Artigo
Guillaume Rongier · jan 11, 2021 9min de leitura

Open Exchange

iOS, FHIR e IRIS for Health

Swift-FHIR-Iris

Aplicativo iOS para exportar dados HealthKit para o InterSystems IRIS for Health (ou qualquer repositório FHIR)

Índice

- Objetivo desta demonstração
- Como executar esta demonstração
  - Pré-requisitos
  - Instale o Xcode
  - Abra o projeto SwiftUi
  - Configure o simulador
  - Inicie o servidor InterSystems FHIR
  - Brinque com o aplicativo iOS
- Como funciona
  - iOS
  - Como verificar a autorização para dados de saúde
  - Como se conectar a um repositório FHIR
  - Como salvar um paciente no repositório FHIR
  - Como extrair dados do HealthKit
  - Como transformar dados HealthKit em FHIR
  - Backend (FHIR)
  - Frontend

- Tarefas

Objetivo desta demonstração

O objetivo é criar uma demonstração de ponta a ponta do protocolo FHIR.

O que quero dizer com de ponta a ponta é, de uma fonte de informação como um iPhone. Colete seus dados de saúde no formato Apple (HealthKit), transforme-os em FHIR e envie para o repositório InterSystems IRIS for Health.

Essas informações devem ser acessíveis por meio de uma interface web.

iPhone -> InterSystems FHIR -> Página Web.

Como executar esta demonstração

Pré-requisitos

- Para a parte do cliente (iOS)
  - Xcode 12
- Para o servidor e aplicativo web
  - Docker

Instale o Xcode

Não há muito o que falar aqui, abra a AppStore, procure por Xcode, instale.

Abra o projeto SwiftUi

Swift é a linguagem de programação da Apple para iOS, Mac, Apple TV e Apple Watch. É o substituto do objective-C.

Clique duas vezes em Swift-FHIR-Iris.xcodeproj

Abra o simulador clicando na seta superior esquerda.

Configure o simulador

Vá em Health

Clique em Steps

Add Data

Inicie o servidor InterSystems FHIR

Na pasta raiz deste git, execute o seguinte comando:

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

No final do processo de construção, você será capaz de se conectar ao repositório FHIR:

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

Este portal foi feito por @diashenrique.

Com algumas modificações para lidar com os passos de atividade da Apple.

Brinque com o aplicativo iOS

O aplicativo primeiro solicitará que você aceite o compartilhamento de algumas informações.

Clique em authorize

Então você pode testar o servidor FHIR clicando em 'Save and test server'

As configurações padrão apontam para a configuração do docker.

Se tiver sucesso, você pode inserir as informações do seu paciente.

Nome, Sobrenome, Aniversário, Gênero.

Salve o paciente para FHIR. Um pop-up mostrará seu ID FHIR único.

Consulte este paciente no portal:

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

Podemos ver aqui, que há um novo paciente "toto" com 0 atividades.

Envie suas atividades:

Volte para o aplicativo iOS e clique em Step count

Este painel resume a contagem de passos da semana. No nosso caso, 2 entradas.

Agora você pode enviá-los para o InterSystems IRIS FHIR clicando em enviar.

Consulte as novas atividades no portal:

Podemos ver agora que o Toto tem duas novas observações e atividades.

Você pode eventualmente clicar no botão do gráfico para exibi-lo como um gráfico.

Como funciona

iOS

A maior parte desta demonstração é construída em SwiftUI.

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

Que é o framework mais recente para iOS e co.

Como verificar a autorização para dados de saúde

Ele está na classe SwiftFhirIrisManager.

Esta classe é um singleton e estará carregando todo o aplicativo com a anotação @EnvironmentObject.

Mais informações em: https://www.hackingwithswift.com/quick-start/swiftui/how-to-use-environm...

O método requestAuthorization:

```
    // Solicita autorização para acessar o HealthKit.
```

```
    func requestAuthorization() {
        // Solicitando autorização.
        /// - Tag: RequestAuthorization

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

        // pedido de autorização
        healthStore.requestAuthorization(toShare: writeDataTypes, read: readDataTypes
) { (success, error) in
            if !success {
                // Trata o erro aqui.
            } else {

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

            }
        }
    }
```

Onde healthStore é o objeto de HKHealthStore().

O HKHealthStore é como um banco de dados de dados de saúde no iOS.

dataTypesToWrite e dataTypesToRead são os objetos que gostaríamos de consultar no banco de dados.

A autorização precisa de um propósito e isso é feito no arquivo xml Info.plist adicionando:

```
    <key>NSHealthClinicalHealthRecordsShareUsageDescription</key>
    <string>Read data for IrisExporter</string>
    <key>NSHealthShareUsageDescription</key>
    <string>Send data to IRIS</string>
    <key>NSHealthUpdateUsageDescription</key>
    <string>Write date for IrisExporter</string>
```

Como conectar a um repositório FHIR

Para esta parte usei o pacote FHIR do Smart-On-FHIR: https://github.com/smart-on-fhir/Swift-FHIR

A classe usada é a 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 = "Connected to the fhir repository"
            } else {
                textError = FHIRError?.description ?? "Unknow error"
                showingSuccess = false
            }

            return
        }

    }
```

Isso cria um novo objeto fhirServer no singleton swiftIrisManager.

Em seguida, usamos getCapabilityStatement()

Se pudermos recuperar a declaração de capacidade do servidor FHIR, isso significa que nos conectamos com sucesso ao repositório FHIR.

Este repositório não está em HTTPS, por padrão, a apple bloqueia este tipo de comunicação.

Para permitir o suporte HTTP, o arquivo xml Info.plist pode ser editado assim:

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

Como salvar um paciente no repositório FHIR

Operação básica verificando primeiro se o paciente já existe no repositório

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

Isso pesquisa por paciente com o mesmo sobrenome.

Aqui, podemos imaginar outros cenários, como token Oauth2 e JWT para unir o patientid e seu token. Mas para

esta demonstração, mantemos as coisas simples.

Em seguida, se o paciente existe, nós o recuperamos, caso contrário, criamos o paciente:

```
    func createPatient(callback: @escaping (Patient?, Error?) -> Void) {
        // Cria um novo paciente
        let patient = Patient.createPatient(given: firstName, family: lastName, dateO
fBirth: birthDay, gender: gender)

        patient?.create(fhirServer, callback: { (error) in
            callback(patient, error)
        })
    }
```

Como extrair dados do HealthKit

Isso é feito consultando o healthkit Store (HKHealthStore())

Aqui nós estamos consultando os passos.

Prepare a consulta com o predicado.

```
        //Semana passada
        let startDate = swiftFhirIrisManager.startDate
        //Agora
        let endDate = swiftFhirIrisManager.endDate

        print("Collecting workouts between \(startDate) and \(endDate)")

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

Em seguida, a própria consulta com seu tipo de dados (HKQuantityType.quantityType(forIdentifier: .stepCount)) e o predicado.

```
func queryStepCount(){

        //Semana passada
        let startDate = swiftFhirIrisManager.startDate
        //Agora
        let endDate = swiftFhirIrisManager.endDate

        print("Collecting workouts between \(startDate) and \(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) { (query, results, error) in
```

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

            process(results, type: .stepCount)

        }

        healthStore.execute(query)

    }
```

Como transformar dados HealthKit em FHIR

Para esta parte, usamos o pacote Microsoft HealthKitToFHIR

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

Este é um pacote útil que oferece factories para transformar HKQuantitySample em FHIR Observation

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

Onde item é um HKQuantitySample, em nosso caso, um tipo stepCount.

O factory faz a maioria do trabalho de converter 'unit' e 'type' para FHIR codeableConcept e 'value' para FHIR valueQuantity.

A referência ao PatientId é feita manualmente, lançando uma referência json fhir.

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

O mesmo é feito para a categoria:

```
observation.category = try! [CodeableConcept(json: [
    "coding": [
      [
        "system": "http://terminology.hl7.org/CodeSystem/observation-category",
        "code": "activity",
        "display": "Activity"
      ]
    ]
])]
```

Por fim, a observação é criada no repositório fhir:

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

Backend (FHIR)

Não há muito a dizer, é baseado no modelo fhir da comunidade InterSystems:

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

Frontend

É baseado no trabalho de Henrique, que é um bom front end para repositórios FHIR feitos em jquery.

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

#FHIR #IoT #InterSystems IRIS for Health
Confira o aplicativo relacionado no InterSystems Open Exchange