

ПРИЛОЖЕНИЕ И

**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук
Департамент программной инженерии

СОГЛАСОВАНО
Профессор департамента
программной инженерии
факультета компьютерных наук
доктор техн. наук

УТВЕРЖДАЮ
Академический руководитель
образовательной программы
«Программная инженерия»
профессор департамента программной
инженерии, канд. техн. наук

_____ Д.В. Александров
« ____ » _____ 2018 г.

_____ В.В. Шилов
« ____ » _____ 2018 г.

**Сервис распознавания лиц для идентификации личности
Клиент**

**Текст программы
ЛИСТ УТВЕРЖДЕНИЯ
RU.17701729.04.01-01 12 01-2-ЛУ**

Исполнитель
студент группы БПИ 143
_____ /Н.О. Константиновский /
« ____ » _____ 2018 г.

Име. № подл	Подп. и дата	Взам. инв. №	Име. № дубл.	Подп. и дата
RU.17701729.04.01-01 12 01-2				

Москва 2018

УТВЕРЖДЕН
RU.17701729.04.0-01 51 01-1-ЛУ

**Сервис распознавания лиц для идентификации личности
Клиент**

Текст программы

RU.17701729.04.01-01 12 01-2

Листов 52

<i>Инв. № подл</i>	<i>Подп. и дата</i>	<i>Взам. инв. №</i>	<i>Инв. № дубл.</i>	<i>Подп. и дата</i>
RU.17701729.04.01-01 12 01-2				

Москва 2018

Содержание

СОДЕРЖАНИЕ	139
1. ТЕКСТ ПРОГРАММЫ	140
1.1. MINDFACE.SWIFT.....	140
1.2. MFDRAW.SWIFT	150
1.3. MFCAMERAFRAMEEXTRACTOR.SWIFT.....	154
1.4. MFNNFACECODING.SWIFT	159
1.5. MFFACENORMALIZATION.SWIFT	161
1.6. MFFACEPROCESSORCONTROLLER.SWIFT	165
1.7. MFFACEDETECT.SWIFT	168
1.8. L2NORMALIZE.SWIFT.....	171
1.9. LRN.SWIFT	173
1.10. MULCONSTANT.SWIFT	176
1.11. SQRT.SWIFT.....	178
1.12. SQUARE.SWIFT	180
1.13. MFFACERECT.SWIFT	182
1.14. WEBREQUESTS.SWIFT	183
ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ.....	189

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.01-0112				
Инв. № подл.	Подп. и дата.	Взам. инв. №	Инв. № дубл.	Подп. и дата

1. ТЕКСТ ПРОГРАММЫ

1.1. MindFace.swift

```
//

// MindFace.swift

// MindFace

//

// Created by Nikita Konstantinovskiy on 16.02.2018.

// Copyright © 2018 nikkonst. All rights reserved.

//

import Foundation

import Vision

import CoreML

/// Mind face delegate

@objc public protocol MindFaceDelegate: class {

    @objc optional func videoFrames(image: UIImage)

    @objc optional func faceRectFrames(image: UIImage, str: String)
```



```
public class MindFace: CameraFrameExtractorDelegate {
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.01-0151				
Инв. № подл.	Подп. и дата.	Взам. инв. №	Инв. № дубл.	Подп. и дата

```
/// Input of photo/video for MindFace processing
```

```
///
```

```
/// - MainCamera: Main device camera
```

```
public enum MFInputType {
```

```
    case Camera
```

```
}
```

```
/// Camera type
```

```
///
```

```
/// - MainCamera: Main device camera
```

```
public enum MFCameraType {
```

```
    case MainCamera
```

```
}
```

```
/// Face process type
```

```
///
```

```
/// - Stop: No process
```

```
/// - FaceDetection: Face detection
```

```
/// - FaceDetectionDrawRect: Face detection with rect drawing
```

```
public enum MFProcessType {
```



Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.01-0151				
Инв. № подл.	Подп. и дата.	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

/// Current input type

private var currentInputType: MFInputType?

/// Current camera type

private var currentCameraType: MFCameraType?

/// Current process type

private var currentProcessType: MFProcessType?

/// Queue for async face processing

private let faceQueue = DispatchQueue(label: "Face processing queue")


/// Current faces

private var faces = [MFFaceRect]()


/// Frame extractor from device camera

private var cameraFrameExtractor: MFCameraFrameExtractor!


/// Face processor

private let faceProcessor = MFFaceProcessorController()


/// Class detegate for callbacks

public weak var delegate: MindFaceDelegate?

```

```

/// Init MindFace object with input type

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.01-0151				
Инв. № подл.	Подп. и дата.	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

///

/// - Parameter inputType: Type of input for processing

public convenience init(inputType: MFInputType) {

    self.init()

    currentInputType = inputType

    switch currentInputType {

    case .Camera?:

        initCamera(cameraType: .MainCamera)

    default:

        break

    }

}

```

```

        initCamera(cameraType: cameraType)

    }

```

```

/// Init camera extractor

```

```

private func initCamera(cameraType: MFCameraType) {

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.01-0151				
Инв. № подл.	Подп. и дата.	Взам. инв. №	Инв. № дубл.	Подп. и дата

```
currentCameraType = cameraType
```

```
switch currentCameraType {
```

```
case .MainCamera? :
```

```
    cameraFrameExtractor = MFCameraFrameExtractor(cameraPosition: .back)
```

```
    cameraFrameExtractor.delegate = self
```

```
default:
```

```
    break
```

```
}
```

```
}
```

```
do {
```

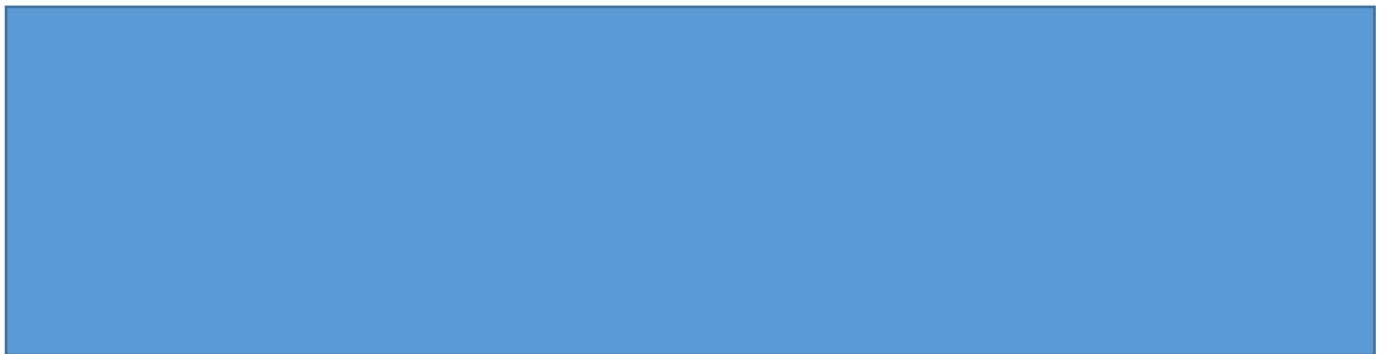
```
    let model = try VNCoreMLModel(for: MindFaceCoreML().model)
```

```
DispatchQueue(label: "NN queue").async {
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.01-0151				
Инв. № подл.	Подп. и дата.	Взам. инв. №	Инв. № дубл.	Подп. и дата


```
let handler = VNImageRequestHandler(cgImage: img, orientation: CGImagePropertyOrientation.up,
options: [:])
```

```
do {
    try handler.perform([request])
}
catch {
    fatalError("Prediction")
}
}
}
catch {
    fatalError("Creation")
}
}
```



```
self.faceProcessor.detectFace(image: img, completion: completion)
}
}
```

/// Face rectangle detection from device camera. Creates video output with face frames and receive its to delegate.

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.01-0151				
Инв. № подл.	Подп. и дата.	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

public func startFaceRectDetectorFromCamera(drawFaceRect: Bool) {

    if (cameraFrameExtractor != nil) {

        if drawFaceRect {

            currentProcessType = .FaceDetectionDrawRect


            faceQueue.async { [unowned self] in

                //          self.faceProcessor.detectFaceAndDraw(image: self.curImage, completion: self.faceDetected)

            }

        }

        else {



        })

    }

}

cameraFrameExtractor.start()

}

```

```

public func startFaceRecognitionFromCamera(drawFaceRect: Bool) {

    if (cameraFrameExtractor != nil) {

        if drawFaceRect {

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.01-0151				
Инв. № подл.	Подп. и дата.	Взам. инв. №	Инв. № дубл.	Подп. и дата

```
currentProcessType = .FaceRecognitionDrawRect
```

```

faceQueue.async { [unowned self] in

    self.faceProcessor.recogniseFaceAndDraw(image: self.curImage, completion: self.faceDetected)

}

}

else {

//      currentProcessType = .FaceDetection

//

//      faceQueue.async { [unowned self] in

//          self.faceProcessor.detectFace(image: self.curImage, completion: { (faces) in

//              self.faceDetected(faces: faces, image: nil)

//          })

//      }

}
}

```



```
private var curImage : UIImage?
```

```
/// Number of frame (0 to 100)
```

```
private var num: Int = 0
```

```
/// Receive frames from camera
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.01-0151				
Инв. № подл.	Подп. и дата.	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

///

/// - Parameter image: Captured frame

func capturedCameraFrame(frame: CGImage) {

    var uiFrame: UIImage

    uiFrame = UIImage(cgImage: frame)

    curImage = uiFrame

    num += 1

    if num == 100 {

        num = 0

    }

    self.delegate?.videoFrames!(image: uiFrame)

}

```

```

/// Get result of image detection processing

```

```

switch self.currentProcessType {

case .FaceDetection?:

    DispatchQueue.main.async() {

    }

}

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.01-0151				
Инв. № подл.	Подп. и дата.	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

faceQueue.async { [unowned self] in

    self.faceProcessor.detectFace(image: self.curImage, completion: { (faces) in

//        self.faceDetected(faces: faces, image: nil)

    })

}

case .FaceDetectionDrawRect?:

    if image != nil {

        DispatchQueue.main.async() {

//            self.delegate?.faceRectFrames!(image: image!)

        }

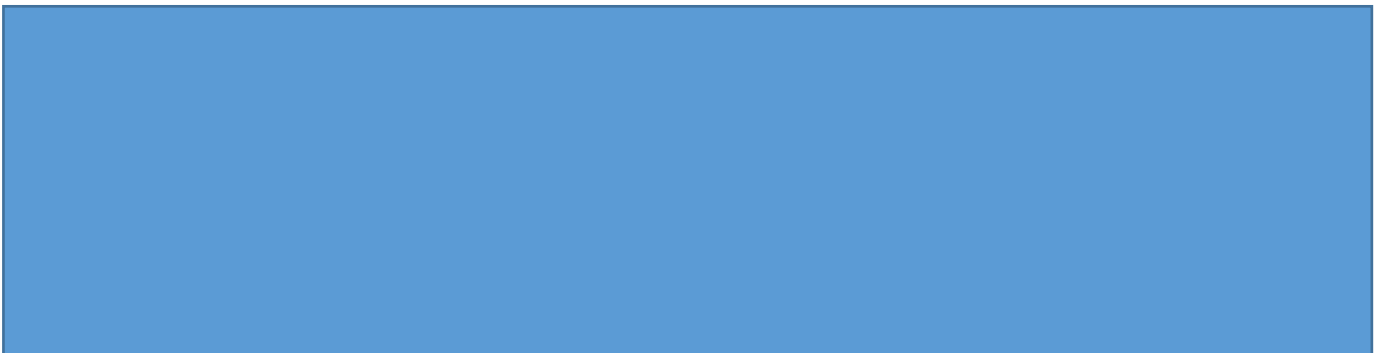
    }

    faceQueue.async { [unowned self] in

//        self.faceProcessor.detectFaceAndDraw(image: self.curImage, completion: self.faceDetected)

    }

```



```

}

```

```

faceQueue.async { [unowned self] in

    self.faceProcessor.recogniseFaceAndDraw(image: self.curImage, completion: self.faceDetected)

}

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.01-0151				
Инв. № подл.	Подп. и дата.	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

    default:
        break
    }
}
}

```

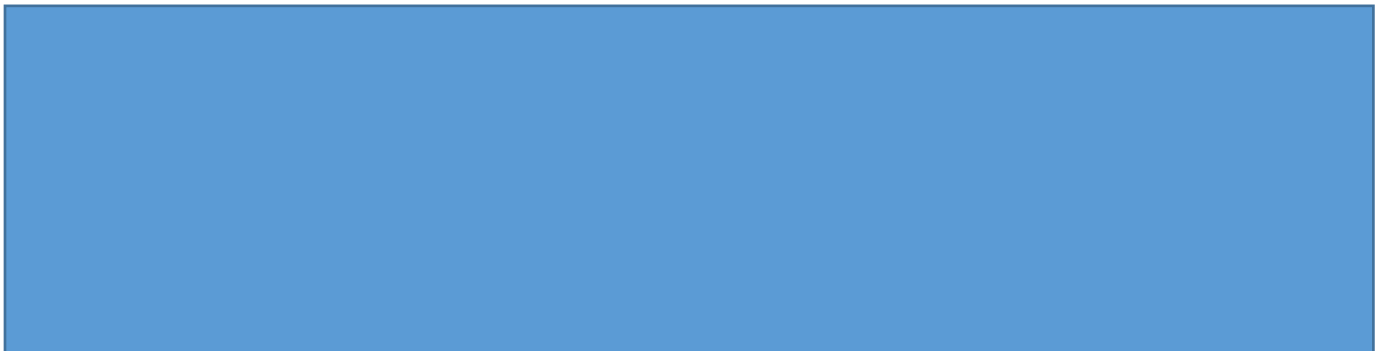
1.2. MFDDraw.swift

```

//
// MFDDraw.swift
// MindFace
//
// Created by Nikita Konstantinovskiy on 16.03.2018.
// Copyright © 2018 nikkonst. All rights reserved.
//

```

```
import Foundation
```



```

        height: image.size.height))

let transform = CGAffineTransform(scaleX: 1, y: -1).translatedBy(x: 0, y: -image.size.height)

let img = renderer.image { ctx in

```

```
    for face in faces {
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.01-0151				
Инв. № подл.	Подп. и дата.	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

let rectangle = CGRect(x: face.faceRect.origin.x * image.size.width,
                        y: face.faceRect.origin.y * image.size.height,
                        width: face.faceRect.size.width * image.size.width,
                        height: face.faceRect.size.height * image.size.height).applying(transform)

ctx.cgContext.setFillColor(UIColor.init(white: 0, alpha: 0).CGColor)
ctx.cgContext.setStrokeColor(UIColor.yellow.CGColor)
ctx.cgContext.setLineWidth(2)

ctx.cgContext.addRect(rectangle)

ctx.cgContext.drawPath(using: .fillStroke)
}
}

return img
}

```



```

let rectangle = CGRect(x: face.faceRect.origin.x * image.size.width,
                        y: face.faceRect.origin.y * image.size.height,
                        width: face.faceRect.size.width * image.size.width,

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.01-0151				
Инв. № подл.	Подп. и дата.	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

        height: face.faceRect.size.width * image.size.width).applying(transform)
//face.faceRect.size.height * image.size.height

    let croppedCGImage:CGImage = (image.cgImage?.cropping(to: rectangle))!

    let croppedImage = UIImage(cgImage: croppedCGImage)

    croppedImages.append(croppedImage)
}

return croppedImages
}

```



```

// draw the image

image.draw(in: CGRect(x: 0, y: 0, width: image.size.width, height: image.size.height))

context?.translateBy(x: 0, y: image.size.height)

context?.scaleBy(x: 1.0, y: -1.0)

for face in faces {

    let points = face.mainPoints

    for point in points {

        let rectangle = CGRect(x: point.x * image.size.width - 2,

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.01-0151				
Инв. № подл.	Подп. и дата.	Взам. инв. №	Инв. № дубл.	Подп. и дата


```

y: point.y * image.size.height - 2,
width: 4,
height: 4)

```

```

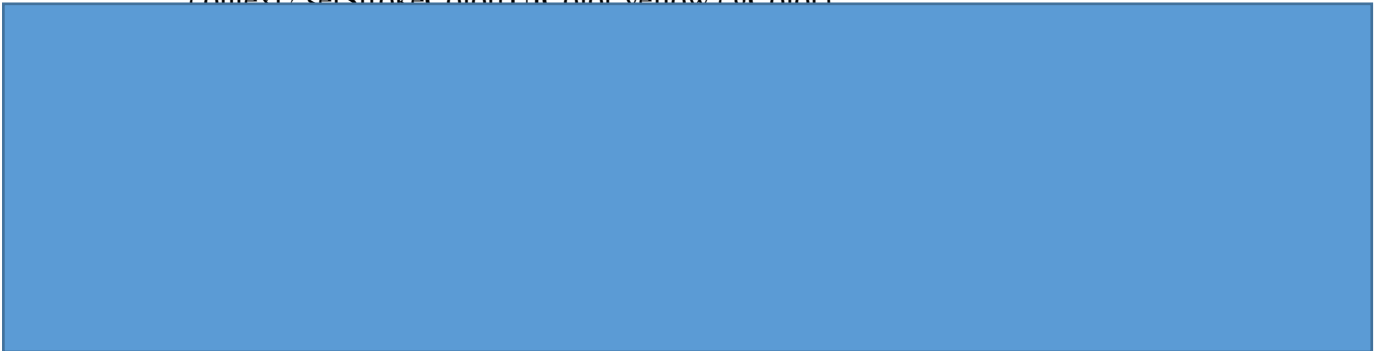
context?.setFillColor(UIColor.init(white: 0, alpha: 0).CGColor)

```

```

context?.setStrokeColor(UIColor.yellow.CGColor)

```



```

context?.drawPath(using: .fillStroke)

```

```

}

```

```

}

```

```

// get the final image

```

```

let img = UIGraphicsGetImageFromCurrentImageContext()

```

```

// end drawing context

```

```

UIGraphicsEndImageContext()

```

```

return img!

```

```

}

```

```

}

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.01-0151				
Инв. № подл.	Подп. и дата.	Взам. инв. №	Инв. № дубл.	Подп. и дата

1.3. MFCameraFrameExtractor.swift

```
//

// MFCameraFrameExtractor.swift

// MindFace

//

// Created by Nikita Konstantinovskiy on 14.03.2018.

// Copyright © 2018 nikkonst. All rights reserved.

//

import Foundation

import AVFoundation
```



```
/// Class for extract frames from camera

internal class MFCameraFrameExtractor: NSObject,
AVCaptureVideoDataOutputSampleBufferDelegate {

    weak var delegate: CameraFrameExtractorDelegate?

    /// Capture session

    private let captureSession = AVCaptureSession();
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.01-0151				
Инв. № подл.	Подп. и дата.	Взам. инв. №	Инв. № дубл.	Подп. и дата

```
/// Queue for async frame extraction
```

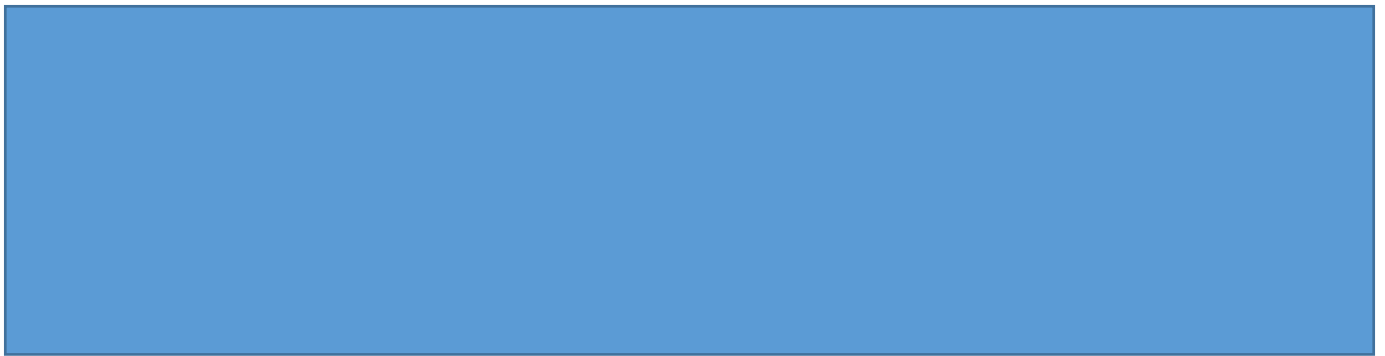
```
private let sessionQueue = DispatchQueue(label: "Session queue")
```

```
/// Context for converting sample buffer to UIImage
```

```
let context = CIContext()
```

```
/// Is permission to camera
```

```
private var permissionGranted = false
```



```
init(cameraPosition: AVCaptureDevice.Position) {
```

```
    super.init()
```

```
    position = cameraPosition
```

```
    checkPermission()
```

```
    sessionQueue.async { [unowned self] in
```

```
        self.configureSession()
```

```
    }
```

```
}
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.01-0151				
Инв. № подл.	Подп. и дата.	Взам. инв. №	Инв. № дубл.	Подп. и дата

```
/// Start frame extraction from camera
```

```
func start() {
    sessionQueue.async { [unowned self] in
        self.captureSession.startRunning()
    }
}
```

```
/// Check does app already has permission
```

```
case .notDetermined:
    requestPermission()
default:
    self.permissionGranted = false
    break
}
```

```
/// Request video permission
```

```
private func requestPermission() {
    sessionQueue.suspend()
    AVCaptureDevice.requestAccess(for: AVMediaType.video) { [unowned self] granted in
        self.permissionGranted = granted
    }
}
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.01-0151				
Инв. № подл.	Подп. и дата.	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

        self.sessionQueue.resume()

    }

}

/// Configure session

private func configureSession() {

    guard permissionGranted else { return }

    captureSession.sessionPreset = quality

    guard let captureDevice = selectCaptureDevice() else { return }

    guard let captureDeviceInput = try? AVCaptureDeviceInput(device: captureDevice) else { return
}

    guard captureSession.canAddInput(captureDeviceInput) else { return }

    guard captureSession.canAddOutput(videoOutput) else { return }

    captureSession.addOutput(videoOutput)

    guard let connection = videoOutput.connection(with: .video) else { return }

    guard connection.isVideoOrientationSupported else { return }

    connection.videoOrientation = .portrait

    guard connection.isVideoMirroringSupported else { return }

    connection.isVideoMirrored = position == .front

}

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.01-0151				
Инв. № подл.	Подп. и дата.	Взам. инв. №	Инв. № дубл.	Подп. и дата

```
/// Sends capured frames to delegate
```

```
///
```

```
/// - Parameters:
```

```
/// - output: Video output
```

```
/// - sampleBuffer: Buffer
```

```
/// - connection: Connection
```

```
}
```

```
}
```

```
/// Select capture device
```

```
///
```

```
/// - Returns: Object of type AVCaptureDevice?
```

```
private func selectCaptureDevice() -> AVCaptureDevice? {
```

```
    return AVCaptureDevice.default(.builtInWideAngleCamera, for: .video, position: position)
```

```
}
```

```
/// Creates UIImage from sample buffer
```

```
///
```

```
/// - Parameter sampleBuffer: Sample buffer
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.01-0151				
Инв. № подл.	Подп. и дата.	Взам. инв. №	Инв. № дубл.	Подп. и дата

/// - **Returns:** Image of type UIImage creates from sample buffer



```
}
```

```
}
```

1.4. MFNNFaceCoding.swift

```
//
```

```
// MFNNFaceCoding.swift
```

```
// MindFace
```

```
//
```

```
// Created by Nikita Konstantinovskiy on 28.05.2018.
```

```
// Copyright © 2018 nikkonst. All rights reserved.
```

```
//
```

```
import Foundation
```

```
import Vision
```

```
import CoreML
```

```
class MFNNFaceCoding {
```

```
    func getFaceCode(img: CGImage, completeon: @escaping ([Float]) -> Void) {
```

```
        do {
```

```
            let model = try VNCoreMLModel(for: MindFaceCoreML().model)
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.01-0151				
Инв. № подл.	Подп. и дата.	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

let request = VNCoreMLRequest(model: model) { (request, error) in

    guard let result = request.results as? [VNCoreMLFeatureValueObservation] else {return}

    var vec: [Float] = [Float]()

    vec.append(result.map { $0.floatValue })

    j += 1

}

completeon(vec)

}

DispatchQueue(label: "NN queue").async {

    let handler = VNImageRequestHandler(cgImage: img, orientation:
CGImagePropertyOrientation.up, options: [:])

    do {

        try handler.perform([request])

    }

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.01-0151				
Инв. № подл.	Подп. и дата.	Взам. инв. №	Инв. № дубл.	Подп. и дата


```

        catch {

            fatalError("Prediction")

        }

    }

}

catch {

    fatalError("Creation")

}

}

}

```

1.5. MFFaceNormalization.swift

```
//
```

```
// MFFaceNormalization.swift
```

```
//
```

```
import Foundation
```

```
import Accelerate
```

```
class MFFaceNormalization {
```

```
    func normalizeAndScaleTo96(face: MFFaceRect, image: UIImage) -> UIImage {
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.01-0151				
Инв. № подл.	Подп. и дата.	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

var A: [Double] = [Double(face.mainPoints[0].x), Double(face.mainPoints[1].x),
Double(face.mainPoints[2].x),
Double(face.mainPoints[0].y), Double(face.mainPoints[1].y),
Double(face.mainPoints[2].y),
1.0, 1.0, 1.0]

```

```

var X: [Double] = [0.194157, 0.7888591, 0.4949509,
0.16926692, 0.15817115, 0.5144414,
1.0, 1.0, 1.0]

```



```

A = self.invert(matrix: A)

```

```

var Result: [Double] = [0, 0, 0,
0, 0, 0,
0, 0, 0]

```

```

vDSP_mmulD(&X, 1, &A, 1, &Result, 1, vDSP_Length(3), vDSP_Length(3), vDSP_Length(3))

```

```

// a c tx

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.01-0151				
Инв. № подл.	Подп. и дата.	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

// b d ty

// 0 0 1

let transform = CGAffineTransform(a: CGFloat(Result[0]), b: -CGFloat(Result[3]), c:
CGFloat(Result[1]), d: -CGFloat(Result[4]), tx: CGFloat(Result[2]), ty: CGFloat(Result[5]))

var coreImage = image.ciImage

if (!(coreImage != nil)) {

    coreImage = CIImage.init(cgImage: image.cgImage!)

}

coreImage = coreImage?.transformed(by: transform)

```



```

let rectangle = CGRect(x: x,

    y: y,

    width: image.size.width,

    height: image.size.height)

let croppedCGImage:CGImage = (uiImage.cgImage?.cropping(to: rectangle))!

uiImage = UIImage(cgImage: croppedCGImage)

return self.resizeImageToSquare(image: uiImage, newWidth: 96)

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.01-0151				
Инв. № подл.	Подп. и дата.	Взам. инв. №	Инв. № дубл.	Подп. и дата

```
}
```

```
func invert(matrix : [Double]) -> [Double] {
```

```
    var error : __CLBK_integer = 0
```

```
    withUnsafeMutablePointer(to: &N) {
```

```
        dgetrf_($0, $0, &inMatrix, $0, &pivots, &error)
```

```
        dgetri_($0, &inMatrix, $0, &pivots, &workspace, $0, &error)
```

```
    }
```

```
    return inMatrix
```

```
}
```

```
func convert(cimage:CImage) -> UIImage
```

```
{
```

```
    let context:CImageContext = CImageContext.init(options: nil)
```

```
    let cgImage:CGImage = context.createCGImage(cimage, from: cimage.extent)!
```

```
    let image:UIImage = UIImage.init(cgImage: cgImage)
```

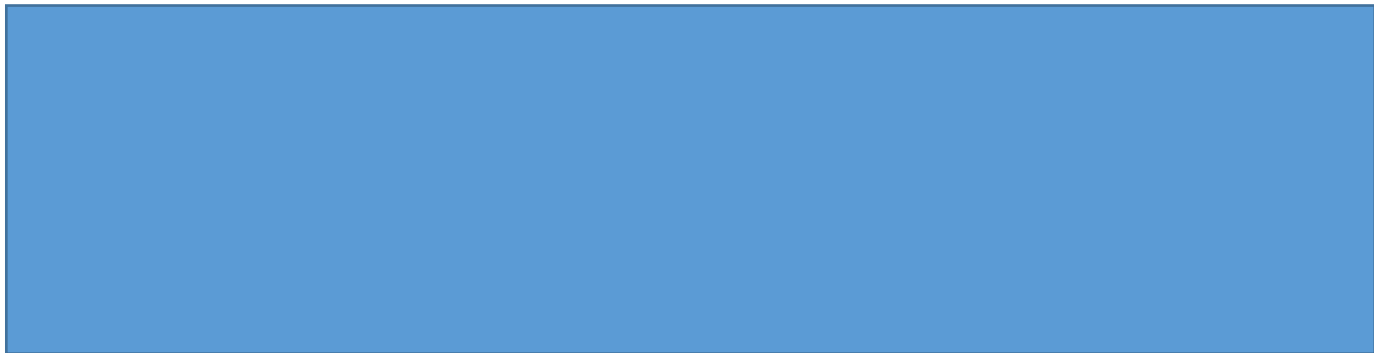
```
    return image
```

```
}
```

```
func resizeImageToSquare(image: UIImage, newWidth: CGFloat) -> UIImage {
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.01-0151				
Инв. № подл.	Подп. и дата.	Взам. инв. №	Инв. № дубл.	Подп. и дата

```
UIGraphicsBeginImageContext(CGSize(width: newWidth, height: newWidth))
```



```
return newImage!
```

```
}
```

```
}
```

1.6. MFFaceProcessorController.swift

```
//
```

```
// MFFaceProcessorController.swift
```

```
// MindFace
```

```
//
```

```
// Created by Nikita Konstantinovskiy on 30.03.2018.
```

```
// Copyright © 2018 nikkonst. All rights reserved.
```

```
//
```

```
import Foundation
```

```
import Vision
```

```
import Accelerate
```

```
internal class MFFaceProcessorController {
```

```
    /// Face detector
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.01-0151				
Инв. № подл.	Подп. и дата.	Взам. инв. №	Инв. № дубл.	Подп. и дата

```
private let faceDetector = MFFaceDetect()
```

```
private let faceNormaliser = MFFaceNormalization()
```

```
private let faceCoder = MFNNFaceCoding()
```

```
/// Draw
```

```
private let draw = MFDraw()
```



```
    })
```

```
    }
```

```
    else {
```

```
        completion([])
```

```
    }
```

```
}
```

```
func detectFaceAndDraw(image: UIImage?,
```

```
    completion: @escaping (_ faces: [MFFaceRect], _ image: UIImage?) -> Void) {
```

```
    if image != nil {
```

```
        faceDetector.detectFace(img: (image?.cgImage)!, completion: { (faces) in
```

```
            let imageWithRect = self.draw.drawFaceRect(faces: faces, image: image!)
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.01-0151				
Инв. № подл.	Подп. и дата.	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

        completion(faces, imageWithRect)
    })
}
else {
    completion([], nil)
}
}

```

```
let web = WebRequests()
```

```

func recogniseFaceAndDraw(image: UIImage?,
    completion: @escaping (_ faces: [MFFaceRect], _ image: UIImage?, _ str: String) -

```

```

    > Void) {

```

```
    for face in faces{
```

```
        let faceRect = self.draw.cropFaceRect(faces: [face], image: image!)[0]
```

```

        let normImage = self.faceNormaliser.normalizeAndScaleTo96(face: face, image:
        faceRect)

```

```
        self.faceCoder.getFaceCode(img: normImage.cgImage!, completeon: { (vec) in
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.01-0151				
Инв. № подл.	Подп. и дата.	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

        })
    }
}
else {
    completion([], nil, "")
}
})
}
else {
    completion([], nil, "")
}
}
}

```

1.7. MFFaceDetect.swift

```

//
// MFFaceDetect.swift
// MindFace
//
// Created by Nikita Konstantinovskiy on 16.02.2018.
// Copyright © 2018 nikkonst. All rights reserved.

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.01-0151				
Инв. № подл.	Подп. и дата.	Взам. инв. №	Инв. № дубл.	Подп. и дата


```
//
```

```
import Foundation
```

```
import Vision
```

```
/// Face detection class
```

```
internal class MFFaceDetect {
```

```
    /// Face detection on image
```

```
    ///
```

```
    /// - Parameters:
```

```
    /// - img: Image of type CGImage, on which the person is detected
```

```
    /// - completion: Completion with detecting result
```

```
}
```

```
var faces = [MFFaceRect]()
```

```
for face in faces {
```

```
    faces.append(MFFaceRect(vnFace: face))
```

```
}
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.01-0151				
Инв. № подл.	Подп. и дата.	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

        completion(faces)
    }

```

```

let imageRequestHandler = VNImageRequestHandler(cgImage: img, options: [:])
try? imageRequestHandler.perform([faceDetectionRequest])
}

```



```

        fatalError("Unexpected result type!")
    }

```

```

var faces = [MFFaceRect]()

for face in fases {

    faces.append(MFFaceRect(vnFace: face))

}

```

```

        completion(faces)
    }

```

```

let imageRequestHandler = VNImageRequestHandler(cgImage: img, options: [:])
try? imageRequestHandler.perform([faceDetectionRequest])
}

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.01-0151				
Инв. № подл.	Подп. и дата.	Взам. инв. №	Инв. № дубл.	Подп. и дата

```
}
```

1.8. L2Normalize.swift

```
//
```

```
// L2Normalize.swift
```

```
// MindFace
```

```
//
```

```
// Created by Nikita Konstantinovskiy on 05.04.2018.
```

```
// Copyright © 2018 nikkonst. All rights reserved.
```

```
//
```

```
import Foundation
```

```
import CoreML
```

```
import Accelerate
```

```
}
```

```
func setWeightData(_ weights: [Data]) throws {
```

```
    print(#function, weights)
```

```
}
```

```
func outputShapes(forInputShapes inputShapes: [[NSNumber]]) throws -> [[NSNumber]] {
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.01-0151				
Инв. № подл.	Подп. и дата.	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

print(#function, inputShapes)

return inputShapes
}

```

```

func evaluate(inputs: [MLMultiArray], outputs: [MLMultiArray]) throws {

    for i in 0..

```

```

// output = input^2

var countAsInt32 = Int32(count)

var pow2: [Float] = Array(repeating: 2, count: count)

vvpowf(optr, &pow2, iptr, &countAsInt32)

// output = sum(input^2)

vDSP_sve(optr, 1, optr, vDSP_Length(count))

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.01-0151				
Инв. № подл.	Подп. и дата.	Взам. инв. №	Инв. № дубл.	Подп. и дата

```
// output = sqrt(sum(input^2))

vvsqrtf(optr, optr, &countAsInt32)
```

```
// for j in 0..inputs.count {
//
//      print(output[j].floatValue)
//
// }

print(#function, inputs.count, outputs.count)

}

}
```

1.9. LRN.swift

```
//

// LRN.swift

// MindFace

//

// Created by Nikita Konstantinovskiy on 05.04.2018.

// Copyright © 2018 nikkonst. All rights reserved.

//
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.01-0151				
Инв. № подл.	Подп. и дата.	Взам. инв. №	Инв. № дубл.	Подп. и дата

```
import Foundation
```

```
import CoreML
```

```
import Accelerate
```

```
@objc(LPN) class LPN: NSObject, MLCustomLayer {
```

```
    required init(parameters: [String : Any]) throws {
```

```
        print(#function, parameters)
```

```
        super.init()
```

```
    }
```

```
    func setWeightData(_ weights: [Data]) throws {
```

```
        print(#function, weights)
```

```
    }
```

```
    func outputShapes(forInputShapes inputShapes: [[NSNumber]]) throws -> [[NSNumber]] {
```

```
        print(#function, inputShapes)
```

```
        return inputShapes
```

```
    }
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.01-0151				
Инв. № подл.	Подп. и дата.	Взам. инв. №	Инв. № дубл.	Подп. и дата

```
func evaluate(inputs: [MLMultiArray], outputs: [MLMultiArray]) throws {
```

```
    for i in 0..

```

```
        let input = inputs[i]
```

```
        let output = outputs[i]
```

```
        // output = input^2
```

```
        var countAsInt32 = Int32(count)
```

```
        var pow2: [Float] = Array(repeating: 2, count: count)
```

```
        vvpowf(optr, &pow2, iptr, &countAsInt32)
```

```
        // output = sum window 5(input^2)
```

```
        vDSP_vswsum(optr, 1, optr, 1, vDSP_Length(count-4), vDSP_Length(depth_radius))
```

```
        // output = alpha * (sum window 5(input^2))
```

```
        vDSP_vsmul(optr, 1, &alpha, optr, 1, vDSP_Length(count))
```

```
        // output = k + alpha * (sum window 5(input^2))
```

```
        vDSP_vsadd(optr, 1, &k, optr, 1, vDSP_Length(count))
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.01-0151				
Инв. № подл.	Подп. и дата.	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

// Output: iptr = (int *) 0x100000000 (countAsInt32 = 0)
        vvddivf(optr, iptr, optr, &countAsInt32)
    }

    print(#function, inputs, outputs.count)
}

}

```

1.10. MulConstant.swift

```

//
// MulConstant.swift
// MindFace
//
// Created by Nikita Konstantinovskiy on 05.04.2018.
// Copyright © 2018 nikkonst. All rights reserved.
//

```

```

import Foundation

import CoreML

import Accelerate

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.01-0151				
Инв. № подл.	Подп. и дата.	Взам. инв. №	Инв. № дубл.	Подп. и дата


```
@objc(MulConstant) class MulConstant: NSObject, MLCustomLayer {
```

```
    var const: Float = 9
```

```
    required init(parameters: [String : Any]) throws {
```

```
        print(#function, parameters)
```

```
        super.init()
```

```
    }
```

```
    func setWeightData(_ weights: [Data]) throws {
```

```
        print(#function, weights)
```

```
    }
```

```
    func evaluate(inputs: [MLMultiArray], outputs: [MLMultiArray]) throws {
```

```
        for i in 0..

```

```
            let input = inputs[i]
```

```
            let output = outputs[i]
```

```
            let count = input.count
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.01-0151				
Инв. № подл.	Подп. и дата.	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

let iptr = UnsafeMutablePointer<Float>(OpaquePointer(input.dataPointer))

let optr = UnsafeMutablePointer<Float>(OpaquePointer(output.dataPointer))


// output = input * const

vDSP_vsmul(iptr, 1, &const, optr, 1, vDSP_Length(count))

}


print(#function, inputs.count, outputs.count)

}

}

```

1.11. Sqrt.swift

```
//
```



```
//
```

```
import Foundation
```

```
import CoreML
```

```
import Accelerate
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.01-0151				
Инв. № подл.	Подп. и дата.	Взам. инв. №	Инв. № дубл.	Подп. и дата

```
@objc(Sqrt) class Sqrt: NSObject, MLCustomLayer {
```

```
    required init(parameters: [String : Any]) throws {
```

```
        print(#function, parameters)
```

```
        super.init()
```

```
    }
```

```
    func setWeightData(_ weights: [Data]) throws {
```

```
        print(#function, weights)
```

```
    }
```

```
    func evaluate(inputs: [MLMultiArray], outputs: [MLMultiArray]) throws {
```

```
        for i in 0..

```

```
            let input = inputs[i]
```

```
            let output = outputs[i]
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.01-0151				
Инв. № подл.	Подп. и дата.	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

    // output = sqrt(input)

    var countAsInt32 = Int32(count)

    vvsqrtf(optr, iptr, &countAsInt32)
}

print(#function, inputs.count, outputs.count)
}

}

```

1.12. Square.swift

```
//
```

```
// Square.swift
```



```
//
```

```
import Foundation
```

```
import CoreML
```

```
import Accelerate
```

```
@objc(Square) class Square: NSObject, MLCustomLayer {
```

```
    required init(parameters: [String : Any]) throws {
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.01-0151				
Инв. № подл.	Подп. и дата.	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

print(#function, parameters)

super.init()

}

```

```

func setWeightData(_ weights: [Data]) throws {

    print(#function, weights)

}

```



```

func evaluate(inputs: [MLMultiArray], outputs: [MLMultiArray]) throws {

    for i in 0..

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.01-0151				
Инв. № подл.	Подп. и дата.	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

    vvpowf(optr, &pow2, iptr, &countAsInt32)

}

```

```

    print(#function, inputs.count, outputs.count)

}

}

```

1.13. MFFaceRect.swift

```

//

// MFFaceRect.swift

// MindFace

//

// Created by Nikita Konstantinovskiy on 16.03.2018.

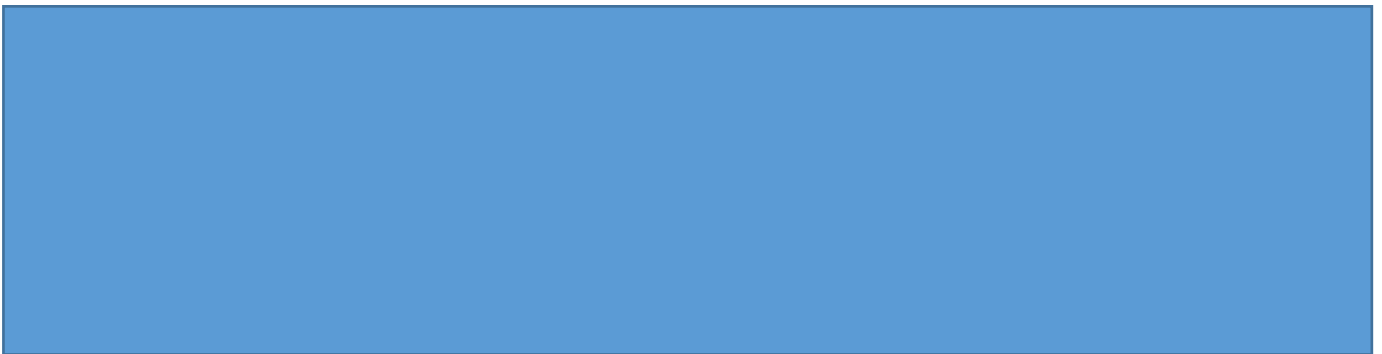
// Copyright © 2018 nikkonst. All rights reserved.

//

```

```
import Foundation
```

```
import Vision
```



```

init(vnFace: VNFaceObservation) {

    let w = vnFace.boundingBox.size.width

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.01-0151				
Инв. № подл.	Подп. и дата.	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

let h = vnFace.boundingBox.size.height

let x = vnFace.boundingBox.origin.x

let y = vnFace.boundingBox.origin.y

faceRect = CGRect(x: x, y: y, width: w, height: h)

```



```

}

else {

    mainPoints = [CGPoint]()

}

}

}

```

1.14. WebRequests.swift

```

//

// WebRequests.swift

// MindFace

//

// Created by Nikita Konstantinovskiy on 24.05.2018.

// Copyright © 2018 nikkonst. All rights reserved.

//

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.01-0151				
Инв. № подл.	Подп. и дата.	Взам. инв. №	Инв. № дубл.	Подп. и дата

```
import Foundation
```

```
class WebRequests {
```

```
    let urlImg = "http://192.168.0.105:5000/api/image"
```

```
    let url = "http://172.20.10.3:5000/api/"
```

```
    let vectorFunc = "vector"
```

```
    let loadFunc = "load"
```

```
    func vector(vec: [Float], completeon: @escaping (String)->()) {
```

```
        let session = URLSession(configuration: URLSessionConfiguration.default)
```

```
        guard let url = URL(string: url + vectorFunc) else { return }
```

```
        var request = URLRequest(url: url)
```

```
        request.httpMethod = "POST"
```

```
        let body = ["vector": vec]
```

```
        do {
```

```
            let dataTask = session.dataTask(with: request) { (data, response, error) in
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.01-0151				
Инв. № подл.	Подп. и дата.	Взам. инв. №	Инв. № дубл.	Подп. и дата


```

if let error = error {
    print("Something went wrong: \(error)")
}

```



```

        completeon(outputStr!)
    }

    dataTask.resume()
}

catch {
    return
}
}

func load(completeon: @escaping (String)->()) {
    let session = URLSession(configuration: URLSessionConfiguration.default)

    guard let url = URL(string: url + loadFunc) else { return }

    var request = URLRequest(url: url)

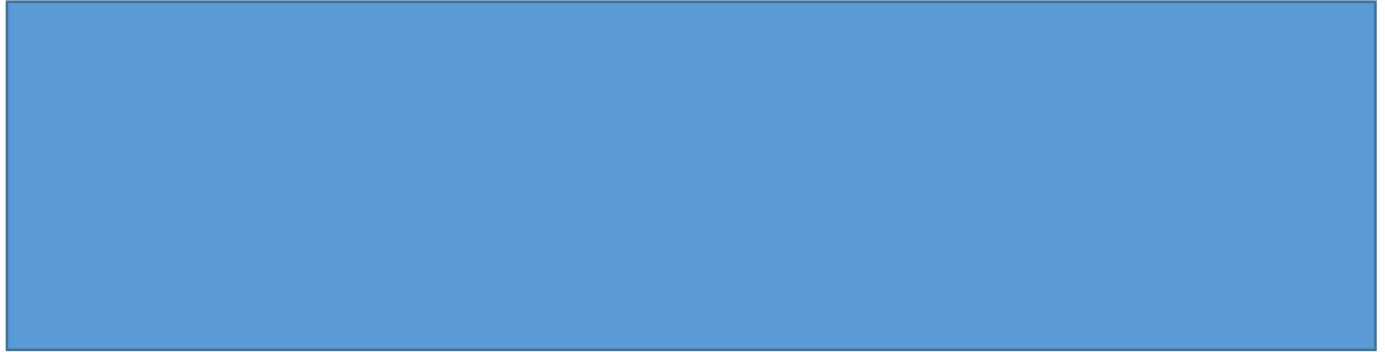
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.01-0151				
Инв. № подл.	Подп. и дата.	Взам. инв. №	Инв. № дубл.	Подп. и дата

```
request.httpMethod = "POST"
```

```
let body = ["load": 1]
```

```
do {
```



```
let dataTask = session.dataTask(with: request) { (data, response, error) in
```

```
    if let error = error {
```

```
        print("Something went wrong: \(error)")
```

```
    }
```

```
    if let response = response {
```

```
        print("Response: \n \(response)")
```

```
    }
```

```
let outputStr = String(data: data!, encoding: String.Encoding.utf8) as String!
```

```
completeon(outputStr!)
```

```
}
```

```
dataTask.resume()
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.01-0151				
Инв. № подл.	Подп. и дата.	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

    }

    catch {

        return

    }

}

```

```
func submit(image: UIImage, completeon: @escaping (String)->()) {
```

```
    let session = URLSession(configuration: URLSessionConfiguration.default)
```

```
    guard let url = URL(string: urlImg) else { return }
```

```
    var request = URLRequest(url: url)
```

```
//    let dataTask = session.dataTask(with: request)
```

```
let dataTask = session.dataTask(with: request) { (data, response, error) in
```

```
    if let error = error {
```

```
        print("Something went wrong: \(error)")
```

```
    }
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.01-0151				
Инв. № подл.	Подп. и дата.	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

if let response = response {
    print("Response: \n \((response)")
}

let outputStr = String(data: data!, encoding: String.Encoding.utf8) as String!

completeon(outputStr!)
}

dataTask.resume()
}
}

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.01-0151				
Инв. № подл.	Подп. и дата.	Взам. инв. №	Инв. № дубл.	Подп. и дата

ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

[illegible]

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.01-0151				
Инв. № подл.	Подп. и дата.	Взам. инв. №	Инв. № дубл.	Подп. и дата