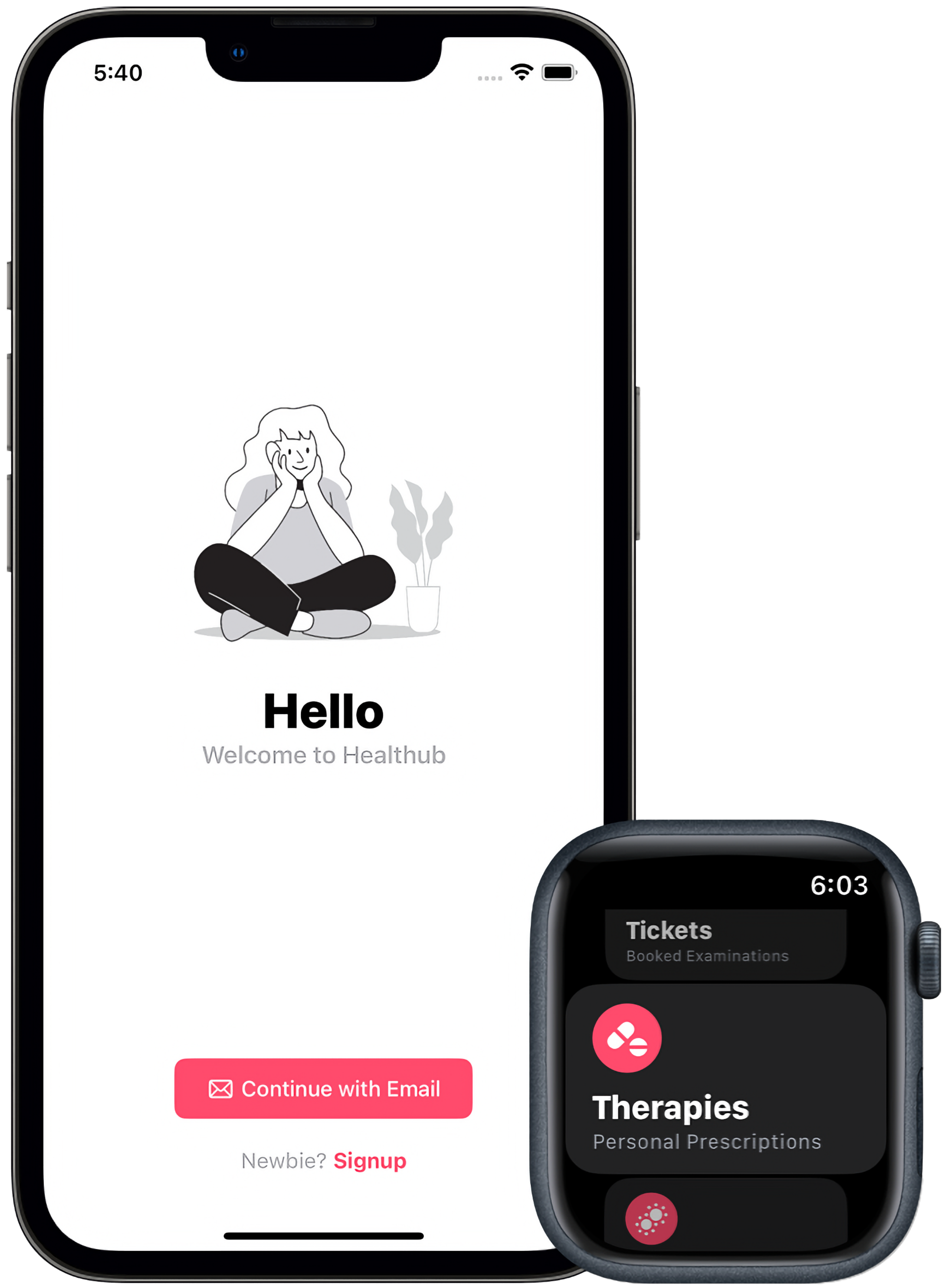
Healthub

Dario Crippa
Giovanni Dispoto

Introduction



Purpose

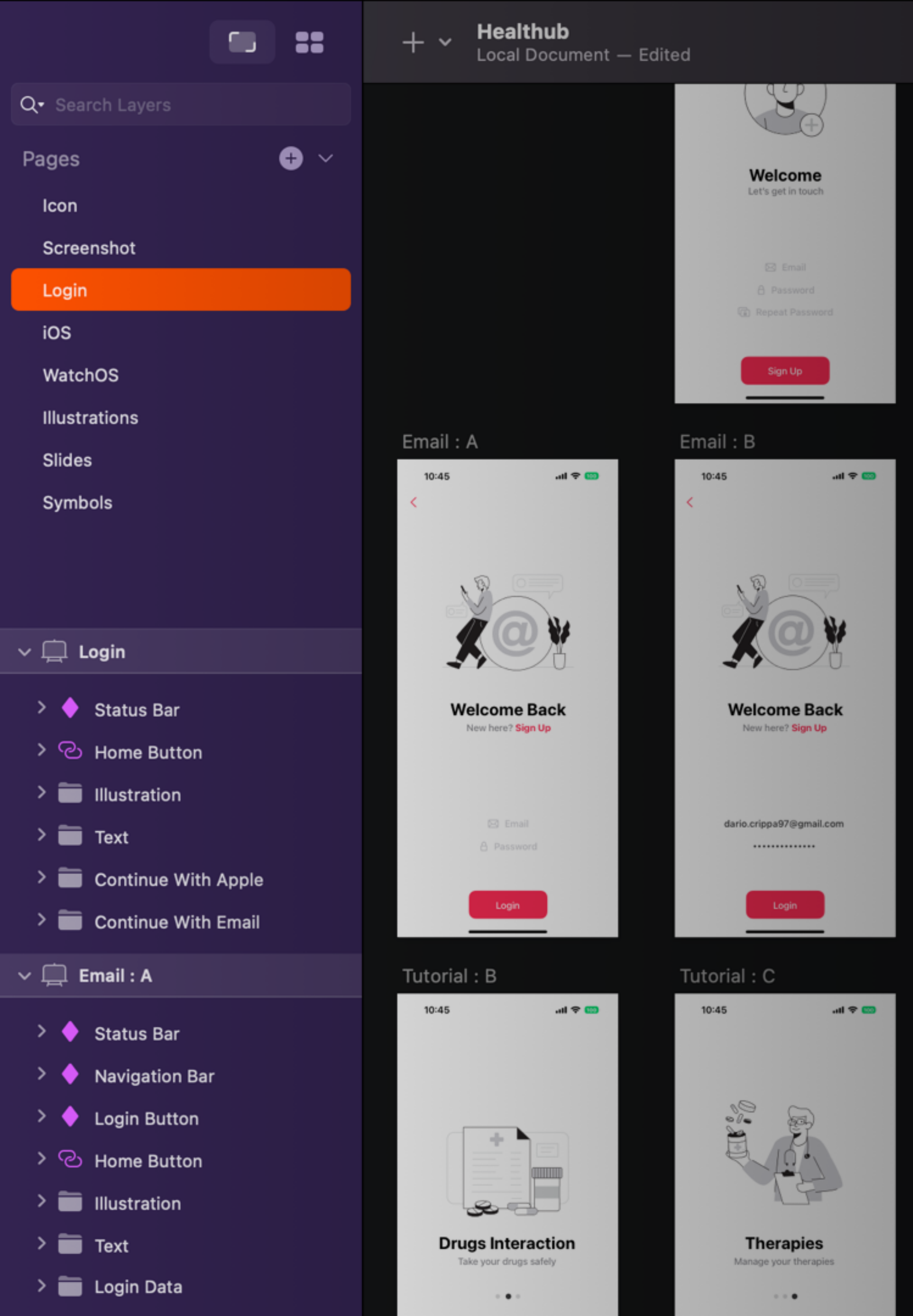
Healthub is an application that aims to digitise the health sphere of each user's life

Functionalities

What Healthub has to offer

Platforms

- iOS
- watchOS



UI

Inspiration

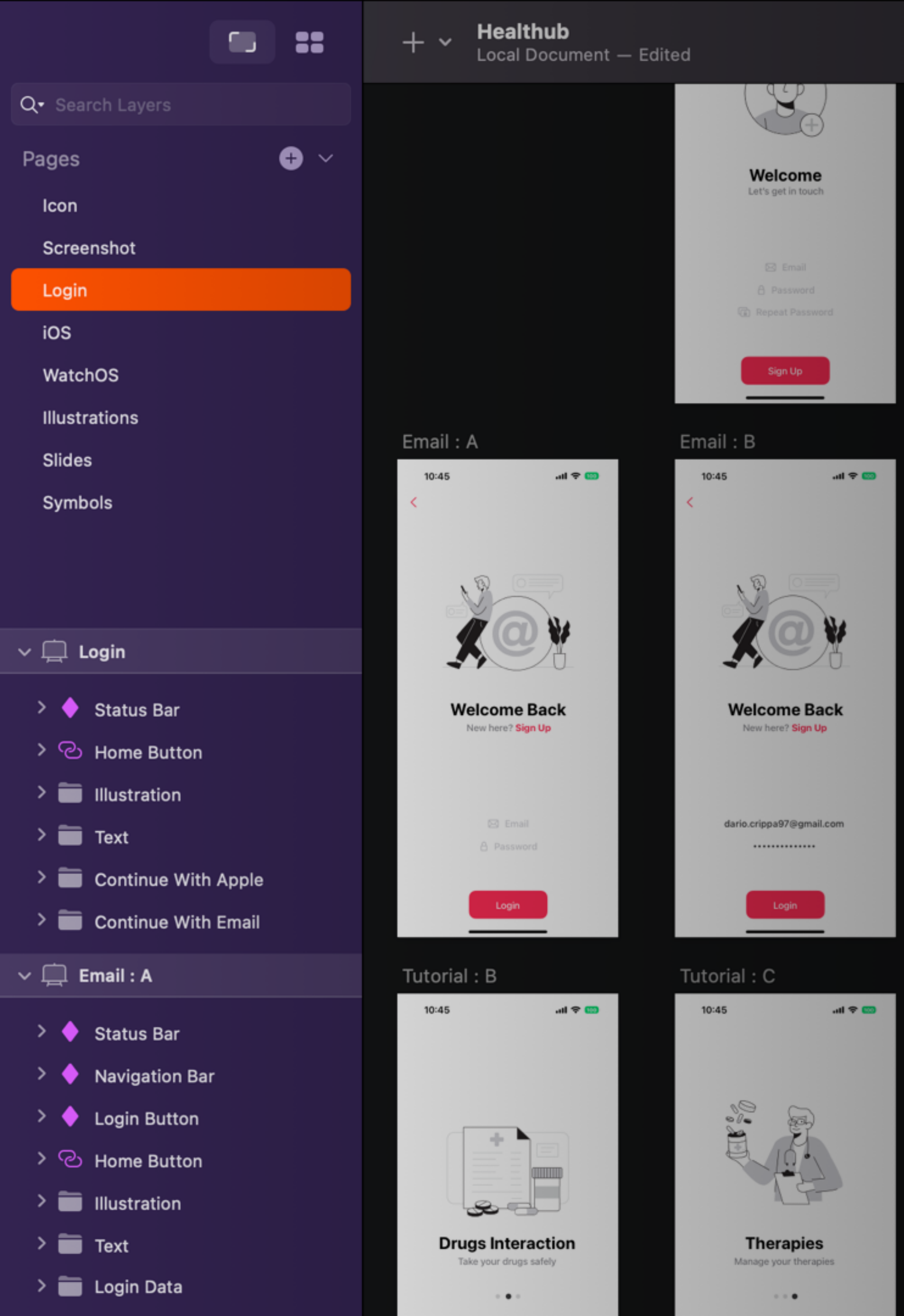
Human Interface Guidelines

Sketch

Mocking the app for preliminary considerations

Libraries

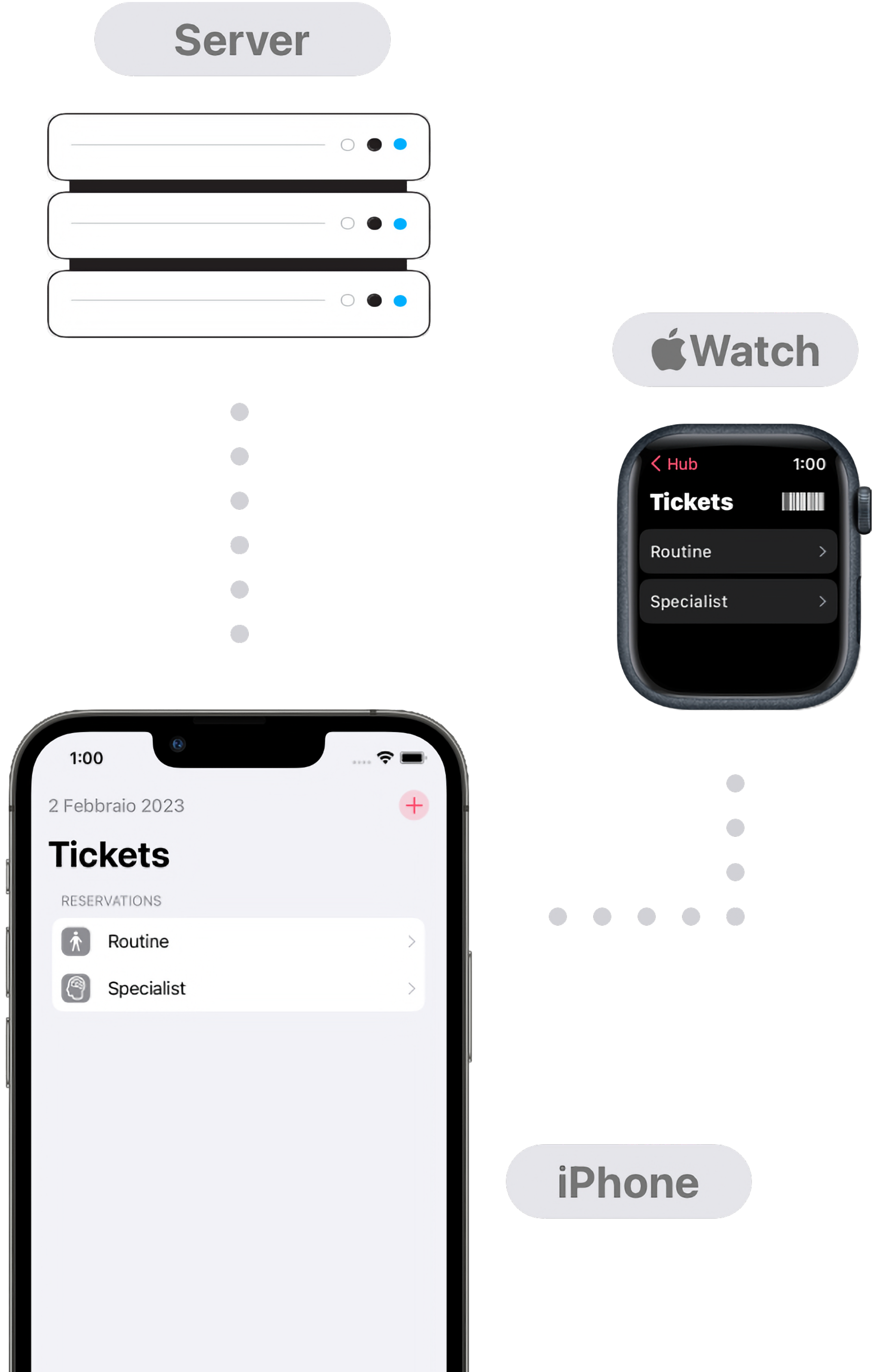
External frameworks from GitHub



SwiftUI

- Swift native
- Aa Describe the UI through code
- No more constraints
- Multiple layouts
- The dark side

Architecture



MVVM

Adopted architectural pattern

APIs

Apple frameworks such as MapKit and CoreData

Drug Interactions

Detect dangerous interactions within therapies

Communication

HTTPS

WatchConnectivity

Backend

SQLite

Software library that provides a relational database management system

FastAPI

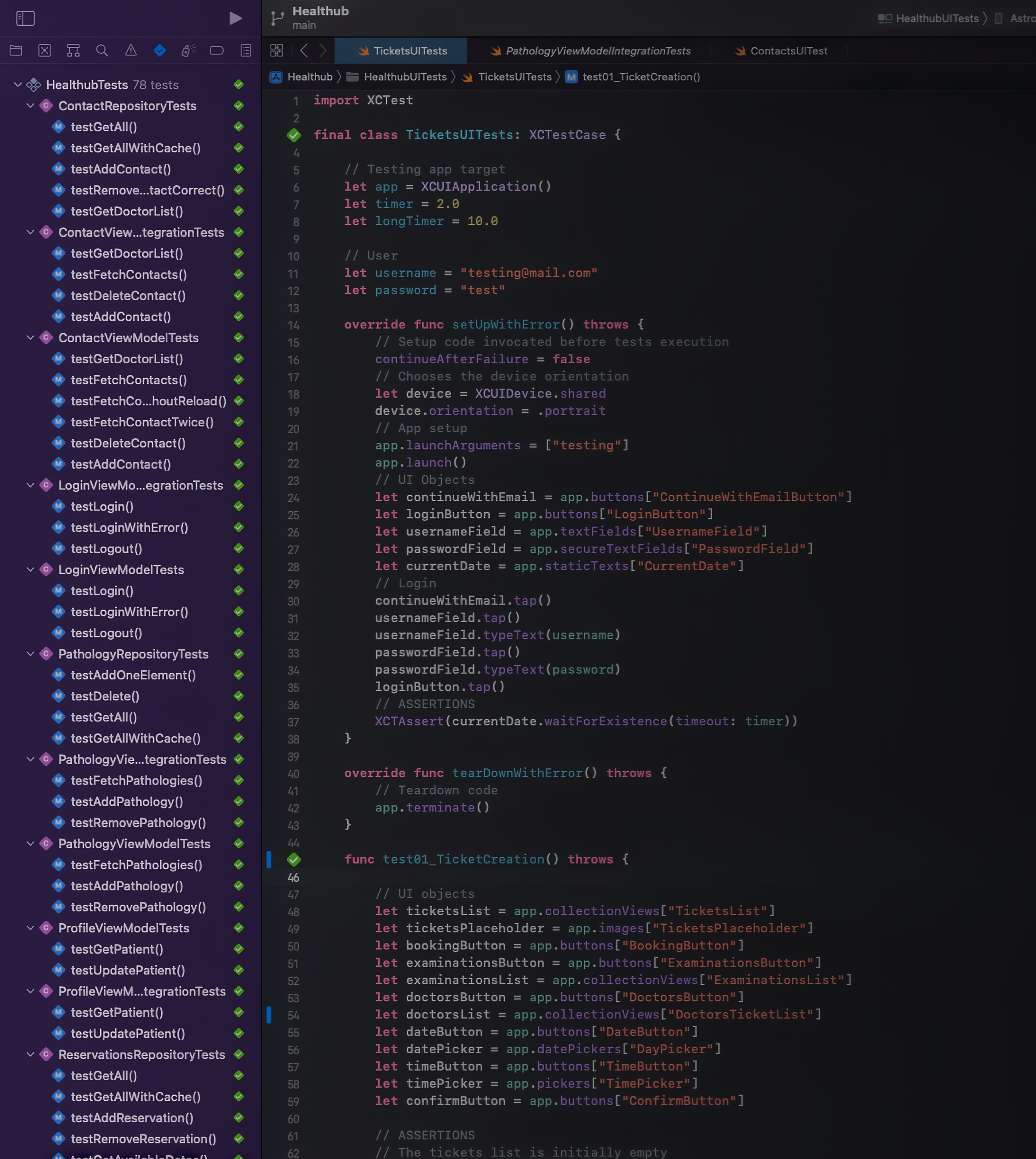
Modern high-performance web framework for building APIs with Python based on standard type hints

Server

Docker container have been shipped on DigitalOcean in order to make the APIs accessible everywhere and at anytime

Libraries

Other packages used for the backend development



Healthhub
main

HealthhubUITests 〉 AstroSimulator

TicketsUITests | PathologyViewModelIntegrationTests | ContactsUITest

Healthub 〉 HealthubUITests 〉 TicketsUITests 〉 test01_TicketCreation()

```
HealthubTests 78 tests
  ContactRepositoryTests
    testGetAll()
    testGetAllWithCache()
    testAddContact()
    testRemove...tactCorrect()
    testGetDoctorList()
  ContactView...tegrationTests
    testGetDoctorList()
    testFetchContacts()
    testDeleteContact()
    testAddContact()
  ContactViewModelTests
    testGetDoctorList()
    testFetchContacts()
    testFetchCo...houtReload()
    testFetchContactTwice()
    testDeleteContact()
    testAddContact()
  LoginViewMo...egrationTests
    testLogin()
    testLoginWithError()
    testLogout()
  LoginViewModelTests
    testLogin()
    testLoginWithError()
    testLogout()
  PathologyRepositoryTests
    testAddOneElement()
    testDelete()
    testGetAll()
    testGetAllWithCache()
  PathologyVie...tegrationTests
    testFetchPathologies()
    testAddPathology()
    testRemovePathology()
  PathologyViewModelTests
    testFetchPathologies()
    testAddPathology()
    testRemovePathology()
  ProfileViewModelTests
    testGetPatient()
    testUpdatePatient()
  ProfileViewM...tegrationTests
    testGetPatient()
    testUpdatePatient()
  ReservationsRepositoryTests
    testGetAll()
    testGetAllWithCache()
    testAddReservation()
    testRemoveReservation()
```

```swift
import XCTest

final class TicketsUITests: XCTestCase {

    // Testing app target
    let app = XCUIApplication()
    let timer = 2.0
    let longTimer = 10.0

    // User
    let username = "testing@mail.com"
    let password = "test"

    override func setUpWithError() throws {
        // Setup code invocated before tests execution
        continueAfterFailure = false
        // Chooses the device orientation
        let device = XCUIDevice.shared
        device.orientation = .portrait
        // App setup
        app.launchArguments = ["testing"]
        app.launch()
        // UI Objects
        let continueWithEmail = app.buttons["ContinueWithEmailButton"]
        let loginButton = app.buttons["LoginButton"]
        let usernameField = app.textFields["UsernameField"]
        let passwordField = app.secureTextFields["PasswordField"]
        let currentDate = app.staticTexts["CurrentDate"]
        // Login
        continueWithEmail.tap()
        usernameField.tap()
        usernameField.typeText(username)
        passwordField.tap()
        passwordField.typeText(password)
        loginButton.tap()
        // ASSERTIONS
        XCTAssert(currentDate.waitForExistence(timeout: timer))
    }

    override func tearDownWithError() throws {
        // Teardown code
        app.terminate()
    }

    func test01_TicketCreation() throws {

        // UI objects
        let ticketsList = app.collectionViews["TicketsList"]
        let ticketsPlaceholder = app.images["TicketsPlaceholder"]
        let bookingButton = app.buttons["BookingButton"]
        let examinationsButton = app.buttons["ExaminationsButton"]
        let examinationsList = app.collectionViews["ExaminationsList"]
        let doctorsButton = app.buttons["DoctorsButton"]
        let doctorsList = app.collectionViews["DoctorsTicketList"]
        let dateButton = app.buttons["DateButton"]
        let datePicker = app.datePickers["DayPicker"]
        let timeButton = app.buttons["TimeButton"]
        let timePicker = app.pickers["TimePicker"]
        let confirmButton = app.buttons["ConfirmButton"]

        // ASSERTIONS
        // The tickets list is initially empty
```

Testing

Unit Testing

Mocked Dependencies

Integration Testing

ViewModel and Repositories interaction

UI Testing

Blackbox inspection of the app

Coverage

89.3%