Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Звіт

З лабораторної роботи № 4

«Мобільний додаток iOS для системи ConditionsController»

З дисципліни «Архітектура, аналіз та проектування програмного забезпечення»

Підготував:

Ст. гр. ПІ-15-1

Євченко Богдан

2018

Для зв’язку з сервером викорустовується класс Server:

class Server

{

Необхідно вказати інтернет адресу для доступу до сервера:

static let url = "https://conditions-controller-olderor.c9users.io/api/"

Запити, які необхідні для отримання данних з серверу:

enum Action : String {

case getProductInfo = "product/get"

case getProductTrackings = "product/get-tracks"

}

Функція, яка отримує дані з серверу та повертає модель з отриманими даними:

static func getProductInfo(productId: String, onSuccess: @escaping (\_ product: ProductModel?) -> ()) {

let uri = url + Action.getProductInfo.rawValue

let params = ["product\_id": productId]

Alamofire.request(uri, method: .post, parameters: params, encoding: JSONEncoding.default, headers: nil)

.responseObject { (response: DataResponse<ProductModel>) in

onSuccess(response.result.value)

}

}

Приклад моделі:

class TrackingStatusModel: ErrorModel {

Необхідні для роботи мобільного додатку поля, які повертає сервер:

var id: Int!

var conditionId: Int!

var productId: Int!

var trackingDeviceId: Int!

var value: Double!

var dateRecordered: String!

var date: Date!

required init?(map: Map){

super.init(map: map)

}

Для встановлення даних використовується інструмент Mapping, який отримує інформацію з данних та назначає її в поля моделі.

override func mapping(map: Map) {

super.mapping(map: map)

id <- map["id"]

conditionId <- map["condition\_id"]

productId <- map["product\_id"]

trackingDeviceId <- map["tracking\_device\_id"]

value <- map["value"]

dateRecordered <- map["date\_recordered"]

let dateFormatter = DateFormatter()

dateFormatter.dateFormat = "yyyy-MM-dd HH:mm:ss.SSSSSS"

date = dateFormatter.date(from: dateRecordered)!

}

}

Для сканування QR-коду використовується інструмент AVCaptureDevice, який оновлює зображення з камери мобільного пристрою, та оброблює отримані з камери метадані.

let captureDevice = AVCaptureDevice.default(for: AVMediaType.video)

do {

let input = try AVCaptureDeviceInput(device: captureDevice!)

captureSession.addInput(input)

let captureMetadataOutput = AVCaptureMetadataOutput()

captureSession.addOutput(captureMetadataOutput)

captureMetadataOutput.setMetadataObjectsDelegate(self, queue: DispatchQueue.main)

captureMetadataOutput.metadataObjectTypes = supportedCodeTypes

} catch {

print(error)

return

}

Шар захоплювання камери додається до ієрархії вью.

videoPreviewLayer = AVCaptureVideoPreviewLayer(session: captureSession)

videoPreviewLayer?.videoGravity = AVLayerVideoGravity.resizeAspectFill

videoPreviewLayer?.frame = view.layer.bounds

view.layer.addSublayer(videoPreviewLayer!)

Частина коду, яка розшифровує отриману інформацію з метаданих, якщо було знайдено значення QR-коду, то користувач переходить до сторінки з інформацією про продукт.

let metadataObj = metadataObjects[0] as! AVMetadataMachineReadableCodeObject

if supportedCodeTypes.contains(metadataObj.type) {

let barCodeObject = videoPreviewLayer?.transformedMetadataObject(for: metadataObj)

qrCodeFrameView?.frame = barCodeObject!.bounds

if metadataObj.stringValue != nil {

decodedData = metadataObj.stringValue!

launchApp()

}

}

Приклад функції, яка малює діаграму стану продукту. Коли отримується з сервера нова інформація про продукт, викликається ця функція, яка оброблює передані дані. Ця функція викликається один раз в секунду.

func addStatuses(\_ productModel: ProductModel?) {

if productModel == nil {

return

}

if productModel!.error != nil {

print(productModel!.error!)

return

}

Дані необхідно отсортувати за часом, щоб діаграма була побудована коректно.

productModel!.trackingStatuses.sort(by: { $0.date < $1.date })

for i in 0..<productModel!.conditions.count {

let condition = productModel!.conditions[i]

if let chartId = chartIdForConditionDict[condition.id!] {

Ініціалізація нової діаграми.

if chartViews[chartId].series.count == 0 {

chartViews[chartId].add(ChartSeries([]))

chartViews[chartId].add(ChartSeries([]))

chartViews[chartId].add(ChartSeries([]))

chartViews[chartId].series[1].color = UIColor.red

chartViews[chartId].series[2].color = UIColor.green

chartViews[chartId].series[1].area = true

chartViews[chartId].series[2].area = true

chartViews[chartId].xLabelsFormatter = { \_,\_ in "" }

}

for status in productModel!.trackingStatuses {

if status.conditionId == condition.id {

if lastDate == nil || lastDate < status.date {

lastDate = status.date

lastDateString = status.dateRecordered

}

Встановлення даних в діаграму (значення відповідного показника в певний момент часу.

chartViews[chartId].series[0].data.append((Double(chartViews[chartId].series[0].data.count), status.value))

if condition.min != nil {

chartViews[chartId].series[1].data.append((Double(chartViews[chartId].series[1].data.count), condition.min!))

}

if condition.max != nil {

chartViews[chartId].series[2].data.append((Double(chartViews[chartId].series[2].data.count), condition.max!))

}

}

}

Викликається функція, яка вказує оновлення діаграми.

chartViews[chartId].setNeedsDisplay()

}

}

}