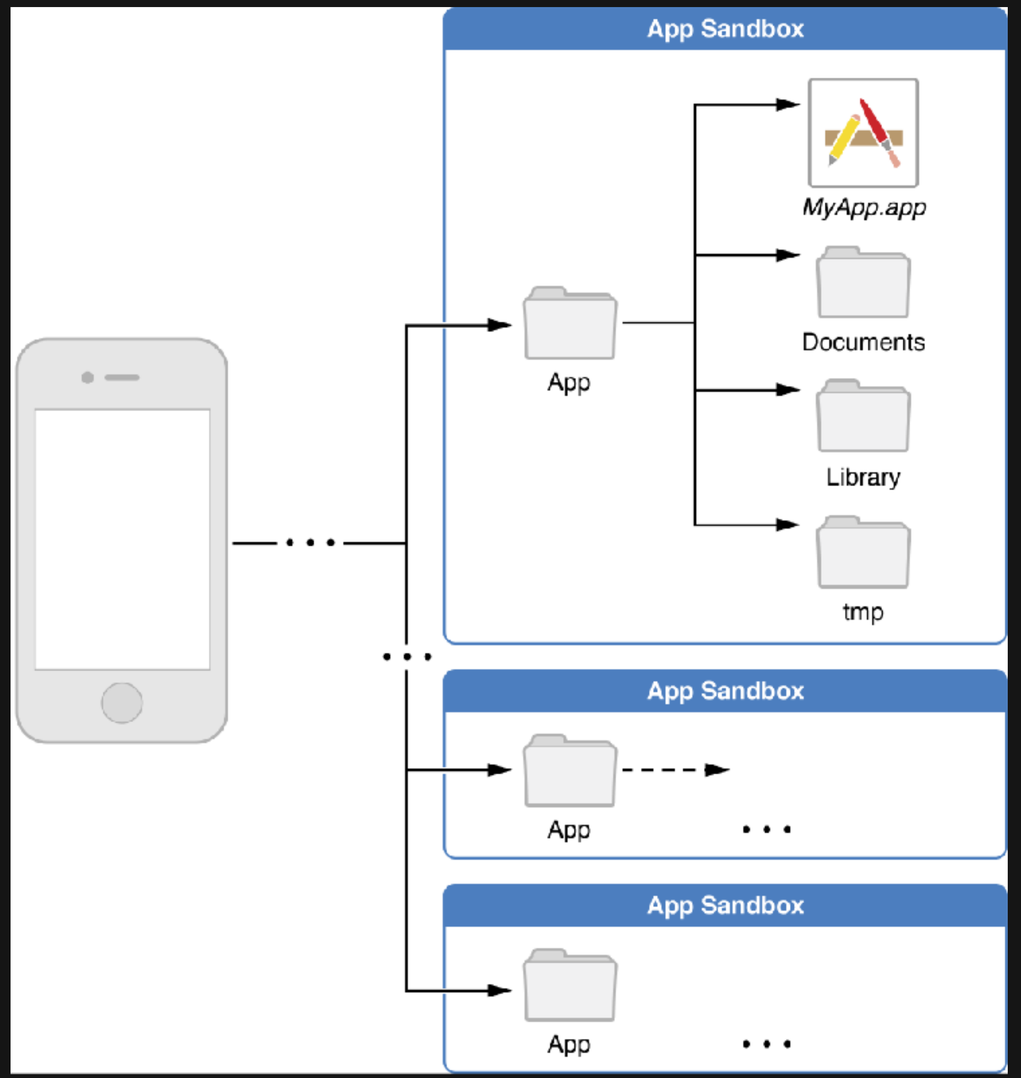
Accès au système de fichiers

Comment écrire sur le disque ?

- Système UNIX
  - Utilisation de chemins UNIX classiques
- Les apps sont en sandbox
  - On ne peut donc pas écrire n’importe où !
  - Augmente la sécurité et la confidentialité

Comment écrire sur le disque ?

- Dans la sandbox de l’app :
  - Bundle de l’app
    - Contient l’app et ses ressources
    - Bundle signé ! Si modifié, l’app ne fonctionnera plus.
  - Dossier Documents
    - Pour stocker des données importantes
  - Et d’autres...



Comment écrire sur le disque ?

1. Récupérer le chemin d'un dossier de base (Documents, Caches, etc.)
2. Ajouter le chemin souhaité pour son fichier
3. Écrire le fichier à l'emplacement souhaité

```
func saveToDisk(data: Data) {

    let fileManager = FileManager()

    guard var URL =
fileManager.urls(for: .documentDirectory,
in:.userDomainMask).first else { return }
    URL = URL.appendingPathComponent("myFile")

    data.write(to: URL, atomically: true)
}
```

Comment lire sur le disque ?

1. Récupérer le chemin d'un dossier de base (Documents, Caches, etc.)
2. Ajouter le chemin souhaité pour son fichier
3. Lire le fichier à l'emplacement souhaité

```
func readFromDisk() -> Data? {

    let fileManager = FileManager()

    guard var URL = fileManager.urls(for: .documentDirectory,
in:.userDomainMask).first else { return nil }
    URL = URL.appendingPathComponent("myFile")

    if fileManager.fileExists(atPath: URL.absoluteString) {

        let data = try? Data(contentsOf: URL)

        return data
    }

    return nil
}
```

- Pour plein d’autres possibilités sur le système de fichier, la documentation de (NS)FileManager est un bon point de départ !
  - Créer des dossiers, savoir si un fichier existe, effacer un fichier, etc.

CloudKit

iCloud

- iCloud propose plusieurs façon de stocker des données
  - iCloud Key-value storage
  - iCloud Documents
  - CloudKit

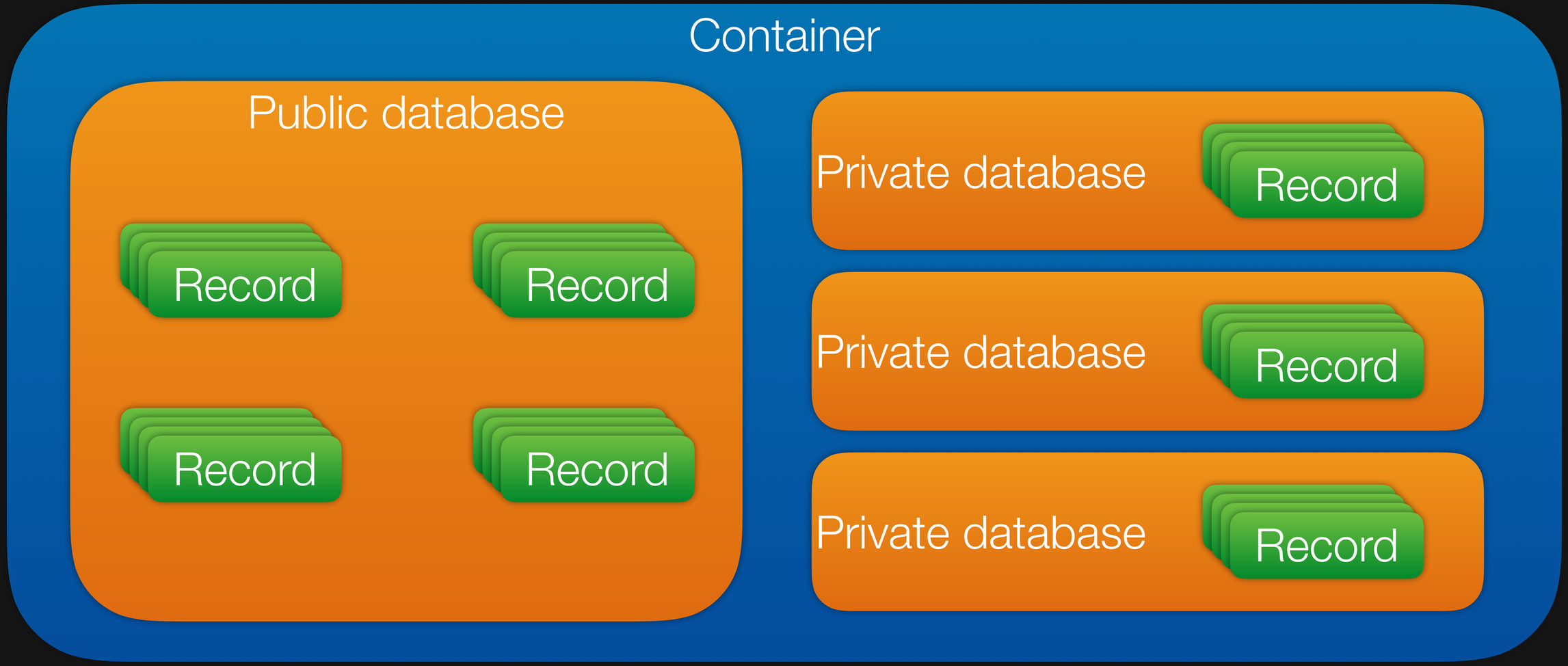
CloudKit : Bases

- Service gratuit, qui peut devenir payant en cas de (très) gros usage
  - Un calculateur est disponible sur le site d'Apple
- Propose 2 bases de données par *container*
  - Une publique
  - Une privée (associée à l'utilisateur)
- Basiquement, une app = un *container*

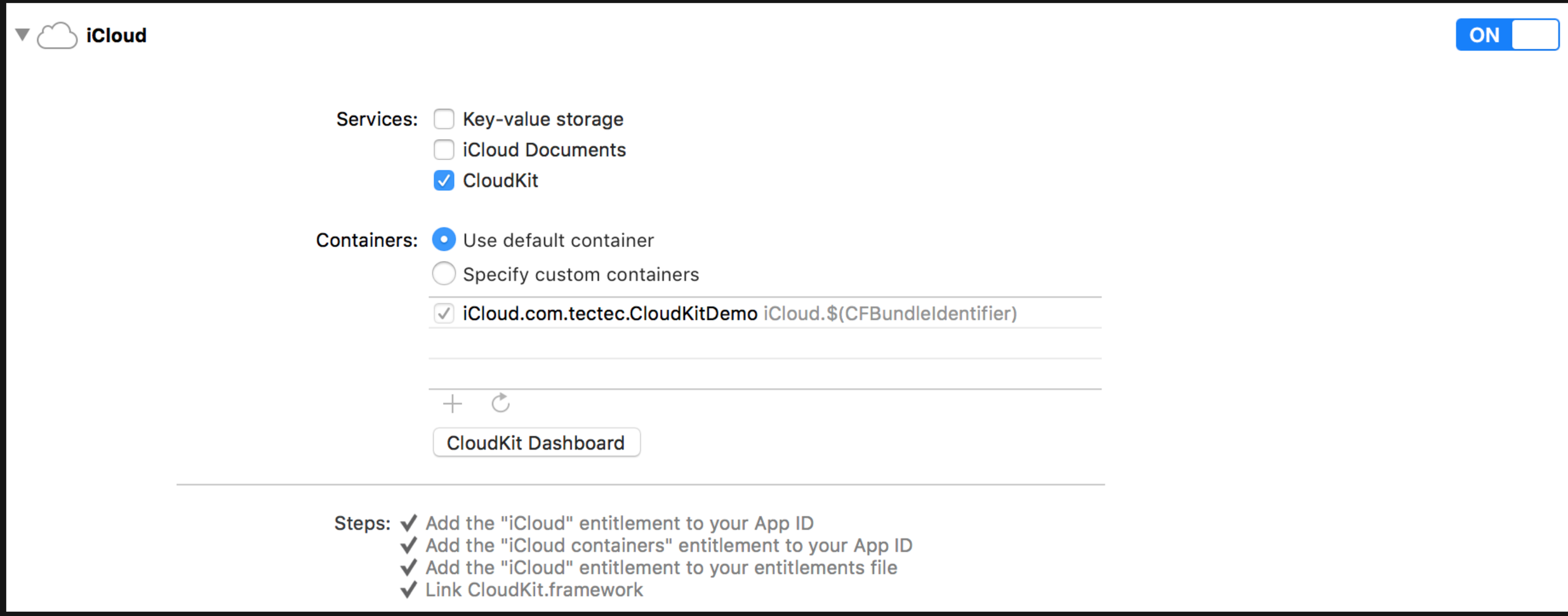
CloudKit : Bases

- Une base de donnée va contenir des enregistrements (*records*)
  - Ces enregistrement sont nos données
  - Fonctionnement clé - valeurs
  - Créées de manière opportuniste

Architecture schématique



Utiliser CloudKit



Objets utiles

- Framework : CloudKit
  - Container : CKContainer
  - Base de donnée : CKDatabase
  - Enregistrement : CKRecord
  - Requête : CKQuery
  - Abonnement : CKSubscription

Créer un enregistrement

```
let container2 = CKContainer.default()
let privateDatabase = container.privateCloudDatabase

let record = CKRecord(recordType: "Party")
record.setValue("WWDC", forKey: "partyName")
record.setValue(Date(), forKey: "partyName")

privateDatabase.save(record) { (record, error) in

   if let error = error {
       //Handle the error
   } else {
      //The object was correctly saved
   }
}
```

Récupérer des enregistrements

```
let container = CKContainer.default()
let privateDatabase = container.privateCloudDatabase

let pred = NSPredicate(format: "partyDate > %@", NSDate())
let query = CKQuery(recordType: "Party", predicate: pred)
pubDB.perform(query, inZoneWith: nil) { (records, error) in
   if let e = error {
      //Handle error
   } else {
      //Use retrieved records
  }
}
```

Abonnement

- Il est possible de créer des abonnements pour être informés automatiquement de certains changements
  - Repose sur le système des notification Push
    - L'app doit être enregistrée avec le système de notifications
  - Permet d'éviter les requêtes régulières
    - Elle nuisent à l'usage data et à la batterie

S'abonner à des changements

```
let pred = NSPredicate(format: "partyDate > %@", NSDate())
let sub = CKSubscription(recordType: "Party", predicate: pred, options: .firesOnRecordCreation)

let notInfo = CKNotificationInfo()
notInfo.alertBody = « New party!"

sub.notificationInfo = notInfo

let db = CKContainer.default().publicCloudDatabase
db.save(sub, completionHandler: { (sub, error) in
   if let e = error {
      //Handle error
   } else {
      //Subscription saved
   }
})
```

Réagir à des changements

```
func application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptions:
[UIApplicationLaunchOptionsKey: Any]?) -> Bool {
   // Override point for customization after application launch.
   application.registerForRemoteNotifications()
   let settings = UIUserNotificationSettings(types: .alert, categories: nil)
   application.registerUserNotificationSettings(settings)

   return true
}


func application(_ application: UIApplication, didReceiveRemoteNotification userInfo: [AnyHashable : Any]) {

   let cloudNotif = CKNotification(fromRemoteNotificationDictionary: userInfo)

   if (cloudNotif.notificationType == .query) {
      let queryNotif = cloudNotif as? CKQueryNotification
      if let recordID = queryNotif?.recordID {
         NotificationCenter.default.post(name: Notification.Name("NewPartyAdded"), object: nil, userInfo:
["recordID":recordID])
      }
   }
}
```

Core Data

- Framework généraliste pour la persistance des données
- Backend SQLite ou XML
- Mono utilisateur
- Fonctionnalités avancées (prédicat, schéma de migration…)

Pour aller plus loin…

- NSUserDefaults Documentation
- Archives and Serializations Programming Guide