Article
Guillaume Rongier · Oct 24, 2022 10m de lecture

Open Exchange

iOS, FHIR et IRIS for Health

Swift-FHIR-Iris

Application iOS pour exporter les données HealthKit vers InterSystems IRIS for Health (ou n'importe quel référentiel FHIR)

Table des matières

Objectif de cette démo

L'objectif est de créer une démonstration de bout en bout du protocole FHIR.

Ce que j'entends par de bout en bout, à partir d'une source d'information telle qu'un iPhone.
Collectez vos données de santé au format Apple (HealthKit), transformez-les en FHIR, puis envoyez-les au référentiel IRIS for Health d'InterSystems.

Ces informations doivent être accessibles via une interface web.

TL;DR: iPhone -> InterSystems FHIR -> Page Web.

Comment lancer cette démo

Préalables

- Pour la partie client (iOS)
    - Xcode 12
- Pour le serveur et l'application Web
    - Docker

Installation de Xcode

Pas grand chose à dire ici, ouvrez l'AppStore, cherchez Xcode, installez.

Ouvrir le projet SwiftUi

Swift est la language de programmation d'Apple pour iOS, Mac, Apple TV et Apple Watch. Elle est le remplaçant d'objective-C.

Double-cliquez sur Swift-FHIR-Iris.xcodeproj.

Ouvrez le simulateur par un clic sur la flèche en haut à gauche.

Configuration du simulateur

Accéder à Santé

Cliquez sur Étapes

Ajouter des données

Lancement du serveur FHIR d'InterSystems

Dans le dossier racine de ce git, exécutez la commande suivante :

```
docker-compose up -d
```

À la fin du processus de construction, vous serez en mesure de vous connecter au référentiel FHIR :

http://localhost:32783/fhir/portal/patientlist.html

Ce portail a été réalisé par @diashenrique.

Avec quelques modifications pour gérer les pas d'activité d'Apple.

Jeu avec l'application iOS

L'application vous demandera d'abord d'accepter de partager certaines informations.

Cliquez sur autoriser

Vous pouvez ensuite tester le serveur FHIR en cliquant sur "Sauvegarder et tester le serveur".

Les paramètres par défaut pointent vers la configuration docker.

En cas de succès, vous pouvez entrer les informations de votre patient.

Prénom, Nom de famille, Date de naissance, Genre.

Enregistrez le patient dans Fhir. Une fenêtre pop-up vous montrera votre ID Fhir unique.

Consultez ce patient sur le portail :

Accédez à: http://localhost:32783/fhir/portal/patientlist.html

Nous pouvons voir ici, qu'il y a un nouveau patient "toto" avec 0 activités.

Envoyez ses activités :

Retournez dans l'application iOS et cliquez sur Compteur de pas.

Ce panneau résume le nombre de pas de la semaine. Dans notre cas, il s'agit de 2 entrées.

Maintenant vous pouvez les envoyer à InterSystems IRIS FHIR par un clic sur envoi.

Consultez les nouvelles activités sur le portail :

Nous pouvons voir maintenant que Toto a deux nouvelles observations et activités.

Vous pouvez éventuellement cliquer sur le bouton de graphique pour l'afficher comme un graphique.

Comment ça marche

iOS

La plupart de cette démo est construite sur SwiftUI.

https://developer.apple.com/xcode/swiftui/

Qui est le dernier framework pour iOS et co.

Comment vérifier l'autorisation pour les travaux sur les données de santé

Il se trouve dans la classe SwiftFhirIrisManager.

Cette classe est un élément singleton et elle sera transportée tout autour de l'application avec l'annotation @EnvironmentObject.

Plus d'informations ici : https://www.hackingwithswift.com/quick-start/swiftui/how-to-use-environm...

La méthode requestAuthorization :

```
    // Demander l'autorisation pour accéder à HealthKit.
    func requestAuthorization() {
        // Demande d'accès.
        /// - Tag: RequestAuthorization

        let writeDataTypes: Set<HKSampleType> = dataTypesToWrite()
        let readDataTypes: Set<HKObjectType> = dataTypesToRead()

        // demande d'accès
        healthStore.requestAuthorization(toShare: writeDataTypes, read: readDataTypes
) { (succès, erreur) dans
            if !success {
                // Traitez l'erreur ici.
            } else {

                DispatchQueue.main.async {
                    self.authorizedHK = true
                }

            }
        }
    }
```

Où healthStore est l'objet de HKHealthStore().

Le HKHealthStore est comme la base de données de santé dans iOS.

dataTypesToWrite et dataTypesToRead sont les objets que nous souhaitons interroger dans la base de données.

L'autorisation doit avoir un but et cela est fait dans le fichier xml Info.plist en ajoutant :

```
    <key>NSHealthClinicalHealthRecordsShareUsageDescription</key>
    <string>Lisez les données pour IrisExporter</string>
    <key>NSHealthShareUsageDescription</key>
    <string>Envoyez les données à IRIS</string>
    <key>NSHealthUpdateUsageDescription</key>
    <string>Date d'inscription pour IrisExporter</string>
```

Comment se connecter à un référentiel FHIR ?

Pour cette partie, j'ai utilisé le paquet FHIR de Smart-On-FHIR : https://github.com/smart-on-fhir/Swift-FHIR.

La classe utilisée est le FHIROpenServer.

```
    private func test() {

        progress = true

        let url = URL(string: self.url)

        swiftIrisManager.fhirServer = FHIROpenServer(baseURL : url! , auth: nil)

        swiftIrisManager.fhirServer.getCapabilityStatement() { FHIRError in

            progress = false
            showingPopup = true

            if FHIRError == nil {
                showingSuccess = true
                textSuccess = "Connecté au référentiel fhir"
            } else {
                textError = FHIRError?.description ?? "Erreur inconnue"
                showingSuccess = false
            }

            return
        }

    }
```

Cela permet de créer un nouvel objet fhirServer dans le singleton swiftIrisManager.

Ensuite, nous utilisons la fonction getCapabilityStatement().

Si nous pouvons récupérer le capabilityStatement du serveur FHIR, cela signifie que nous nous sommes connectés avec succès au référentiel FHIR.

Ce référentiel n'est pas en HTTPS, par défaut apple interdit ce type de communication.

Pour permettre le support HTTP, le fichier xml Info.plist est édité comme suit :

```
    <key>NSAppTransportSecurity</key>
    <dict>
        <key>NSExceptionDomains</key>
        <dict>
            <key>localhost</key>
            <dict>
                <key>NSIncludesSubdomains</key>
                <true/>
                <key>NSExceptionAllowsInsecureHTTPLoads</key>
                <true/>
            </dict>
        </dict>
    </dict>
```

Comment enregistrer un patient dans le référentiel FHIR ?

Opération de base en vérifiant d'abord si le patient existe déjà dans le référentiel.

```
Patient.search(["family": "\(self.lastName)"]).perform(fhirServer)
```

Cela permet de rechercher les patients ayant le même nom de famille.

Ici, nous pouvons imaginer d'autres scénarios comme avec Oauth2 et un jeton JWT pour joindre le patient et son jeton. Mais pour cette démo, nous gardons les choses simples.

Ensuite, si le patient existe, nous le récupérons, sinon nous le créons :

```
    func createPatient(callback: @escaping (Patient?, Error?) -> Void) {
        // Créer une nouvelle ressource pour le patient
        let patient = Patient.createPatient(prénom: firstName, nom: lastName, date de
 naissance: birthDay, sex: gender)

        patient?.create(fhirServer, callback: { (erreur) dans
            callback(patient, erreur)
        })
    }
```

Comment extraire les données de HealthKit ?

Cela se fait en interrogeant le magasin Healthkit (HKHealthStore()).

Ici, nous faisons une requête pour les pas.

Préparez la requête avec le prédicat.

```
        //La semaine dernière
        let startDate = swiftFhirIrisManager.startDate
        //Now
        let endDate = swiftFhirIrisManager.endDate

        print("Collecte des séances d'entraînement entre \(startDate) et \(endDate)")

        let predicate = HKQuery.predicateForSamples(withStart: startDate, end: endDat
e, options: HKQueryOptions.strictEndDate)
```

Puis la requête elle-même avec son type de données (HKQuantityType.quantityType(forIdentifier : .stepCount)) et le prédicat.

```
func queryStepCount(){

        //La semaine dernière
        let startDate = swiftFhirIrisManager.startDate
        //Maintenant
        let endDate = swiftFhirIrisManager.endDate

        print("Collecte des séances d'entraînement entre \(startDate) et \(endDate)")
```

```
        let predicate = HKQuery.predicateForSamples(withStart: startDate, end: endDat
e, options: HKQueryOptions.strictEndDate)

        let query = HKSampleQuery(sampleType: HKQuantityType.quantityType(forIdentifi
er: .stepCount)!, predicate: predicate, limit: HKObjectQueryNoLimit, sortDescriptors:
 nil) { (requête, résultats, erreur) dans

            guard let results = results as? [HKQuantitySample] else {
                  return
            }

            process(results, type: .stepCount)

        }

        healthStore.execute(query)

    }
```

Comment transformer les données de HealthKit en FHIR ?

Pour cette partie, nous utilisons le paquet Microsoft HealthKitToFHIR

https://github.com/microsoft/healthkit-to-fhir

Il s'agit d'un paquet utile qui offre des facteurs pour transformer HKQuantitySample en FHIR Observation

```
     let observation = try! ObservationFactory().observation(from: item)
      let patientReference = try! Reference(json: ["référence" : "Patient/\(patientId
)"])
      observation.category = try! [CodeableConcept(json: [
          "coding": [
            [
              "system": "http://terminology.hl7.org/CodeSystem/observation-category",
              "code": "activité",
              "display": "Activité"
            ]
          ]
      ])]
      observation.subject = patientReference
      observation.status = .final
      print(observation)
      observation.create(self.fhirServer,callback: { (erreur) dans
          if error != nil {
              completion(error)
          }
      })
```

Où élément est un HKQuantitySample, dans notre cas un type stepCount.

Le facteur fait le gros du travail en convertissant 'élément' et 'type' en FHIR codeableConcept et 'valeur' en FHIR valueQuantity.

La référence au patientId est faite manuellement en intégrant une référence json fhir.

```
let patientReference = try! Reference(json: ["référence" : "Patient/\(patientId)"])
```

On fait de même pour la catégorie :

```
      observation.category = try! [CodeableConcept(json: [
          "codage": [
            [
              "systèmem": "http://terminology.hl7.org/CodeSystem/observation-
category",
              "code": "activité",
              "affichage": "Activité"
            ]
          ]
      ])]
```

Enfin, l'observation est créée dans le référentiel fhir :

```
      observation.create(self.fhirServer,callback: { (erreur) in
          if error != nil {
              completion(error)
          }
      })
```

Logiciel de gestion (FHIR)

Pas grand chose à dire, il est basé sur le modèle fhir de la communauté InterSystems :

https://openexchange.intersystems.com/package/iris-fhir-template

Interface utilisateur

Il est basé sur les travaux de Henrique et est un joli interface utilisateur pour les dépôts FHIR fait en jquery.

https://openexchange.intersystems.com/package/iris-fhir-portal

#Bonnes pratiques #FHIR #IoT #InterSystems IRIS for Health
Voir l'application sur InterSystems Open Exchange