金信诺iOS代码

//

//  GroupViewController.swift

//  JinXingNuoProject

//

//  Created by 刘俊威 on 2019/7/5.

//  Copyright © 2019 wadeLiu.com. All rights reserved.

//

import UIKit

class GroupViewController: JLKFBaseViewController, UITableViewDelegate, UITableViewDataSource {

    // MARK: - properties

    lazy var tableView: UITableView = {

        let tableview = UITableView.init(frame: .zero, style: .grouped)

        tableview.delegate = self as UITableViewDelegate

        tableview.dataSource = self as UITableViewDataSource

        tableview.backgroundColor = HexNumber(r: 243, g: 245, b: 248, alpha: 1)

        tableview.separatorStyle = UITableViewCell.SeparatorStyle.none

        tableview.tableFooterView = UIView.init(frame: .zero)

        tableview.bounces = false

        tableview.register(UINib(nibName: "SwitchTableViewCell", bundle: nil), forCellReuseIdentifier: "SwitchTableViewCell")

        tableview.register(UINib(nibName: "NormalTableViewCell", bundle: nil), forCellReuseIdentifier: "NormalTableViewCell")

        tableview.register(UINib(nibName: "TextFieldTableViewCell", bundle: nil), forCellReuseIdentifier: "TextFieldTableViewCell")

        return tableview

    }()

    lazy var dataSource: [TextTableCellModel] = []

    lazy var footerview: TabelViewFooterView = {

        let footerView = TabelViewFooterView.init(frame: .zero)

        footerView.isEnable = true

        footerView.confirmBlock = {

            if self.group == nil {

                // 添加

                self.requestAddChange()

            } else {

                // 修改

                self.requestSaveChange()

            }

        }

        return footerView

    }()

    /// 为空 添加 、 不为空 则为修改

    open var group: GroupModel?

    // 保存旧设备名字 修改的时候要用

    var oldGroupName: String?

    // 选择要添加的数组

    var addMacArray: Array<String>?

    // 所有已添加分组的名字

    lazy var deviceNames: [String] = []

    // MARK: - ViewController Life CirCle

    override func viewDidLoad() {

        super.viewDidLoad()

        initview()

        reloadData()

    }

    override func viewWillAppear(\_ animated: Bool) {

        super.viewWillAppear(animated)

        SocketManager.shared().delegate = self

    }

    override func viewWillDisappear(\_ animated: Bool) {

        super.viewWillDisappear(animated)

        SVProgressHUD.dismiss()

        SocketManager.shared().delegate = nil

    }

    func reloadData() {

        if group != nil {

            CocoaAsyncSocketManager.shared().getInformationCommand(command: "RequestParentalSettings", tag: .RequestParentalSetings)

        } else {

            initdata()

        }

    }

    private func requestAddChange() {

        let on = dataSource[0].on

        let name = dataSource[1].textfieldText

        // 分组名验证 不能和其他设备同名

        var tempNames = deviceNames// 此处因为是存储stirng类型，所以会深拷贝

        if let model = group {

            tempNames.remove(at: model.row!)

        }

        if tempNames.contains(name ?? "") {

            // 新名字和其他的分组名有重合

            let alert = UIAlertController.init(title: language("请重新输入分组名"), message: language("分组名不能和其他分组的名字相同"), preferredStyle: .alert)

            self.present(alert, animated: true, completion: nil)

            DispatchQueue.main.asyncAfter(deadline: DispatchTime.now() + 2) {

                alert.dismiss(animated: true, completion: nil)

            }

            return

        }

        // 分组名不能为空

        if name == nil || name == "" {

            let alert = UIAlertController.init(title: language("分组名不能为空"), message: "", preferredStyle: .alert)

            self.present(alert, animated: true, completion: nil)

            DispatchQueue.main.asyncAfter(deadline: DispatchTime.now() + 1) {

                alert.dismiss(animated: true, completion: nil)

            }

            return

        }

        let timeModel = dataSource[2]

        var time = timeModel.detailTitle

        if let temp = time {

            let float = NSString.init(string: temp).floatValue

            time = String(float)

        } else {

            let alert = UIAlertController.init(title: language("未选择时间"), message: "", preferredStyle: .alert)

            self.present(alert, animated: true, completion: nil)

            DispatchQueue.main.asyncAfter(deadline: DispatchTime.now() + 1) {

                alert.dismiss(animated: true, completion: nil)

            }

            return

        }

        var tempArray = [Dictionary<String, String>]()

        if let macArray = addMacArray {

            for mac in macArray {

                tempArray.append(["mac":mac])

            }

        }

        let params = [

            "ParentalSetings":[

                [

                    "parentalGroupsSet":"add",

                    "networkEnable":on ? "on" : "off",

                    "groupName":name ?? "",

                    "groupTime":time ?? "",

                    "addDev":tempArray,

                ]

            ]

        ]

        SVProgressHUD.show()

        CocoaAsyncSocketManager.shared().sendSetCommand(setDictionary: params, tag: .NONE)

    }

    private func requestSaveChange() {

        /// 对比设备mac数组和新添加mac数组  获取分组添加和删除数组

        ///

        /// - Returns: 添加的mac字典，删除的mac字典

        func compareDeviceMacArrayAndAddMacArray() -> (addDevArray: [Dictionary<String, String>], deleteDevArray: [Dictionary<String, String>]) {

            // 对比设备mac数组和新添加mac数组  获取分组添加和删除数组

            var tempAddDevArray = [Dictionary<String, String>]()

            var tempDeleteDevArray = [Dictionary<String, String>]()

            if let allMacArray = addMacArray {

                // 本身设备自带mac数组存在与否

                if group?.groupDev != nil && group?.groupDev?.count != 0 {

                    // 设备带有mac数组

                    // 做对比

                    var deviceMacArray = [String]()

                    for groupDev in group!.groupDev! {

                        if let mac = groupDev.mac {

                            deviceMacArray.append(mac)

                        }

                    }

                    // 取出所有的添加的数组

                    for mac in allMacArray {

                        // 如果选择的mac在设备数组中不存在 则为新添加mac

                        if deviceMacArray.contains(mac) == false {

                            tempAddDevArray.append(["mac":mac])

                        }

                    }

                    // 取出所有的设备数组

                    for mac in deviceMacArray {

                        // 如果设备的mac在添加数组中不存在 则为删除的mac

                        if allMacArray.contains(mac) == false {

                            tempDeleteDevArray.append(["mac":mac])

                        }

                    }

                } else {

                    // 设备不带有mac数组

                    for mac in allMacArray {

                        tempAddDevArray.append(["mac":mac])

                    }

                }

            }

            return (tempAddDevArray, tempDeleteDevArray)

        }

        let on = dataSource[0].on

        let name = dataSource[1].textfieldText

        // 分组名验证 不能和其他设备同名

        var tempNames = deviceNames// 此处因为是存储stirng类型，所以会深拷贝

        if let model = group {

            tempNames.remove(at: model.row!)

        }

        if tempNames.contains(name ?? "") {

            // 新名字和其他的分组名有重合

            let alert = UIAlertController.init(title: language("请重新输入分组名"), message: language("分组名不能和其他分组的名字相同"), preferredStyle: .alert)

            self.present(alert, animated: true, completion: nil)

            DispatchQueue.main.asyncAfter(deadline: DispatchTime.now() + 1) {

                alert.dismiss(animated: true, completion: nil)

            }

            return

        }

        // 分组名不能为空

        if name == nil || name == "" {

            let alert = UIAlertController.init(title: language("分组名不能为空"), message: "", preferredStyle: .alert)

            self.present(alert, animated: true, completion: nil)

            DispatchQueue.main.asyncAfter(deadline: DispatchTime.now() + 1) {

                alert.dismiss(animated: true, completion: nil)

            }

            return

        }

        // 时间

        let timeModel = dataSource[2]

        var time = timeModel.detailTitle

        if let temp = time {

            // 去除"小时"

            let float = NSString.init(string: temp).floatValue

            time = String(float)

        } else {

            let alert = UIAlertController.init(title: language("未选择时间"), message: "", preferredStyle: .alert)

            self.present(alert, animated: true, completion: nil)

            DispatchQueue.main.asyncAfter(deadline: DispatchTime.now() + 1) {

                alert.dismiss(animated: true, completion: nil)

            }

            return

        }

        let tempData = compareDeviceMacArrayAndAddMacArray()

        let params = [

            "ParentalSetings":[

                [

                    "parentalGroupsSet":"modify",

                    "networkEnable":on ? "on" : "off",

                    "groupName":oldGroupName ?? "",// 旧的名字

                    "groupNameChange":name ?? "",// 修改名字的时候需要传的参数

                    "groupTime":time ?? "",

                    "addDev":tempData.addDevArray,

                    "deleteDev":tempData.deleteDevArray,

                ]

            ]

        ]

        SVProgressHUD.show()

        CocoaAsyncSocketManager.shared().sendSetCommand(setDictionary: params, tag: .NONE)

    }

    // MARK: - custom ui

    func initview() {

        self.title = language("分组")

        self.view.addSubview(self.tableView)

        self.tableView.snp.makeConstraints { (make) in

            make.left.right.bottom.equalTo(self.view)

            make.top.equalTo(self.navigationBarView.snp.bottom)

        }

    }

    // MARK: - data

    func initdata() {

        dataSource.removeAll()

        let titleArray = [language("访问互联网"), language("分组名称"), language("控制时长"), language("接入设备控制")]

        let typeArray = [UITableViewCellType.mySwitch, UITableViewCellType.textfield, UITableViewCellType.normal, UITableViewCellType.normal]

        for index in 0..<titleArray.count {

            let model = TextTableCellModel.init()

            model.title = titleArray[index]

            model.cellType = typeArray[index]

            model.textfieldPlaceholder = language("点击设置")

            if group != nil {

                if index == 0 {

                    if group?.networkEnable == "on" {

                        model.on = true

                    } else if group?.networkEnable == "off" {

                        model.on = false

                    } else {

                        model.on = true

                    }

                } else if index == 1 {

                    model.textfieldText = group?.groupName

                } else if index == 2 {

                    model.detailTitle = group?.time

                }

            }

            dataSource.append(model)

        }

        tableView.reloadData()

    }

}

extension GroupViewController: SocketManagerProtocol {

    func socketManager(\_ socketManager: SocketManager, parentalSetingsModel: ParentalSetingsModel) {

        let parentalSetting = parentalSetingsModel.ParentalSetings?.first

        if let groups = parentalSetting?.group {

            if group != nil {

                let index = (group?.row)!

                let group = groups[index]

                let model = GroupModel.init()

                model.groupName = group.groupName

                model.time = group.groupTime

                model.networkEnable = group.networkEnable

                model.row = index

                model.groupDev = group.groupDev

                self.oldGroupName = model.groupName

                self.group = model// 替换掉传进来的模型

                self.initdata()

            }

        }

    }

    func socketManager(\_ socketManager: SocketManager, requestResultSuccess: Bool) {

        SVProgressHUD.dismiss()

        let alert = UIAlertController.init(title: language("保存成功"), message: "", preferredStyle: .alert)

        self.present(alert, animated: true, completion: nil)

        DispatchQueue.main.asyncAfter(deadline: DispatchTime.now() + 1.5) {

            alert.dismiss(animated: true, completion: nil)

            self.navigationController?.popViewController(animated: true)

        }

    }

    func socketManager(\_ socketManager: SocketManager, requestResultError: CommandType) {

        SVProgressHUD.dismiss()

    }

}

extension GroupViewController {

    func numberOfSections(in tableView: UITableView) -> Int {

        return 1

    }

    func tableView(\_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {

        return dataSource.count

    }

    func tableView(\_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {

        let model = self.dataSource[indexPath.row]

        if model.cellType == .normal {

            let cell = tableView.dequeueReusableCell(withIdentifier: "NormalTableViewCell", for: indexPath) as! NormalTableViewCell

            cell.model(model)

            return cell

        } else if model.cellType == .textfield {

            let cell = tableView.dequeueReusableCell(withIdentifier: "TextFieldTableViewCell", for: indexPath) as! TextFieldTableViewCell

            cell.model(model)

            return cell

        } else if model.cellType == .mySwitch {

            let cell = tableView.dequeueReusableCell(withIdentifier: "SwitchTableViewCell", for: indexPath) as! SwitchTableViewCell

            cell.model(model)

            cell.onBlock = { (on) in

                model.on = on

            }

            return cell

        } else {

            let cell = tableView.dequeueReusableCell(withIdentifier: "cell", for: indexPath)

            return cell

        }

    }

    func tableView(\_ tableView: UITableView, viewForHeaderInSection section: Int) -> UIView? {

        return UIView.init(frame: .zero)

    }

    func tableView(\_ tableView: UITableView, heightForHeaderInSection section: Int) -> CGFloat {

        return 0.001

    }

    func tableView(\_ tableView: UITableView, viewForFooterInSection section: Int) -> UIView? {

        self.footerview.frame = CGRect(x: 0, y: 0, width: kScreenWidth, height: 300)

        return self.footerview

    }

    func tableView(\_ tableView: UITableView, heightForFooterInSection section: Int) -> CGFloat {

        return 300

    }

    func tableView(\_ tableView: UITableView, heightForRowAt indexPath: IndexPath) -> CGFloat {

        return 55

    }

    func tableView(\_ tableView: UITableView, didSelectRowAt indexPath: IndexPath) {

        if indexPath.row == 2 {

            let vc = ControlTimeViewController.init()

            vc.callBlock = { (text) in

                let textModel = self.dataSource[2]

                textModel.detailTitle = text

                self.tableView.reloadData()

            }

            self.navigationController?.pushViewController(vc, animated: true)

        } else if indexPath.row == 3 {

            let vc = ConnectSettingViewController.init()

            var deviceMacArray = [String]()

            if group != nil {

                if let groupDevs = group!.groupDev {

                    for groupDev in groupDevs {

                        if let mac = groupDev.mac {

                            deviceMacArray.append(mac)

                        }

                    }

                }

            }

            vc.allMacArray = deviceMacArray

            vc.callBlock = { (array) in

                self.addMacArray = array

            }

            self.navigationController?.pushViewController(vc, animated: true)

        }

    }

}

//

//  ParentsControlViewController.swift

//  JinXingNuoProject

//

//  Created by 刘俊威 on 2019/7/5.

//  Copyright © 2019 wadeLiu.com. All rights reserved.

//

import UIKit

class ParentsControlViewController: JLKFBaseViewController, UITableViewDelegate, UITableViewDataSource {

    // MARK: - properties

    lazy var tableView: UITableView = {

        let tableview = UITableView.init(frame: .zero, style: .grouped)

        tableview.delegate = self as UITableViewDelegate

        tableview.dataSource = self as UITableViewDataSource

        tableview.backgroundColor = HexNumber(r: 243, g: 245, b: 248, alpha: 1)

        tableview.separatorStyle = UITableViewCell.SeparatorStyle.none

        tableview.tableFooterView = UIView.init(frame: .zero)

        tableview.bounces = false

        tableview.register(UINib(nibName: "NewGroupTableViewCell", bundle: nil), forCellReuseIdentifier: "NewGroupTableViewCell")

        tableview.register(UINib(nibName: "GroupTableViewCell", bundle: nil), forCellReuseIdentifier: "GroupTableViewCell")

        return tableview

    }()

    lazy var dataSource: [GroupModel] = []

    lazy var footerview: TabelViewFooterView = {

        let view = TabelViewFooterView.init(frame: .zero)

        return view

    }()

    lazy var deviceNames: [String] = []

    // MARK: - ViewController Life CirCle

    override func viewDidLoad() {

        super.viewDidLoad()

        initview()

        initdata()

    }

    override func viewWillAppear(\_ animated: Bool) {

        super.viewWillAppear(animated)

        SocketManager.shared().delegate = self

        reloadData()

    }

    override func viewWillDisappear(\_ animated: Bool) {

        super.viewWillDisappear(animated)

        SocketManager.shared().delegate = nil

    }

    func reloadData() {

        CocoaAsyncSocketManager.shared().getInformationCommand(command: "RequestParentalSettings", tag: .RequestParentalSetings)

    }

    func requestDeleteCommand(model: GroupModel) {

        let params = ["ParentalSetings":

            [

                [

                    "parentalGroupsSet":"delete",

                    "groupName":model.groupName

                    ]

            ]

        ]

        CocoaAsyncSocketManager.shared().sendSetCommand(setDictionary: params, tag: .NONE)

    }

    // MARK: - custom ui

    func initview() {

        self.title = language("家长控制")

        self.view.addSubview(self.tableView)

        self.tableView.snp.makeConstraints { (make) in

            make.left.right.bottom.equalTo(self.view)

            make.top.equalTo(self.navigationBarView.snp.bottom)

        }

    }

    // MARK: - data

    func initdata() {

//        let timeArray = ["2小时", "3小时", "4小时", "5小时", "6小时"]

//        for index in 0..<timeArray.count {

//            let model = GroupModel.init()

//            model.groupName = "分组名称..."

//            model.time = timeArray[index]

//            dataSource.append(model)

//        }

    }

}

var ParentsControlAllDeviceMac: [String] = []

extension ParentsControlViewController: SocketManagerProtocol {

    func socketManager(\_ socketManager: SocketManager, parentalSetingsModel: ParentalSetingsModel) {

        let parentalSetting = parentalSetingsModel.ParentalSetings?.first

        dataSource.removeAll()

        deviceNames.removeAll()

        ParentsControlAllDeviceMac.removeAll()

        if let groups = parentalSetting?.group {

            var index: Int = 0

            for group in groups {

                let model = GroupModel.init()

                model.groupName = group.groupName

                model.time = group.groupTime

                model.networkEnable = group.networkEnable

                model.row = index

                dataSource.append(model)

                index += 1

                // 保存所有设备的名字

                deviceNames.append(group.groupName ?? "")

                if let groupDev = group.groupDev {

                    for mac in groupDev {

                        if let tempMac = mac.mac {

                            ParentsControlAllDeviceMac.append(tempMac)

                        }

                    }

                }

            }

        }

        tableView.reloadData()

    }

    func socketManager(\_ socketManager: SocketManager, requestResultError: CommandType) {

    }

    func socketManager(\_ socketManager: SocketManager, requestResultSuccess: Bool) {

        reloadData()

    }

}

extension ParentsControlViewController {

    func numberOfSections(in tableView: UITableView) -> Int {

        return 2

    }

    func tableView(\_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {

        if section == 0 {

            return 1

        } else {

            return dataSource.count

        }

    }

    func tableView(\_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {

        if indexPath.section == 0 {

            let cell = tableView.dequeueReusableCell(withIdentifier: "NewGroupTableViewCell", for: indexPath) as! NewGroupTableViewCell

            return cell

        } else {

            let cell = tableView.dequeueReusableCell(withIdentifier: "GroupTableViewCell", for: indexPath) as! GroupTableViewCell

            cell.model(model: dataSource[indexPath.row])

            cell.deleteBlock = { (model) in

                self.requestDeleteCommand(model: model!)

            }

            return cell

        }

    }

    func tableView(\_ tableView: UITableView, viewForHeaderInSection section: Int) -> UIView? {

        return UIView.init(frame: .zero)

    }

    func tableView(\_ tableView: UITableView, heightForHeaderInSection section: Int) -> CGFloat {

        return 0.001

    }

    func tableView(\_ tableView: UITableView, viewForFooterInSection section: Int) -> UIView? {

//        if section == 1 {

//            self.footerview.frame = CGRect(x: 0, y: 0, width: kScreenWidth, height: 300)

//            return self.footerview

//        }

        return UIView.init(frame: .zero)

    }

    func tableView(\_ tableView: UITableView, heightForFooterInSection section: Int) -> CGFloat {

//        if section == 1 {

//            return 300

//        }

        return 0.001

    }

    func tableView(\_ tableView: UITableView, heightForRowAt indexPath: IndexPath) -> CGFloat {

        if indexPath.section == 0 {

            return 55

        } else {

            return 75

        }

    }

    func tableView(\_ tableView: UITableView, didSelectRowAt indexPath: IndexPath) {

        if indexPath.section == 0 {

            let vc = GroupViewController.init()

            vc.deviceNames = deviceNames

            self.navigationController?.pushViewController(vc, animated: true)

        }

        if indexPath.section == 1 {

            let vc = GroupViewController.init()

            let group = dataSource[indexPath.row]

            vc.group = group

            vc.deviceNames = deviceNames

            self.navigationController?.pushViewController(vc, animated: true)

        }

    }

}

//

//  SignalViewController.swift

//  JinXingNuoProject

//

//  Created by 刘俊威 on 2019/7/4.

//  Copyright © 2019 wadeLiu.com. All rights reserved.

//

import UIKit

class SignalViewController: JLKFBaseViewController, UITableViewDelegate, UITableViewDataSource {

    // MARK: - properties

    lazy var tableView: UITableView = {

        let tableview = UITableView.init(frame: .zero, style: .grouped)

        tableview.delegate = self as UITableViewDelegate

        tableview.dataSource = self as UITableViewDataSource

        tableview.backgroundColor = HexNumber(r: 243, g: 245, b: 248, alpha: 1)

        tableview.separatorStyle = UITableViewCell.SeparatorStyle.none

        tableview.tableFooterView = UIView.init(frame: .zero)

        tableview.bounces = false

        tableview.register(UINib(nibName: "SignalTableViewCell", bundle: nil), forCellReuseIdentifier: "SignalTableViewCell")

        return tableview

    }()

    lazy var dataSource: [Ssid] = []

    lazy var hud: UIActivityIndicatorView = {

        let hud = UIActivityIndicatorView.init()

        hud.color = .black

        return hud

    }()

    // MARK: - ViewController Life CirCle

    override func viewDidLoad() {

        super.viewDidLoad()

        initview()

        initdata()

        reloadData()

    }

    override func viewWillAppear(\_ animated: Bool) {

        super.viewWillAppear(animated)

        SocketManager.shared().delegate = self

    }

    override func viewWillDisappear(\_ animated: Bool) {

        super.viewWillDisappear(animated)

        SocketManager.shared().delegate = nil

    }

    func reloadData() {

        SVProgressHUD.showMask()

        CocoaAsyncSocketManager.shared().getInformationCommand(command: "RequestEnvironmentSettings", tag: .RequestEnvironmentGettings)

    }

    // MARK: - custom ui

    func initview() {

        self.title = language("信号环境")

        self.view.addSubview(self.tableView)

        self.tableView.snp.makeConstraints { (make) in

            make.left.right.bottom.equalTo(self.view)

            make.top.equalTo(self.navigationBarView.snp.bottom)

        }

        view.addSubview(hud)

        hud.snp.makeConstraints { (make) in

            make.center.equalToSuperview()

        }

        hud.startAnimating()

    }

    // MARK: - data

    func initdata() {

        tableView.reloadData()

    }

}

extension SignalViewController: SocketManagerProtocol {

    func socketManager(\_ socketManager: SocketManager, environmentGettingsModel: EnvironmentGettingsModel) {

        SVProgressHUD.dismissMask()

        hud.isHidden = true

        let environmentSetting = environmentGettingsModel.EnvironmentGettings?.first

        dataSource.removeAll()

        dataSource = environmentSetting?.ssids ?? []

        // 排序逻辑 按照信号值由高到低排序

        dataSource = dataSource.sorted { (ssid1, ssid2) -> Bool in

            return Int(ssid1.signal!)! > Int(ssid2.signal!)!

        }

        tableView.reloadData()

    }

    func socketManager(\_ socketManager: SocketManager, requestResultSuccess: Bool) {

        hud.isHidden = true

    }

    func socketManager(\_ socketManager: SocketManager, requestResultError: CommandType) {

        hud.isHidden = true

    }

}

extension SignalViewController {

    func tableView(\_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {

        return self.dataSource.count

    }

    func tableView(\_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {

        let cell = tableView.dequeueReusableCell(withIdentifier: "SignalTableViewCell", for: indexPath) as! SignalTableViewCell

        cell.model(dataSource[indexPath.row])

        return cell

    }

    func tableView(\_ tableView: UITableView, heightForHeaderInSection section: Int) -> CGFloat {

        return 0.001

    }

    func tableView(\_ tableView: UITableView, viewForHeaderInSection section: Int) -> UIView? {

        return UIView.init(frame: .zero)

    }

    func tableView(\_ tableView: UITableView, heightForRowAt indexPath: IndexPath) -> CGFloat {

        return 100

    }

    func tableView(\_ tableView: UITableView, didSelectRowAt indexPath: IndexPath) {

    }

}

//

//  WirelessSettingViewController.swift

//  JinXingNuoProject

//

//  Created by 刘俊威 on 2019/7/4.

//  Copyright © 2019 wadeLiu.com. All rights reserved.

//

import UIKit

class WirelessSettingViewController: JLKFBaseViewController, UITableViewDelegate, UITableViewDataSource {

    // MARK: - properties

    lazy var tableView: UITableView = {

        let tableview = UITableView.init(frame: .zero, style: .grouped)

        tableview.delegate = self as UITableViewDelegate

        tableview.dataSource = self as UITableViewDataSource

        tableview.backgroundColor = HexNumber(r: 243, g: 245, b: 248, alpha: 1)

        tableview.separatorStyle = UITableViewCell.SeparatorStyle.none

        tableview.tableFooterView = UIView.init(frame: .zero)

        tableview.bounces = false

        tableview.register(UINib(nibName: "TextFieldTableViewCell", bundle: nil), forCellReuseIdentifier: "TextFieldTableViewCell")

        tableview.register(UINib(nibName: "WirelessPasswordCell", bundle: nil), forCellReuseIdentifier: "WirelessPasswordCell")

        return tableview

    }()

    lazy var dataSource: [TextTableCellModel] = []

    lazy var footerview: TabelViewFooterView = {

        let view = TabelViewFooterView.init(frame: .zero)

        view.confirmBlock = {

            self.sendCommand()

        }

        return view

    }()

    lazy var button: UIButton = {

        let button = UIButton.init(type: .custom)

        button.setImage(UIImage.init(named: "login\_icon\_unhide"), for: .normal)

        button.setImage(UIImage.init(named: "login\_icon\_hide"), for: .selected)

        button.addTarget(self, action: #selector(buttonAction(\_:)), for: .touchUpInside)

        return button

    }()

    var textFields: Array<UITextField> = []

    var hasEdited = false

    var pwdTextField: UITextField?

    // MARK: - ViewController Life CirCle

    override func viewDidLoad() {

        super.viewDidLoad()

        initview()

        initdata()

        reloadData()

    }

    override func viewWillAppear(\_ animated: Bool) {

        super.viewWillAppear(animated)

        SocketManager.shared().delegate = self

    }

    override func viewWillDisappear(\_ animated: Bool) {

        super.viewWillDisappear(animated)

        SocketManager.shared().delegate = nil

    }

    func reloadData() {

        CocoaAsyncSocketManager.shared().getInformationCommand(command: "RequestWirelessSettings", tag: .RequestWirelessSettings)

    }

    private func sendCommand() {

        var name = ""

        var pwd = ""

        for textModel in self.dataSource {

            if textModel.title == language("无线网络名称") {

                name = textModel.textfieldText!

            } else if textModel.title == language("无线网络密码") {

                pwd = textModel.textfieldText!

            }

        }

        if pwd.count < 8 {

            let alert = UIAlertController.init(title: language("密码格式错误"), message: language("需为八位数或以上"), preferredStyle: .alert)

            self.present(alert, animated: true, completion: nil)

            DispatchQueue.main.asyncAfter(deadline: DispatchTime.now() + 2) {

                alert.dismiss(animated: true, completion: nil)

            }

            return

        }

        if pwd.count > 32 {

            let alert = UIAlertController.init(title: language("密码格式错误"), message: language("需为三十二位数以下"), preferredStyle: .alert)

            self.present(alert, animated: true, completion: nil)

            DispatchQueue.main.asyncAfter(deadline: DispatchTime.now() + 2) {

                alert.dismiss(animated: true, completion: nil)

            }

            return

        }

        let params = [

            "WirelessSettings":[

                [

                    "ssid":name,

                    "pskValue":pwd,

                    "firstLogin":"true"

                ]

            ]

        ]

        CocoaAsyncSocketManager.shared().sendSetCommand(setDictionary: params, tag: .RequestWirelessSettings)

    }

    // MARK: - custom ui

    func initview() {

        self.title = language("无线设置")

        self.view.addSubview(self.tableView)

        self.tableView.snp.makeConstraints { (make) in

            make.left.right.bottom.equalTo(self.view)

            make.top.equalTo(self.navigationBarView.snp.bottom)

        }

    }

    // MARK: - data

    func initdata() {

        let titleArray = [language("无线网络名称"), language("无线网络密码")]

        dataSource.removeAll()

        for index in 0..<titleArray.count {

            let model = TextTableCellModel.init()

            model.title = titleArray[index]

            if index == 0 {

                model.cellType = .textfield

            } else {

                model.cellType = .password

            }

            dataSource.append(model)

        }

        tableView.reloadData()

    }

    // MARK: - button action

    @objc func buttonAction(\_ sender: UIButton) {

        sender.isSelected = !sender.isSelected

        let model = dataSource[1]

        model.isSecureTextEntry = sender.isSelected

        tableView.reloadData()

    }

}

extension WirelessSettingViewController: SocketManagerProtocol {

    func socketManager(\_ socketManager: SocketManager, wirelessSettingsModel: WirelessSettingsModel) {

        let wirelessSettings = wirelessSettingsModel.WirelessSettings?.first

        let ssidModel = dataSource[0]

        ssidModel.textfieldText = wirelessSettings?.ssid

        let pskValueModel = dataSource[1]

        pskValueModel.textfieldText = wirelessSettings?.pskValue

        tableView.reloadData()

    }

    func socketManager(\_ socketManager: SocketManager, requestResultSuccess: Bool) {

        let alert = UIAlertController.init(title: language("设置成功"), message: language("设置成功，软件即将退出"), preferredStyle: .alert)

        self.present(alert, animated: true) {

            DispatchQueue.main.asyncAfter(deadline: DispatchTime.now() + 3, execute: {

                alert.dismiss(animated: true, completion: nil)

                AppUser.shared().logout()

            })

        }

    }

    func socketManager(\_ socketManager: SocketManager, requestResultError: CommandType) {

        print(requestResultError)

    }

}

extension WirelessSettingViewController {

    func tableView(\_ tableView: UITableView, viewForFooterInSection section: Int) -> UIView? {

//        let view = UIView.init(frame: CGRect(x: 0, y: 0, width: kScreenWidth, height: 30+300))

//

//        view.addSubview(button)

//        button.snp.makeConstraints { (make) in

//            make.top.equalToSuperview()

//            make.height.equalTo(30)

//            make.width.equalTo(40)

//            make.right.equalTo(view).offset(-5)

//        }

        footerview.frame = CGRect(x: 0, y: 0, width: kScreenWidth, height: 300)

//        view.addSubview(footerview)

        return footerview

    }

    func tableView(\_ tableView: UITableView, heightForFooterInSection section: Int) -> CGFloat {

        return 300

    }

    func tableView(\_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {

        return self.dataSource.count

    }

    func tableView(\_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {

        let model = self.dataSource[indexPath.row]

        if model.cellType == .textfield {

            let cell = tableView.dequeueReusableCell(withIdentifier: "TextFieldTableViewCell", for: indexPath) as! TextFieldTableViewCell

            if !textFields.contains(cell.textField) {

                textFields.append(cell.textField)

            }

            NotificationCenter.default.addObserver(self, selector: #selector(textChange(\_:)), name: UITextField.textDidChangeNotification, object: cell.textField)

            cell.model(model)

            cell.textField.delegate = self

            if indexPath.row == 1 { pwdTextField = cell.textField }

            return cell

        } else if model.cellType == .password {

            let cell = tableView.dequeueReusableCell(withIdentifier: "WirelessPasswordCell", for: indexPath) as! WirelessPasswordCell

            if !textFields.contains(cell.textField) {

                textFields.append(cell.textField)

            }

            NotificationCenter.default.addObserver(self, selector: #selector(textChange(\_:)), name: UITextField.textDidChangeNotification, object: cell.textField)

            cell.model(model)

            cell.textField.delegate = self

            if indexPath.row == 1 { pwdTextField = cell.textField }

            return cell

        } else {

            let cell = tableView.dequeueReusableCell(withIdentifier: "cell", for: indexPath)

            return cell

        }

    }

    func tableView(\_ tableView: UITableView, heightForHeaderInSection section: Int) -> CGFloat {

        return 0.00001

    }

    func tableView(\_ tableView: UITableView, viewForHeaderInSection section: Int) -> UIView? {

        return UIView.init(frame: .zero)

    }

    func tableView(\_ tableView: UITableView, heightForRowAt indexPath: IndexPath) -> CGFloat {

        return 57

    }

}

extension WirelessSettingViewController: UITextFieldDelegate {

    func textField(\_ textField: UITextField, shouldChangeCharactersIn range: NSRange, replacementString string: String) -> Bool {

        if textField == pwdTextField {

            if string == "" { return true }

            let characterSet = NSCharacterSet(charactersIn: "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz1234567890,./;'[]-=!@#$%^&\*()\_+\\|<>?:\"{}").inverted

            let filtered = string.components(separatedBy: characterSet)

            let ss = filtered.joined(separator: "")

            return ss == string

        }

        return true

    }

}

extension WirelessSettingViewController {

    @objc func textChange(\_ notification: NSNotification) {

        hasEdited = true

        enableLogin()

    }

    func enableLogin() {

        let name = dataSource[0].textfieldText ?? ""

        if name.count > 0 {

            footerview.isEnable = true

        } else {

            footerview.isEnable = false

        }

    }

}

//

//  VisitorViewController.swift

//  JinXingNuoProject

//

//  Created by 刘俊威 on 2019/7/4.

//  Copyright © 2019 wadeLiu.com. All rights reserved.

//

import UIKit

class VisitorViewController: JLKFBaseViewController, UITableViewDelegate, UITableViewDataSource {

    // MARK: - properties

    lazy var tableView: UITableView = {

        let tableview = UITableView.init(frame: .zero, style: .grouped)

        tableview.delegate = self as UITableViewDelegate

        tableview.dataSource = self as UITableViewDataSource

        tableview.backgroundColor = HexNumber(r: 243, g: 245, b: 248, alpha: 1)

        tableview.separatorStyle = UITableViewCell.SeparatorStyle.none

        tableview.tableFooterView = UIView.init(frame: .zero)

        tableview.bounces = false

        tableview.register(UINib(nibName: "TextFieldTableViewCell", bundle: nil), forCellReuseIdentifier: "TextFieldTableViewCell")

        tableview.register(UINib(nibName: "NormalTableViewCell", bundle: nil), forCellReuseIdentifier: "NormalTableViewCell")

        tableview.register(UINib(nibName: "SwitchTableViewCell", bundle: nil), forCellReuseIdentifier: "SwitchTableViewCell")

        return tableview

    }()

    lazy var dataSource: [TextTableCellModel] = []

    lazy var footerview: TabelViewFooterView = {

        let view = TabelViewFooterView.init(frame: .zero)

        view.isEnable = true

        view.confirmBlock = {

            self.requestCommand()

        }

        return view

    }()

    var guestNetworkSetting: GuestNetworkSettings?

    var pwdTextField: UITextField?

    // MARK: - ViewController Life CirCle

    override func viewDidLoad() {

        super.viewDidLoad()

        initview()

        initdata()

        reloadData()

    }

    override func viewWillAppear(\_ animated: Bool) {

        super.viewWillAppear(animated)

        SocketManager.shared().delegate = self

    }

    override func viewWillDisappear(\_ animated: Bool) {

        super.viewWillDisappear(animated)

        SVProgressHUD.dismiss()

    }

    func reloadData() {

        CocoaAsyncSocketManager.shared().getInformationCommand(command: "RequestGuestNetworkSettings", tag: .RequestGuestNetworkSettings)

    }

    func requestCommand() {

        let on = dataSource[0].on

        let name = dataSource[1].textfieldText

        if name == nil || name == "" {

            let alert = UIAlertController.init(title: language("保存失败"), message: language("访客网络名称不能为空"), preferredStyle: .alert)

            self.present(alert, animated: true, completion: nil)

            DispatchQueue.main.asyncAfter(deadline: DispatchTime.now() + 2) {

                alert.dismiss(animated: true, completion: nil)

            }

            return

        }

        let pwd = dataSource[2].textfieldText ?? ""

        let time = numberDictinary[dataSource[3].detailTitle ?? "0"]

        if pwd.count < 8 {

            let alert = UIAlertController.init(title: language("密码格式错误"), message: language("需为八位数或以上"), preferredStyle: .alert)

            self.present(alert, animated: true, completion: nil)

            DispatchQueue.main.asyncAfter(deadline: DispatchTime.now() + 2) {

                alert.dismiss(animated: true, completion: nil)

            }

            return

        }

        if pwd.count > 32 {

            let alert = UIAlertController.init(title: language("密码格式错误"), message: language("需为三十二位数以下"), preferredStyle: .alert)

            self.present(alert, animated: true, completion: nil)

            DispatchQueue.main.asyncAfter(deadline: DispatchTime.now() + 2) {

                alert.dismiss(animated: true, completion: nil)

            }

            return

        }

        let timeModel = dataSource[3]

        if timeModel.detailTitle == nil || timeModel.detailTitle == "" {

            let alert = UIAlertController.init(title: language("未选择有效时间"), message: language(""), preferredStyle: .alert)

            self.present(alert, animated: true, completion: nil)

            DispatchQueue.main.asyncAfter(deadline: DispatchTime.now() + 2) {

                alert.dismiss(animated: true, completion: nil)

            }

            return

        }

        let params = ["GuestNetworkSettings":

            [

                [

                    "guestEnable":on ? "on" : "off",

                    "guestSsid":name ?? "",

                    "guestPwd":pwd,

                    "guestTimes":time

                ]

            ]

        ]

        SVProgressHUD.show()

        CocoaAsyncSocketManager.shared().sendSetCommand(setDictionary: params, tag: .RequestGuestNetworkSettings)

    }

    // MARK: - custom ui

    func initview() {

        self.title = language("访客模式")

        self.view.addSubview(self.tableView)

        self.tableView.snp.makeConstraints { (make) in

            make.left.right.bottom.equalTo(self.view)

            make.top.equalTo(self.navigationBarView.snp.bottom)

        }

    }

    // MARK: - data

    func initdata() {

        let titleArray = [language("访客网络"), language("访客网络名称"), language("访客网络密码"), language("有效时长")]

        let typeArray = [UITableViewCellType.mySwitch, UITableViewCellType.textfield, UITableViewCellType.textfield, UITableViewCellType.normal]

        dataSource.removeAll()

        for index in 0..<titleArray.count {

            let model = TextTableCellModel.init()

            model.title = titleArray[index]

            model.cellType = typeArray[index]

            dataSource.append(model)

        }

        tableView.reloadData()

    }

    let timtDictionary = ["0":language("一直有效"), "4":language("4小时"), "8":language("8小时")]

    let numberDictinary = [language("一直有效"):"0", language("4小时"):"4", language("8小时"):"8"]

}

extension VisitorViewController: SocketManagerProtocol {

    func socketManager(\_ socketManager: SocketManager, guestNetworkSettingsModel: GuestNetworkSettingsModel) {

        let guestNetworkSetting = guestNetworkSettingsModel.GuestNetworkSettings?.first

        self.guestNetworkSetting = guestNetworkSetting

        let switchModel = dataSource[0]

        if guestNetworkSetting?.guestEnable == "on" {

            switchModel.on = true

        } else if guestNetworkSetting?.guestEnable == "off" {

            switchModel.on = false

        }

        setVisitorEnable(switchModel.on)

        let nameModel = dataSource[1]

        nameModel.textfieldText = guestNetworkSetting?.guestSsid

        let pwdModel = dataSource[2]

        pwdModel.textfieldText = guestNetworkSetting?.guestPwd

        let timeModel = dataSource[3]

        if let guestTime = guestNetworkSetting?.guestTimes {

            timeModel.detailTitle = timtDictionary[guestTime]

        } else {

            timeModel.detailTitle = guestNetworkSetting?.guestTimes

        }

        tableView.reloadData()

    }

    func socketManager(\_ socketManager: SocketManager, requestResultSuccess: Bool) {

        SVProgressHUD.dismiss()

        let alert = UIAlertController.init(title: language("设置成功，软件即将退出"), message: "", preferredStyle: .alert)

        self.present(alert, animated: true, completion: nil)

        DispatchQueue.main.asyncAfter(deadline: DispatchTime.now() + 2) {

            alert.dismiss(animated: true, completion: nil)

            AppUser.shared().logout()

        }

    }

    func socketManager(\_ socketManager: SocketManager, requestResultError: CommandType) {

        SVProgressHUD.dismiss()

    }

}

extension VisitorViewController {

    func tableView(\_ tableView: UITableView, viewForFooterInSection section: Int) -> UIView? {

        self.footerview.frame = CGRect(x: 0, y: 0, width: kScreenWidth, height: 300)

        return self.footerview

    }

    func tableView(\_ tableView: UITableView, heightForFooterInSection section: Int) -> CGFloat {

        return 300

    }

    func tableView(\_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {

        return self.dataSource.count

    }

    func setVisitorEnable(\_ on: Bool) {

        let model = dataSource[0]

        model.on = on

        for tempModel in dataSource {

            tempModel.textfieldEnable = model.on

        }

    }

    func tableView(\_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {

        let model = self.dataSource[indexPath.row]

        if model.cellType == .normal {

            let cell = tableView.dequeueReusableCell(withIdentifier: "NormalTableViewCell", for: indexPath) as! NormalTableViewCell

            cell.model(model)

            return cell

        } else if model.cellType == .mySwitch {

            let cell = tableView.dequeueReusableCell(withIdentifier: "SwitchTableViewCell", for: indexPath) as! SwitchTableViewCell

            cell.model(model)

            cell.onBlock = {(on) in

                self.setVisitorEnable(on)

                tableView.reloadData()

            }

            return cell

        } else if model.cellType == .textfield {

            let cell = tableView.dequeueReusableCell(withIdentifier: "TextFieldTableViewCell", for: indexPath) as! TextFieldTableViewCell

            cell.model(model)

            cell.textField.delegate = self

            if indexPath.row == 2 { pwdTextField = cell.textField }

            cell.textFieldBolck = { (text) in

            }

            return cell

        } else {

            let cell = tableView.dequeueReusableCell(withIdentifier: "cell", for: indexPath)

            return cell

        }

    }

    func tableView(\_ tableView: UITableView, heightForHeaderInSection section: Int) -> CGFloat {

        return 0.001

    }

    func tableView(\_ tableView: UITableView, viewForHeaderInSection section: Int) -> UIView? {

        return UIView.init(frame: .zero)

    }

    func tableView(\_ tableView: UITableView, heightForRowAt indexPath: IndexPath) -> CGFloat {

        return 57

    }

    func tableView(\_ tableView: UITableView, didSelectRowAt indexPath: IndexPath) {

        if indexPath.row == 3 {

            /// 开关不打开 不可用

            let model = dataSource[0]

            if !model.textfieldEnable { return }

            let vc = EffectiveTimeViewController.init()

            if let guestTime = self.dataSource[indexPath.row].detailTitle {

                let time = numberDictinary[guestTime]

                vc.time = time

            }

            vc.callBlock = { (guestTime) in

                let timeModel = self.dataSource[indexPath.row]

                timeModel.detailTitle = self.timtDictionary[guestTime]

                self.tableView.reloadData()

            }

            self.navigationController?.pushViewController(vc, animated: true)

        }

    }

}

extension VisitorViewController: UITextFieldDelegate {

    func textField(\_ textField: UITextField, shouldChangeCharactersIn range: NSRange, replacementString string: String) -> Bool {

        if textField == pwdTextField {

            if string == "" { return true }

            let characterSet = NSCharacterSet(charactersIn: "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz1234567890,./;'[]-=!@#$%^&\*()\_+\\|<>?:\"{}").inverted

            let filtered = string.components(separatedBy: characterSet)

            let ss = filtered.joined(separator: "")

            return ss == string

        }

        return true

    }

}

extension SVProgressHUD {

    static public func showMask() {

        SVProgressHUD.setDefaultMaskType(.black)

        SVProgressHUD.show()

    }

    static public func dismissMask() {

        SVProgressHUD.dismiss()

        SVProgressHUD.setDefaultMaskType(.none)

    }

}

//

//  QOSViewController.swift

//  JinXingNuoProject

//

//  Created by 刘俊威 on 2019/7/4.

//  Copyright © 2019 wadeLiu.com. All rights reserved.

//

import UIKit

class QOSViewController: JLKFBaseViewController, UITableViewDelegate, UITableViewDataSource {

    // MARK: - properties

    lazy var tableView: UITableView = {

        let tableview = UITableView.init(frame: .zero, style: .grouped)

        tableview.delegate = self as UITableViewDelegate

        tableview.dataSource = self as UITableViewDataSource

        tableview.backgroundColor = HexNumber(r: 243, g: 245, b: 248, alpha: 1)

        tableview.separatorStyle = UITableViewCell.SeparatorStyle.none

        footerview.frame = CGRect(x: 0, y: 0, width: kScreenWidth, height: 300)

        tableview.tableFooterView = footerview

        tableview.bounces = false

        tableview.register(UINib(nibName: "QOSTableViewCell", bundle: nil), forCellReuseIdentifier: "QOSTableViewCell")

        tableview.register(UINib(nibName: "SelectTableViewCell", bundle: nil), forCellReuseIdentifier: "SelectTableViewCell")

        return tableview

    }()

    lazy var footerview: TabelViewFooterView = {

        let view = TabelViewFooterView.init(frame: .zero)

        view.isEnable = true

        view.confirmBlock = {

            if let currentModel = self.currentModel {

                let index = NSArray.init(array: self.topArray).index(of: currentModel)

                self.requestSentCommand(index)

            }

        }

        return view

    }()

    lazy var dataSource: [TextTableCellModel] = []

    lazy var topArray: [TextTableCellModel] = []

    private var currentModel: TextTableCellModel?

    private func selectModel(\_ model: TextTableCellModel) {

        self.currentModel?.isSelect = false

        self.currentModel = model

        self.currentModel?.isSelect = true

    }

    // MARK: - ViewController Life CirCle

    override func viewDidLoad() {

        super.viewDidLoad()

        initview()

        initdata()

        reloadData()

    }

    override func viewWillAppear(\_ animated: Bool) {

        super.viewWillAppear(animated)

        SocketManager.shared().delegate = self

    }

    override func viewWillDisappear(\_ animated: Bool) {

        super.viewWillDisappear(animated)

        SVProgressHUD.dismiss()

        SocketManager.shared().delegate = nil

    }

    func reloadData() {

        CocoaAsyncSocketManager.shared().getInformationCommand(command: "RequestQosSettings", tag: .RequestQosSettings)

    }

    func requestSentCommand(\_ index: Int) {

        var enable: String?

        switch index {

        case 0:

            enable = "off"

        case 1:

            enable = "on"

        case 2:

            enable = "auto"

        default:

            enable = "off"

        }

        let downloadSpeed = dataSource[0]

        let uploadSpeed = dataSource[1]

        let params = [

            "QosSettings":[

                [

                    "enable":enable,

                    "bandwidth\_downlink":downloadSpeed.textfieldText,

                    "bandwidth":uploadSpeed.textfieldText,

                ]

            ]

        ]

        SVProgressHUD.show()

        CocoaAsyncSocketManager.shared().sendSetCommand(setDictionary: params, tag: .NONE)

    }

    // MARK: - custom ui

    func initview() {

        self.title = language("智能QOS")

        self.view.addSubview(self.tableView)

        self.tableView.snp.makeConstraints { (make) in

            make.left.right.bottom.equalTo(self.view)

            make.top.equalTo(self.navigationBarView.snp.bottom)

        }

    }

    // MARK: - data

    func initdata() {

        // top

//        let topTitleArray = [language("关闭"), language("开启"), language("自动")]

        let topTitleArray = [language("关闭"), language("开启")]

        dataSource.removeAll()

        for index in 0..<topTitleArray.count {

            let model = TextTableCellModel.init()

            model.title = topTitleArray[index]

            model.cellType = .select

            topArray.append(model)

        }

        // buttom

        let titleArray = [language("下载速度"), language("上传速度")]

        let detailArray = ["", ""]

        dataSource.removeAll()

        for index in 0..<titleArray.count {

            let model = TextTableCellModel.init()

            model.title = titleArray[index]

            model.cellType = .textfield

            model.textfieldEnable = true

            model.textfieldKeyboardType = .numbersAndPunctuation

            model.textfieldText = detailArray[index]

            model.textfieldPlaceholder = language("点击设置")

            model.textfieldReturnType = .send

            dataSource.append(model)

        }

        tableView.reloadData()

    }

}

extension QOSViewController: SocketManagerProtocol {

    func socketManager(\_ socketManager: SocketManager, qosSettingsModel: QosSettingsModel) {

        let qosSetting = qosSettingsModel.QosSettings?.first

        if qosSetting?.enable == "on" {

            selectModel(topArray[1])

        } else if qosSetting?.enable == "off" {

            selectModel(topArray[0])

        } else if qosSetting?.enable == "auto" {

            selectModel(topArray[2])

        }

        let downloadSpeed = dataSource[0]

        downloadSpeed.textfieldText = qosSetting?.bandwidth\_downlink

        let uploadSpeed = dataSource[1]

        uploadSpeed.textfieldText = qosSetting?.bandwidth

        tableView.reloadData()

    }

    func socketManager(\_ socketManager: SocketManager, requestResultSuccess: Bool) {

        SVProgressHUD.dismiss()

        let alert = UIAlertController.init(title: language("操作成功"), message: "", preferredStyle: .alert)

        self.present(alert, animated: true, completion: nil)

        DispatchQueue.main.asyncAfter(deadline: DispatchTime.now() +  2) {

            alert.dismiss(animated: true, completion: nil)

        }

    }

    func socketManager(\_ socketManager: SocketManager, requestResultError: CommandType) {

        SVProgressHUD.dismiss()

    }

}

extension QOSViewController {

    func numberOfSections(in tableView: UITableView) -> Int {

        return 2

    }

    func tableView(\_ tableView: UITableView, viewForFooterInSection section: Int) -> UIView? {

        if section == 1 {

            let view = UIView.init()

            view.frame = CGRect(x: 0, y: 0, width: kScreenWidth, height: 50)

            let label = UILabel.init()

            label.font = UIFont.systemFont(ofSize: 16)

            label.text = language("开启后将限制设备的上传和下载速率")

            label.textAlignment = .center

            label.textColor = HexNumber(r: 0x9C, g: 0x9c, b: 0x9c, alpha: 1)

            label.numberOfLines = 0

            view.addSubview(label)

            label.snp.makeConstraints { (make) in

                make.left.right.bottom.equalTo(view)

            }

            return view

        }

        return UIView.init(frame: .zero)

    }

    func tableView(\_ tableView: UITableView, heightForFooterInSection section: Int) -> CGFloat {

        if section == 0 {

            return 10

        }

        return 50

    }

    func tableView(\_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {

        switch section {

        case 0:

            return self.topArray.count

        default:

            return self.dataSource.count

        }

    }

    func tableView(\_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {

        switch indexPath.section {

        case 0:

            let model = self.topArray[indexPath.row]

            let cell = tableView.dequeueReusableCell(withIdentifier: "SelectTableViewCell", for: indexPath) as! SelectTableViewCell

            cell.model(model)

            return cell

        default:

            let model = self.dataSource[indexPath.row]

            let cell = tableView.dequeueReusableCell(withIdentifier: "QOSTableViewCell", for: indexPath) as! QOSTableViewCell

            cell.model(model)

            cell.textFeild.delegate = self

            return cell

        }

    }

    func tableView(\_ tableView: UITableView, heightForHeaderInSection section: Int) -> CGFloat {

        return 0.001

    }

    func tableView(\_ tableView: UITableView, viewForHeaderInSection section: Int) -> UIView? {

        return UIView.init(frame: .zero)

    }

    func tableView(\_ tableView: UITableView, heightForRowAt indexPath: IndexPath) -> CGFloat {

        return 57

    }

    func tableView(\_ tableView: UITableView, didSelectRowAt indexPath: IndexPath) {

        if indexPath.section == 0 {

            selectModel(topArray[indexPath.row])

            tableView.reloadData()

        }

    }

}

extension QOSViewController: UITextFieldDelegate {

    func textField(\_ textField: UITextField, shouldChangeCharactersIn range: NSRange, replacementString string: String) -> Bool {

        if string == "" { return true }

        let characterSet = NSCharacterSet(charactersIn: "1234567890").inverted

        let filtered = string.components(separatedBy: characterSet)

        let ss = filtered.joined(separator: "")

        return ss == string

    }

    func textFieldShouldReturn(\_ textField: UITextField) -> Bool {

        view.endEditing(true)

        return true

    }

}

//

//  NetworkSettingViewController.swift

//  JinXingNuoProject

//

//  Created by 刘俊威 on 2019/7/4.

//  Copyright © 2019 wadeLiu.com. All rights reserved.

//

import UIKit

class NetworkSettingViewController: JLKFBaseViewController, UITableViewDelegate, UITableViewDataSource {

    enum WanType: String {

        case autoIp = "autoIp"

        case fixedIp = "fixedIp"

        case ppp = "ppp"

        case bridge = "bridge"

        case none = "none"

    }

    // MARK: - properties

    lazy var tableView: UITableView = {

        let tableview = UITableView.init(frame: .zero, style: .grouped)

        tableview.delegate = self as UITableViewDelegate

        tableview.dataSource = self as UITableViewDataSource

        tableview.backgroundColor = HexNumber(r: 243, g: 245, b: 248, alpha: 1)

        tableview.separatorStyle = UITableViewCell.SeparatorStyle.none

        tableview.bounces = false

        tableview.register(UINib(nibName: "TextFieldTableViewCell", bundle: nil), forCellReuseIdentifier: "TextFieldTableViewCell")

        tableview.register(UINib(nibName: "NormalTableViewCell", bundle: nil), forCellReuseIdentifier: "NormalTableViewCell")

        return tableview

    }()

    lazy var nodataView: UIView = {

        let view = UIView.init()

        view.frame = CGRect(x: 0, y: 0, width: kScreenWidth, height: 50)

        let label = UILabel.init()

        label.text = language("当前上网方式无需配置。")

        label.font = UIFont.systemFont(ofSize: 16)

        label.textColor = HexString(string: "353535")

        label.numberOfLines = 0

        label.textAlignment = .center

        view.addSubview(label)

        label.snp.makeConstraints({ (make) in

            make.center.equalToSuperview()

        })

        return view

    }()

    lazy var dataSource: [TextTableCellModel] = []

    lazy var topModel: TextTableCellModel = {

        let model = TextTableCellModel.init()

        model.cellType = .normal

        model.title = language("WAN接入类型")

        return model

    }()

    lazy var footerview: TabelViewFooterView = {

        let view = TabelViewFooterView.init(frame: .zero)

        view.isEnable = true

        view.confirmBlock = {

            let type = self.currentType.rawValue

            let pppUserName = type == "ppp" ? self.dataSource[0].textfieldText : ""

            let pppPassword = type == "ppp" ? self.dataSource[1].textfieldText ?? "" : ""

            let pppMtuSize = type == "ppp" ? self.dataSource[2].textfieldText : ""

            let wan\_ip = type == "fixedIp" ? self.dataSource[0].textfieldText : ""

            let wan\_mask = type == "fixedIp" ? self.dataSource[1].textfieldText : ""

            let wan\_gateway = type == "fixedIp" ? self.dataSource[2].textfieldText : ""

            let dns1 = type == "fixedIp" ? self.dataSource[3].textfieldText : ""

            let dns2 = type == "fixedIp" ? self.dataSource[4].textfieldText : ""

            if type == "ppp" {

                if pppUserName?.count == 0 {

                    let alert = UIAlertController.init(title: language("请输入账号"), message: language(""), preferredStyle: .alert)

                    self.present(alert, animated: true, completion: nil)

                    DispatchQueue.main.asyncAfter(deadline: DispatchTime.now() + 2) {

                        alert.dismiss(animated: true, completion: nil)

                    }

                    return

                }

                if pppPassword.count < 8 {

                    let alert = UIAlertController.init(title: language("密码格式错误"), message: language("需为八位数或以上"), preferredStyle: .alert)

                    self.present(alert, animated: true, completion: nil)

                    DispatchQueue.main.asyncAfter(deadline: DispatchTime.now() + 2) {

                        alert.dismiss(animated: true, completion: nil)

                    }

                    return

                }

                if pppPassword.count > 32 {

                    let alert = UIAlertController.init(title: language("密码格式错误"), message: language("需为三十二位数以下"), preferredStyle: .alert)

                    self.present(alert, animated: true, completion: nil)

                    DispatchQueue.main.asyncAfter(deadline: DispatchTime.now() + 2) {

                        alert.dismiss(animated: true, completion: nil)

                    }

                    return

                }

            }

            let params = [

                "NetworkSettings":[

                    [

                        "wanType":type,

                        "pppUserName": pppUserName,

                        "pppPassword": pppPassword,

                        "pppMtuSize": pppMtuSize,

                        "wan\_ip": wan\_ip,

                        "wan\_mask": wan\_mask,

                        "wan\_gateway": wan\_gateway,

                        "dns1": dns1,

                        "dns2": dns2,

                    ]

                ]

            ]

            SVProgressHUD.show()

            CocoaAsyncSocketManager.shared().sendSetCommand(setDictionary: params, tag: .NONE)

        }

        return view

    }()

    var currentType: WanType = .none

    var pwdTextField: UITextField?// 保存宽带密码输入框

    let nameDictionary = ["ppp":language("PPPoE账号"),

                          "autoIp":language("动态IP"),

                          "fixedIp":language("静态IP"),

                          "bridge":language("桥模式")]

    // MARK: - ViewController Life CirCle

    override func viewDidLoad() {

        super.viewDidLoad()

        initview()

        initdata()

        reloadData()

    }

    override func viewWillAppear(\_ animated: Bool) {

        super.viewWillAppear(animated)

        SocketManager.shared().delegate = self

    }

    override func viewWillDisappear(\_ animated: Bool) {

        super.viewWillDisappear(animated)

        SVProgressHUD.dismiss()

        SocketManager.shared().delegate = nil

    }

    func reloadData() {

        CocoaAsyncSocketManager.shared().getInformationCommand(command: "RequestNetworkSettings", tag: .RequestNetworkSettings)

    }

    // MARK: - custom ui

    func initview() {

        self.title = language("上网设置")

        self.view.addSubview(self.tableView)

        self.tableView.snp.makeConstraints { (make) in

            make.left.right.bottom.equalTo(self.view)

            make.top.equalTo(self.navigationBarView.snp.bottom)

        }

    }

    var networkSetting: NetworkSettings?

    let fixedIpTitles = [language("IP地址"), language("子网掩码"), language("默认网关"), language("首选DNS服务器"), language("备选DNS服务器(可不填)")]

    let pppTitles = [language("宽带账号"), language("宽带密码"), language("MTU")]

    // MARK: - data

    func formatModel() {

        for model in dataSource {

            switch model.title {

            case fixedIpTitles[0]:

                model.textfieldText = networkSetting?.wan\_ip

            case fixedIpTitles[1]:

                model.textfieldText = networkSetting?.wan\_mask

            case fixedIpTitles[2]:

                model.textfieldText = networkSetting?.wan\_gateway

            case fixedIpTitles[3]:

                model.textfieldText = networkSetting?.dns1

            case fixedIpTitles[4]:

                model.textfieldText = networkSetting?.dns2

            case pppTitles[0]:

                model.textfieldText = networkSetting?.pppUserName

            case pppTitles[1]:

                model.textfieldText = networkSetting?.pppPassword

            case pppTitles[2]:

                model.textfieldText = networkSetting?.pppMtuSize

                model.textfieldKeyboardType = .numberPad

            default:

                print("未知")

            }

        }

    }

    func initdata() {

        var titleArray = [String]()

        switch currentType {

        case .fixedIp:

            titleArray = fixedIpTitles

            nodataView.isHidden = true

        case .ppp:

            titleArray = pppTitles

            nodataView.isHidden = true

        default:

            titleArray = []

            nodataView.isHidden = false

        }

        dataSource.removeAll()

        for index in 0..<titleArray.count {

            let model = TextTableCellModel.init()

            model.title = titleArray[index]

            model.cellType = .textfield

            model.textfieldEnable = true

            model.textfieldPlaceholder = language("点击设置")

            dataSource.append(model)

        }

        formatModel()

        tableView.reloadData()

    }

}

extension NetworkSettingViewController: SocketManagerProtocol {

    func socketManager(\_ socketManager: SocketManager, networkSettingsModel: NetworkSettingsModel) {

        let networkSetting = networkSettingsModel.NetworkSettings?.first

        self.networkSetting = networkSetting

        currentType = WanType.init(rawValue: networkSetting?.wanType ?? "none") ?? .none

        if let wanType = networkSetting?.wanType {

            topModel.detailTitle = nameDictionary[wanType]

        }

        initdata()

    }

    func socketManager(\_ socketManager: SocketManager, requestResultError: CommandType) {

        SVProgressHUD.dismiss()

    }

    func socketManager(\_ socketManager: SocketManager, requestResultSuccess: Bool) {

        SVProgressHUD.dismiss()

        let alert = UIAlertController.init(title: language("设置成功，软件即将退出"), message: "", preferredStyle: .alert)

        self.present(alert, animated: true, completion: nil)

        DispatchQueue.main.asyncAfter(deadline: DispatchTime.now() + 2) {

            alert.dismiss(animated: true, completion: nil)

            AppUser.shared().logout()

        }

    }

}

extension NetworkSettingViewController {

    func numberOfSections(in tableView: UITableView) -> Int {

        return 2

    }

    func tableView(\_ tableView: UITableView, viewForFooterInSection section: Int) -> UIView? {

        if section == 0 {

            return UIView.init(frame: .zero)

        }

        let bgview = UIView.init(frame: CGRect(x: 0, y: 0, width: kScreenWidth, height: 100+300))

        bgview.addSubview(self.nodataView)

        self.footerview.frame = CGRect(x: 0, y: 100, width: kScreenWidth, height: 300)

        bgview.addSubview(self.footerview)

        return bgview

    }

    func tableView(\_ tableView: UITableView, heightForFooterInSection section: Int) -> CGFloat {

        if section == 0 {

            return 10

        }

        return 400

    }

    func tableView(\_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {

        switch section {

        case 0:

            return 1

        default:

            return self.dataSource.count

        }

    }

    func tableView(\_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {

        switch indexPath.section {

        case 0:

            let cell = tableView.dequeueReusableCell(withIdentifier: "NormalTableViewCell", for: indexPath) as! NormalTableViewCell

            cell.model(self.topModel)

            return cell

        default:

            let model = self.dataSource[indexPath.row]

            let cell = tableView.dequeueReusableCell(withIdentifier: "TextFieldTableViewCell", for: indexPath) as! TextFieldTableViewCell

            cell.model(model)

            cell.textField.delegate = self

            if dataSource[indexPath.row].title == language("宽带密码") {

                pwdTextField = cell.textField

            }

            return cell

        }

    }

    func tableView(\_ tableView: UITableView, heightForHeaderInSection section: Int) -> CGFloat {

        return 0.001

    }

    func tableView(\_ tableView: UITableView, viewForHeaderInSection section: Int) -> UIView? {

        return UIView.init(frame: .zero)

    }

    func tableView(\_ tableView: UITableView, heightForRowAt indexPath: IndexPath) -> CGFloat {

        return 57

    }

    func tableView(\_ tableView: UITableView, didSelectRowAt indexPath: IndexPath) {

        if indexPath.section == 0 {

            let vc = NetworkSubViewController.init()

            switch currentType {

            case .autoIp:

                vc.currentIndex = 0

            case .fixedIp:

                vc.currentIndex = 1

            case .ppp:

                vc.currentIndex = 2

            case .bridge:

                vc.currentIndex = 3

            case .none:

                return

            }

            vc.callBlock = { (index) in

                switch index {

                case 0:

                    self.currentType = .autoIp

                case 1:

                    self.currentType = .fixedIp

                case 2:

                    self.currentType = .ppp

                case 3:

                    self.currentType = .bridge

                default:

                    print("")

                }

                self.topModel.detailTitle = self.nameDictionary[self.currentType.rawValue]

                self.initdata()

            }

            self.navigationController?.pushViewController(vc, animated: true)

        }

    }

}

extension NetworkSettingViewController: UITextFieldDelegate {

    func textField(\_ textField: UITextField, shouldChangeCharactersIn range: NSRange, replacementString string: String) -> Bool {

        if textField == pwdTextField {

            if string == "" { return true }

            let characterSet = NSCharacterSet(charactersIn: "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz1234567890,./;'[]-=!@#$%^&\*()\_+\\|<>?:\"{}").inverted

            let filtered = string.components(separatedBy: characterSet)

            let ss = filtered.joined(separator: "")

            return ss == string

        }

        return true

    }

}

//

//  NetworkSubViewController.swift

//  JinXingNuoProject

//

//  Created by 刘俊威 on 2019/7/4.

//  Copyright © 2019 wadeLiu.com. All rights reserved.

//

import UIKit

class NetworkSubViewController: JLKFBaseViewController, UITableViewDelegate, UITableViewDataSource {

    // MARK: - properties

    lazy var tableView: UITableView = {

        let tableview = UITableView.init(frame: .zero, style: .grouped)

        tableview.delegate = self as UITableViewDelegate

        tableview.dataSource = self as UITableViewDataSource

        tableview.backgroundColor = HexNumber(r: 243, g: 245, b: 248, alpha: 1)

        tableview.separatorStyle = UITableViewCell.SeparatorStyle.none

        tableview.tableFooterView = UIView.init(frame: .zero)

        tableview.bounces = false

        tableview.register(UINib(nibName: "SelectTableViewCell", bundle: nil), forCellReuseIdentifier: "SelectTableViewCell")

        return tableview

    }()

    lazy var dataSource: [TextTableCellModel] = []

    lazy var footerview: TabelViewFooterView = {

        let view = TabelViewFooterView.init(frame: .zero)

        view.confirmBlock = {

            self.requestCommand()

        }

        return view

    }()

    var currentIndex: Int?

    var currentModel: TextTableCellModel?

    // MARK: - ViewController Life CirCle

    override func viewDidLoad() {

        super.viewDidLoad()

        initview()

        initdata()

    }

    override func viewDidAppear(\_ animated: Bool) {

        super.viewDidAppear(animated)

        SocketManager.shared().delegate = self

    }

    override func viewWillDisappear(\_ animated: Bool) {

        super.viewWillDisappear(animated)

        SocketManager.shared().delegate = nil

    }

    public var callBlock: ((\_ index: Int) -> Void)?

    func requestCommand() {

        if let block = callBlock {

            if let model = currentModel {

                let index = NSArray.init(array: dataSource).index(of: model)

                block(index)

                navigationController?.popViewController(animated: true)

            }

        }

    }

    // MARK: - custom ui

    func initview() {

        self.title = language("WAN接入类型")

        self.view.addSubview(self.tableView)

        self.tableView.snp.makeConstraints { (make) in

            make.left.right.bottom.equalTo(self.view)

            make.top.equalTo(self.navigationBarView.snp.bottom)

        }

    }

    private func selectModel(model: TextTableCellModel) {

        self.currentModel?.isSelect = false

        self.currentModel = model

        self.currentModel?.isSelect = true

    }

    // MARK: - data

    func initdata() {

        let titleArray = [language("动态IP"), language("静态IP"), language("PPPoE账号")]

        dataSource.removeAll()

        for index in 0..<titleArray.count {

            let model = TextTableCellModel.init()

            model.title = titleArray[index]

            model.cellType = .select

            dataSource.append(model)

        }

        selectModel(model: dataSource[currentIndex ?? 0])

        tableView.reloadData()

    }

}

extension NetworkSubViewController: SocketManagerProtocol {

    func socketManager(\_ socketManager: SocketManager, requestResultSuccess: Bool) {

    }

    func socketManager(\_ socketManager: SocketManager, requestResultError: CommandType) {

    }

}

extension NetworkSubViewController {

    func numberOfSections(in tableView: UITableView) -> Int {

        return 1

    }

    func tableView(\_ tableView: UITableView, viewForFooterInSection section: Int) -> UIView? {

        self.footerview.frame = CGRect(x: 0, y: 0, width: kScreenWidth, height: 300)

        return self.footerview

    }

    func tableView(\_ tableView: UITableView, heightForFooterInSection section: Int) -> CGFloat {

        return 300

    }

    func tableView(\_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {

        return self.dataSource.count

    }

    func tableView(\_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {

        let model = self.dataSource[indexPath.row]

        let cell = tableView.dequeueReusableCell(withIdentifier: "SelectTableViewCell", for: indexPath) as! SelectTableViewCell

        cell.model(model)

        return cell

    }

    func tableView(\_ tableView: UITableView, heightForHeaderInSection section: Int) -> CGFloat {

        return 0.001

    }

    func tableView(\_ tableView: UITableView, viewForHeaderInSection section: Int) -> UIView? {

        return UIView.init(frame: .zero)

    }

    func tableView(\_ tableView: UITableView, heightForRowAt indexPath: IndexPath) -> CGFloat {

        return 57

    }

    func tableView(\_ tableView: UITableView, didSelectRowAt indexPath: IndexPath) {

        let model = dataSource[indexPath.row]

        selectModel(model: model)

        tableView.reloadData()

        footerview.isEnable = true

    }

}

//

//  SettingViewController.swift

//  JinXingNuoProject

//

//  Created by 刘俊威 on 2019/7/2.

//  Copyright © 2019 wadeLiu.com. All rights reserved.

//

import Foundation

import Alamofire

class SettingViewController: JLKFBaseViewController {

    // MARK: - properties

    lazy var tableView: UITableView = {

        let tableview = UITableView.init(frame: .zero, style: .grouped)

        tableview.delegate = self as UITableViewDelegate

        tableview.dataSource = self as UITableViewDataSource

        tableview.backgroundColor = HexNumber(r: 243, g: 245, b: 248, alpha: 1)

        tableview.separatorStyle = UITableViewCell.SeparatorStyle.none

        tableview.tableFooterView = UIView.init(frame: .zero)

        tableview.showsHorizontalScrollIndicator = false

        tableview.register(UINib(nibName: "NormalTableViewCell", bundle: nil), forCellReuseIdentifier: "NormalTableViewCell")

        tableview.register(UINib(nibName: "SwitchTableViewCell", bundle: nil), forCellReuseIdentifier: "SwitchTableViewCell")

        return tableview

    }()

    lazy var dataSource: [TextTableCellModel] = []

    var newVersionModel: MainData?

    // MARK: - ViewController Life CirCle

    override func viewDidLoad() {

        super.viewDidLoad()

        initview()

        initdata()

    }

    override func viewWillAppear(\_ animated: Bool) {

        super.viewWillAppear(animated)

        SocketManager.shared().delegate = self

        // 每次进入都检测当前版本

        requestData()

//        reloadData()

    }

    // MARK: - Http Request

    func reloadData() {

//        var model = MainData.init()

//        model.binurl = "http://www.baidu.com"

//        model.content = "测试更新内容lkasjfklasjflkasdjfkdsjfksadjfklasdjfklsdjklsdjfklsjfsnbvjbiuawvoasnvksnvkasvhoasasioasoinasonokasdflkasjflkasjfklasj"

//        model.name = "banbenneirong"

//        model.version = "v3.4.10"

//        model.fid = "fid"

//        model.createtime = "createtime"

//        model.ssid = "ssid"

//        newVersionModel = model

        CocoaAsyncSocketManager.shared().getInformationCommand(command: "RequestUpgradeSettings", tag: .RequestUpgradeSettings)

    }

    private func requestData() {

        if let url = URL.init(string: "http://120.79.61.154/firmware/latest") {

            AF.request(url, method: .get).responseJSON { (response) in

                switch response.result {

                case .success(let json):

                    if let res = json as? [String: Any]{

                        if let code = res["code"] as? Int {

                            if code == 200 {

                                // 成功

                                if let data = res["data"] as? [String : Any] {

                                    var model = MainData.init()

                                    model.binurl = data["binurl"] as? String

                                    model.content = data["content"] as? String

                                    model.name = data["name"] as? String

                                    model.version = data["version"] as? String

                                    model.fid = data["fid"] as? String

                                    model.createtime = data["createtime"] as? String

                                    model.ssid = data["ssid"] as? String

                                    model.downurl = data["downurl"] as? String

                                    self.newVersionModel = model

                                    self.reloadData()

                                }

                            }

                        }

                    }

                case .failure(let error):

                    print(error)

                    let versionDiscovery = self.dataSource[7]

                    versionDiscovery.detailTitle = language("")

                    self.tableView.reloadData()

                }

            }

        }

    }

    // MARK: - custom ui

    func initview() {

        self.leftButton.isHidden = true

        self.view.addSubview(self.tableView)

        self.tableView.snp.makeConstraints { (make) in

            make.left.right.bottom.equalTo(self.view)

            make.top.equalTo(self.navigationBarView.snp.bottom)

        }

    }

    // MARK: - data

    func initdata() {

        let titleArray = [language("无线设置"), language("访客网络"), language("家长控制"), language("上网设置"), language("智能QOS"), language("新增节点"), language("防火墙"), language("软件更新"), language("信号环境"), language("系统管理")]

        dataSource.removeAll()

        for index in 0..<titleArray.count {

            let model = TextTableCellModel.init()

            model.title = titleArray[index]

            if model.title == language("软件更新") {

                model.detailTitle = language("发现新版本")

            } else {

                model.detailTitle = ""

            }

            dataSource.append(model)

        }

        tableView.reloadData()

    }

}

extension SettingViewController: SocketManagerProtocol {

    func socketManager(\_ socketManager: SocketManager, upgradeSettingsModel: UpgradeSettingsModel) {

        let upgradeSetting = upgradeSettingsModel.UpgradeSettings?.first

        // 是否显示发现新版本

        var isShowNewVersion = false

        // 主节点新版本对比

        if let version = upgradeSetting?.mainDevVersion {

            if version != newVersionModel?.version {

                isShowNewVersion = true

            }

        }

        // 子节点新版本对比

        if isShowNewVersion == false {

            if let childs = upgradeSetting?.childs {

                loop: for child in childs {

                    if let version = child.childLocalVersion {

                        if version != newVersionModel?.version {

                            // 显示新版本对比，直接退出此循环

                            isShowNewVersion = true

                            break loop

                        }

                    }

                }

            }

        }

        let versionDiscovery = dataSource[7]

        if isShowNewVersion {

            versionDiscovery.detailTitle = language("发现新版本")

        } else {

            versionDiscovery.detailTitle = ""

        }

        tableView.reloadData()

    }

    func socketManager(\_ socketManager: SocketManager, requestResultSuccess: Bool) {

        let alert = UIAlertController.init(title: language("添加成功"), message: "", preferredStyle: .alert)

        self.present(alert, animated: true) {

            DispatchQueue.main.asyncAfter(deadline: DispatchTime.now() + 1.5, execute: {

                alert.dismiss(animated: true, completion: nil)

            })

        }

    }

    func socketManager(\_ socketManager: SocketManager, requestResultError: CommandType) {

        let alert = UIAlertController.init(title: language("添加失败"), message: "", preferredStyle: .alert)

        self.present(alert, animated: true) {

            DispatchQueue.main.asyncAfter(deadline: DispatchTime.now() + 1.5, execute: {

                alert.dismiss(animated: true, completion: nil)

            })

        }

    }

}

extension SettingViewController: UITableViewDelegate, UITableViewDataSource {

    func tableView(\_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {

        return self.dataSource.count

    }

    func tableView(\_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {

        let model = self.dataSource[indexPath.row]

        if model.cellType == .normal {

            let cell = tableView.dequeueReusableCell(withIdentifier: "NormalTableViewCell", for: indexPath) as! NormalTableViewCell

            cell.model(model)

            return cell

        } else if model.cellType == .mySwitch {

            let cell = tableView.dequeueReusableCell(withIdentifier: "SwitchTableViewCell", for: indexPath) as! SwitchTableViewCell

            cell.model(model)

            return cell

        } else {

            let cell = tableView.dequeueReusableCell(withIdentifier: "cell", for: indexPath)

            return cell

        }

    }

    func tableView(\_ tableView: UITableView, heightForHeaderInSection section: Int) -> CGFloat {

        return 0.001

    }

    func tableView(\_ tableView: UITableView, viewForHeaderInSection section: Int) -> UIView? {

        return UIView.init(frame: .zero)

    }

    func tableView(\_ tableView: UITableView, heightForRowAt indexPath: IndexPath) -> CGFloat {

        return 57

    }

    func tableView(\_ tableView: UITableView, didSelectRowAt indexPath: IndexPath) {

        switch indexPath.row {

        case 0:

            let vc = WirelessSettingViewController.init()

            self.navigationController?.pushViewController(vc, animated: true)

        case 1:

            let vc = VisitorViewController.init()

            self.navigationController?.pushViewController(vc, animated: true)

        case 2:

            let vc = ParentsControlViewController.init()

            self.navigationController?.pushViewController(vc, animated: true)

        case 3:

            let vc = NetworkSettingViewController.init()

            self.navigationController?.pushViewController(vc, animated: true)

        case 4:

            let vc = QOSViewController.init()

            self.navigationController?.pushViewController(vc, animated: true)

        case 5:

            SelectViewController.showAlert(scan: {

                let vc = WQCodeScanner.init()

                self.present(vc, animated: true, completion: nil)

                vc.resultBlock = { (string) in

                    if let text = string {

                        let params = [

                            "AddsnSettings":[

                                ["addSn":text]

                            ]

                        ]

                        CocoaAsyncSocketManager.shared().sendSetCommand(setDictionary: params, tag: .NONE)

                    }

                }

            }, sn: {

                SNAddViewController.showAlert(cancel: nil, confirm: { (text) in

                    let params = [

                        "AddsnSettings":[

                            ["addSn":text]

                        ]

                    ]

                    CocoaAsyncSocketManager.shared().sendSetCommand(setDictionary: params, tag: .NONE)

                })

            })

        case 6:

            let vc = FirewallViewController.init()

            self.navigationController?.pushViewController(vc, animated: true)

        case 7:

            let vc = UpdateViewController.init()

            vc.newVersionModel = newVersionModel

            self.navigationController?.pushViewController(vc, animated: true)

        case 8:

            let vc = SignalViewController.init()

            self.navigationController?.pushViewController(vc, animated: true)

        case 9:

            let vc = SystemManagerViewController.init()

            self.navigationController?.pushViewController(vc, animated: true)

        default:

            print("无页面")

        }

    }

}

//

//  HomeViewController.swift

//  JinXingNuoProject

//

//  Created by 刘俊威 on 2019/7/2.

//  Copyright © 2019 wadeLiu.com. All rights reserved.

//

import Foundation

class HomeViewController: JLKFBaseViewController {

    // MARK: - Properties

    lazy private var titleLabel: UILabel = {

        let view = UILabel.init()

        view.font = UIFont.boldSystemFont(ofSize: 20)

        view.textColor = UIColor.white

        view.text = ""

        return view

    }()

    lazy private var topMainView: UIView = {

        let view = UIView.init()

        view.layer.cornerRadius = 5

        view.backgroundColor = UIColor.white

        view.layer.shadowColor = UIColor.lightGray.cgColor

        view.layer.shadowRadius = 5

        view.layer.shadowOpacity = 0.5

        view.layer.shadowOffset = CGSize(width: 0, height: 2)

        return view

    }()

    lazy private var leftLabel: UILabel = {

        let view = UILabel.init()

        view.font = UIFont.boldSystemFont(ofSize: 15)

        view.textColor = UIColor.black

        view.text = ""

        return view

    }()

    lazy private var rightLabel: UILabel = {

        let view = UILabel.init()

        view.font = UIFont.boldSystemFont(ofSize: 15)

        view.textColor = UIColor.black

        view.text = ""

        return view

    }()

    let leftUnit = UILabel.init()

    let rightUnit = UILabel.init()

    lazy private var middleView: UIView = {

        let view = UIView.init()

        view.layer.cornerRadius = 5

        view.backgroundColor = UIColor.white

        view.layer.shadowColor = UIColor.lightGray.cgColor

        view.layer.shadowRadius = 5

        view.layer.shadowOpacity = 0.5

        view.layer.shadowOffset = CGSize(width: 0, height: 2)

        let line = UIView.init()

        line.backgroundColor = HexNumber(r: 0xf7, g: 0xf8, b: 0xfa, alpha: 1)

        view.addSubview(line)

        line.snp.makeConstraints({ (make) in

            make.centerX.centerY.equalTo(view)

            make.height.equalTo(25)

            make.width.equalTo(1)

        })

        let leftIamgeView = UIImageView.init()

        leftIamgeView.image = UIImage(named: "home\_icon\_upload")

        view.addSubview(leftIamgeView)

        leftIamgeView.snp.makeConstraints({ (make) in

            make.centerY.equalToSuperview()

            make.left.equalTo(10)

        })

        let leftTitleLabel = UILabel.init()

        leftTitleLabel.text = language("上传")

        leftTitleLabel.font = UIFont.systemFont(ofSize: 15)

        view.addSubview(leftTitleLabel)

        leftTitleLabel.snp.makeConstraints({ (make) in

            make.centerY.equalToSuperview()

            make.left.equalTo(leftIamgeView.snp.right).offset(5)

        })

        leftUnit.textColor = HexNumber(r: 0xb3, g: 0xb3, b: 0xb3, alpha: 1)

        leftUnit.font = UIFont.systemFont(ofSize: 10)

        view.addSubview(leftUnit)

        leftUnit.snp.makeConstraints({ (make) in

            make.centerY.equalToSuperview().offset(2)

            make.right.equalTo(line).offset(-10)

        })

        view.addSubview(self.leftLabel)

        self.leftLabel.snp.makeConstraints({ (make) in

            make.centerY.equalToSuperview()

            make.right.equalTo(leftUnit.snp\_left).offset(-5)

        })

        let rightIamgeView = UIImageView.init()

        rightIamgeView.image = UIImage(named: "home\_icon\_download")

        view.addSubview(rightIamgeView)

        rightIamgeView.snp.makeConstraints({ (make) in

            make.centerY.equalToSuperview()

            make.left.equalTo(line).offset(10)

        })

        let rightTitleLabel = UILabel.init()

        rightTitleLabel.text = language("下载")

        rightTitleLabel.font = UIFont.systemFont(ofSize: 15)

        view.addSubview(rightTitleLabel)

        rightTitleLabel.snp.makeConstraints({ (make) in

            make.centerY.equalToSuperview()

            make.left.equalTo(rightIamgeView.snp.right).offset(5)

        })

        rightUnit.textColor = HexNumber(r: 0xb3, g: 0xb3, b: 0xb3, alpha: 1)

        rightUnit.font = UIFont.systemFont(ofSize: 10)

        view.addSubview(rightUnit)

        rightUnit.snp.makeConstraints({ (make) in

            make.centerY.equalToSuperview().offset(2)

            make.right.equalTo(view).offset(-10)

        })

        view.addSubview(self.rightLabel)

        self.rightLabel.snp.makeConstraints({ (make) in

            make.centerY.equalToSuperview()

            make.right.equalTo(rightUnit.snp\_left).offset(-5)

        })

        return view

    }()

    var bottomTitleLabel: UILabel?

    lazy private var bottomView: UIButton = {

        let view = UIButton.init(type: .custom)

        view.layer.cornerRadius = 20

        view.backgroundColor = HexNumber(r: 0xf3, g: 0xc3, b: 0x39, alpha: 1)

        view.addTarget(self, action: #selector(bottomButtonAction(sender:)), for: .touchUpInside)

        let leftIamgeView = UIImageView.init()

        leftIamgeView.image = UIImage(named: "nav\_icon\_terminal")

        view.addSubview(leftIamgeView)

        leftIamgeView.snp.makeConstraints({ (make) in

            make.centerY.equalToSuperview()

            make.left.equalTo(25).priority(.high)

        })

        let leftTitleLabel = UILabel.init()

        leftTitleLabel.text = ""

        leftTitleLabel.font = UIFont.systemFont(ofSize: 15)

        leftTitleLabel.textColor = UIColor.white

        view.addSubview(leftTitleLabel)

        leftTitleLabel.snp.makeConstraints({ (make) in

            make.centerY.equalToSuperview()

            make.left.equalTo(leftIamgeView.snp.right).offset(10).priority(.high)

            make.right.equalTo(view.snp.right).offset(-25)

        })

        self.bottomTitleLabel = leftTitleLabel;

        return view

    }()

    lazy private var bgScrollView: UIScrollView = {

        let view = UIScrollView.init()

        view.contentSize = CGSize(width: kScreenWidth, height: 700)

        view.showsVerticalScrollIndicator = false

        view.showsHorizontalScrollIndicator = false

        view.contentInset = UIEdgeInsets(top: -20, left: 0, bottom: 0, right: 0)

        return view

    }()

    /// 顶部view的宽度

    let topMainViewWidth = (kScreenWidth-(15\*2))

    /// 顶部view的高度

    let topMainViewHeight: CGFloat = 360

    /// 保存gifview

    var gifViews: [MainGifView] = []

    var gifViewLoop: [Int : MainGifView] = [:]

    var socketManager = SocketManager.shared()

    // MARK: - ViewController Life CirCle

    override func viewDidLoad() {

        super.viewDidLoad()

        initView()

    }

    override func viewWillAppear(\_ animated: Bool) {

        super.viewWillAppear(animated)

        socketManager.delegate = self

        socketManager.startTimer()

//        let mainModel = HomeModel()

//        mainModel.name = "主页面"

//        mainModel.number = "1"

//        mainModel.linkStatus = "0"

//        mainModel.isChildNobe = false

//        mainModel.isOnline = true

//        dataSource.append(mainModel)

//        otherInitGifView(dataSource)

    }

    override func viewDidDisappear(\_ animated: Bool) {

        super.viewDidDisappear(animated)

        socketManager.stopTimer()

    }

    // MARK: - custom ui

    func initView() {

        view.backgroundColor = UIColor.white

        leftButton.isHidden = true

        navigationBarView.backgroundColor = .clear

        view.addSubview(bgScrollView)

        bgScrollView.snp.makeConstraints { (make) in

            make.top.equalTo(self.navigationBarView.snp.bottom)

            make.bottom.left.right.equalToSuperview()

        }

        let topBgImageView = UIImageView.init(image: UIImage(named: "home\_bg"))

        navigationBarView.addSubview(topBgImageView)

        navigationBarView.sendSubviewToBack(topBgImageView)

        topBgImageView.snp.makeConstraints { (make) in

            make.top.equalToSuperview()

            make.left.right.equalToSuperview()

        }

        self.bgScrollView.addSubview(self.topMainView)

        self.topMainView.snp.makeConstraints { (make) in

            make.top.equalTo(self.bgScrollView.snp.top).offset(self.topHeight-20)

            make.left.equalTo(self.view).offset(15)

            make.right.equalTo(self.view).offset(-15)

            make.height.equalTo(self.topMainViewHeight)

        }

        self.bgScrollView.addSubview(self.middleView)

        self.middleView.snp.makeConstraints { (make) in

            make.top.equalTo(self.topMainView.snp.bottom).offset(20)

            make.left.equalTo(self.view).offset(15)

            make.right.equalTo(self.view).offset(-15)

            make.height.equalTo(66)

        }

        self.bgScrollView.addSubview(self.bottomView)

        self.bottomView.snp.makeConstraints { (make) in

            make.top.equalTo(self.middleView.snp.bottom).offset(40)

            make.centerX.equalToSuperview()

            make.width.equalTo(180).priority(.low)

            make.height.equalTo(40)

        }

    }

    // MARK: - button action

    @objc func bottomButtonAction(sender: UIButton) {

        let vc = ConnectViewController.init()

        self.navigationController?.pushViewController(vc, animated: true)

//        let model = HomeModel()

//        model.name = "子节点"

//        model.number = "2"

//        model.mac = "jddjdj"

//

//        model.isChildNobe = true

//        model.isOnline = true

//        dataSource.append(model)

//        otherInitGifView(dataSource)

    }

    var dataSource: Array<HomeModel> = []

}

extension HomeViewController: SocketManagerProtocol {

    func socketManager(\_ socketManager: SocketManager, homeSettingsModel: HomeSettingsModel) {

        let homeSettings = homeSettingsModel.HomeSettings?.first

        setUpAndDownSpeed(homeSettings: homeSettings)

        setSubDevice(homeSettings: homeSettings)

        setMainSsid(homeSettings?.mainDevSsid)

        setConnectedDevice(homeSettings?.mainDevSum)

        print("代理正在运行")

    }

}

// MARK: - set view data

extension HomeViewController {

    func setUpAndDownSpeed(homeSettings: HomeSettings?) {

        let wanUpSpeed = (homeSettings?.wanUpSpeed)!

        let wanDownSpeed = (homeSettings?.wanDownSpeed)!

        // 获取单位

        if wanUpSpeed.count > 4 {

            let leftUnit = NSString.init(format: "%@", wanUpSpeed).substring(from: wanUpSpeed.count-4)

            let leftNumber = NSString.init(format: "%@", wanUpSpeed).substring(to: wanUpSpeed.count-4)

            leftLabel.text = String(NSString.init(format: "%@", leftNumber))

            self.leftUnit.text = String(NSString.init(format: "%@", leftUnit))

        }

        if wanDownSpeed.count > 4 {

            let rightUnit = NSString.init(format: "%@", wanDownSpeed).substring(from: wanDownSpeed.count-4)

            let rightNumber = NSString.init(format: "%@", wanDownSpeed).substring(to: wanDownSpeed.count-4)

            rightLabel.text = String(NSString.init(format: "%@", rightNumber))

            self.rightUnit.text = String(NSString.init(format: "%@", rightUnit))

        }

    }

    func setSubDevice(homeSettings: HomeSettings?) {

        dataSource.removeAll()

        let mainModel = HomeModel()

        mainModel.name = (homeSettings?.mainDevName)!

        mainModel.number = (homeSettings?.mainDevSum)!

        mainModel.linkStatus = (homeSettings?.linkStatus)!

        mainModel.isChildNobe = false

        mainModel.isOnline = true

        dataSource.append(mainModel)

        if let array = homeSettings?.array {

            for homeSetting in array {

                let model = HomeModel()

                model.name = (homeSetting.childName)!

                model.number = (homeSetting.childDevNum)!

                model.mac = homeSetting.childDevMac!

                model.isChildNobe = true

                model.isOnline = true

                dataSource.append(model)

            }

        }

        otherInitGifView(dataSource)

    }

    func setMainSsid(\_ ssid: String?) {

        if let ssid = ssid {

            headerTitleLabel.text = ssid

        }

    }

    func setConnectedDevice(\_ number: String?) {

        if let number = number {

            bottomTitleLabel?.text = String(NSString.init(format: "%@ %@", number, language("台设备连接")))

        }

    }

}

// MARK: - just for gif view

extension HomeViewController {

    /// 获取所有位置信息的方法

    /// 返回的第一个永远是主view

    ///

    /// - Parameter count: gifview 的个数

    /// - Returns: 返回当前所有 gifview 的中心点信息

    func centerLocation(count: Int) -> [CGPoint] {

        switch count {

        case 1:

            let center = CGPoint(x: topMainViewWidth/2.0, y: topMainViewHeight/2.0)

            return [center]

        case 2:

            let center1 = CGPoint(x: topMainViewWidth/2.0, y: topMainViewHeight/2.0 - 90)

            let center2 = CGPoint(x: topMainViewWidth/2.0, y: topMainViewHeight/2.0 + 90)

            return [center1, center2]

        case 3:

            let center1 = CGPoint(x: topMainViewWidth/2.0, y: topMainViewHeight/2.0 - 90)

            let center2 = CGPoint(x: topMainViewWidth/2.0-topMainViewWidth/4, y: topMainViewHeight/2.0 + 90)

            let center3 = CGPoint(x: topMainViewWidth/2.0+topMainViewWidth/4, y: topMainViewHeight/2.0 + 90)

            return [center1, center2, center3]

        case 4:

            let center1 = CGPoint(x: topMainViewWidth/2.0, y: topMainViewHeight/2.0)

            let center2 = CGPoint(x: topMainViewWidth/2.0, y: topMainViewHeight/2.0 - 110)

            let center3 = CGPoint(x: topMainViewWidth/2.0-topMainViewWidth/4, y: topMainViewHeight/2.0 + 90)

            let center4 = CGPoint(x: topMainViewWidth/2.0+topMainViewWidth/4, y: topMainViewHeight/2.0 + 90)

            return [center1, center2, center3, center4]

        case 5:

            let center1 = CGPoint(x: topMainViewWidth/2.0, y: topMainViewHeight/2.0)

            let center2 = CGPoint(x: topMainViewWidth/2.0 - topMainViewWidth/4, y: topMainViewHeight/2.0 - 90)

            let center3 = CGPoint(x: topMainViewWidth/2.0 + topMainViewWidth/4, y: topMainViewHeight/2.0 - 90)

            let center4 = CGPoint(x: topMainViewWidth/2.0 - topMainViewWidth/4, y: topMainViewHeight/2.0 + 90)

            let center5 = CGPoint(x: topMainViewWidth/2.0 + topMainViewWidth/4, y: topMainViewHeight/2.0 + 90)

            return [center1, center2, center3, center4, center5]

        case 6:

            let center1 = CGPoint(x: topMainViewWidth/2.0, y: topMainViewHeight/2.0)

            let center2 = CGPoint(x: topMainViewWidth/2.0, y: topMainViewHeight/2.0 - 110)

            let center3 = CGPoint(x: topMainViewWidth/2.0 - topMainViewWidth/3, y: topMainViewHeight/2.0 - 30)

            let center4 = CGPoint(x: topMainViewWidth/2.0 + topMainViewWidth/3, y: topMainViewHeight/2.0 - 30)

            let center5 = CGPoint(x: topMainViewWidth/2.0 - topMainViewWidth/5, y: topMainViewHeight/2.0 + 100)

            let center6 = CGPoint(x: topMainViewWidth/2.0 + topMainViewWidth/5, y: topMainViewHeight/2.0 + 100)

            return [center1, center2, center3, center4, center5, center6]

        default:

            return [CGPoint.zero]

        }

    }

    // 设置当前view 的信息

    func setDeviceInformation(gifview: MainGifView,  model: HomeModel) {

        gifview.setMainLabel(text: String(model.name! + "(" + model.number! + ")"))

        gifview.viewType = model.isChildNobe ? .blue : .yellow

        if let linkStatus = model.linkStatus {

            if linkStatus == "1" {

                gifview.stop()

            } else if linkStatus == "0" {

                gifview.start()

            }

        } else {

            gifview.start()

        }

        gifview.viewTapBlock = { (gifView) in

            let vc = ChildNodeViewController.init()

            vc.model = model

            self.navigationController?.pushViewController(vc, animated: true)

        }

    }

    /// 初始化gifview

    func otherInitGifView(\_ array: Array<HomeModel>) {

        if array.count > 6 { return}

        if array.count < gifViewLoop.count {

            for (index, view) in gifViewLoop {

                if index > array.count-1 {

                    view.removeFromSuperview()

                }

            }

        }

        for index in 0..<array.count {

            let model = array[index]

            // 从数据池中获取view，若有则直接设置，没有则创建新的

            var view = gifViewLoop[index]

            if view == nil {

                // 没有这个view

                view = model.isChildNobe ? MainGifView.init(GifType.blue) : MainGifView.init(GifType.yellow)

                gifViewLoop[index] = view!

            }

            setDeviceInformation(gifview: view!, model: model)

            self.topMainView.addSubview(view!)

            view!.snp.remakeConstraints { (make) in

                make.center.equalTo(centerLocation(count: array.count)[index])

                make.width.equalTo(70)

                make.height.equalTo(90)

            }

        }

    }

}