ПРИЛОЖЕНИЕ

Листинг программы

Класс ZiroWeb

```
import Foundation

class ZiroWeb {

    private static let api = "http://ziroweb.azurewebsites.net/api"

    func signIn(withEmail email: String, andPassword passwod: String, completionHandler: @escaping (_ success: Bool, _ errors: [String]?) -> Void) {
        let json: [String: Any] = ["email": email,
                                   "password": passwod]
        let jsonData = try? JSONSerialization.data(withJSONObject: json)
        let loginUrl = URL(string: "\(ZiroWeb.api)/Account/Login")!
        var loginRequest = URLRequest(url: loginUrl)
        loginRequest.httpMethod = "POST"
        loginRequest.setValue("application/json; charset=utf-8", forHTTPHeaderField: "Content-Type")
        loginRequest.httpBody = jsonData

        let task = URLSession.shared.dataTask(with: loginRequest) { (data, response, error) in
            if let error = error {
                completionHandler(false, [error.localizedDescription])
                return
            }
            print(response!)
            if let httpResponse = response as? HTTPURLResponse, httpResponse.statusCode != 200 {
                completionHandler(false, ["Операция завершена неудачно"])
                return
            }
            guard
                let data = data,
                let responseJSON = try? JSONSerialization.jsonObject(with: data, options: []),
                let responseData = responseJSON as? [String: Any]
            else {
                completionHandler(false, ["Данные не получены"])
                return
            }
            let errors = responseData["errors"] as? [String]
            completionHandler(errors == nil, errors)
        }

        task.resume()
    }

    func signOut(completionHandler: @escaping () -> Void) {
        let logoutUrl = URL(string: "\(ZiroWeb.api)/Account/Logout")!
        let task = URLSession.shared.dataTask(with: logoutUrl) { (data, response, error) in
            guard let data = data, error == nil else {
                print(error?.localizedDescription ?? "No data")
                return
            }
            print(response!)
            let responseJSON = try? JSONSerialization.jsonObject(with: data, options: [])
            if let responseJSON = responseJSON as? [String: Any] {
                print(responseJSON)
            }
            completionHandler()
        }
        task.resume()
    }

    func testUser() {
        let testUrl = URL(string: "\(ZiroWeb.api)/Test/TestUser")!
        let testTask = URLSession.shared.dataTask(with: testUrl) { (data, response, error) in
            guard let data = data, error == nil else {
                print(error?.localizedDescription ?? "No data")
                return
            }
            print(response!)
            let responseJSON = try? JSONSerialization.jsonObject(with: data, options: [])
            if let responseJSON = responseJSON as? [String: Any] {
                print(responseJSON)
            }
        }
        testTask.resume()
    }

    func getUserProjects(withCompletion completion: @escaping (_ success: Bool, _ errors: [String]?, _ projects: [Project]?) -> Void)
{

        let projectsUrl = URL(string: "\(ZiroWeb.api)/project/getCurrentProjects")!
        let task = URLSession.shared.dataTask(with: projectsUrl) { (data, response, error) in
            if let error = error {
                completion(false, [error.localizedDescription], nil)
                return
            }
            print(response!)
            if let httpResponse = response as? HTTPURLResponse, httpResponse.statusCode != 200 {
                completion(false, ["Операция завершена неудачно"], nil)
                return
            }
            guard
                let data = data,
                let responseJSON = try? JSONSerialization.jsonObject(with: data, options: []),
                let responseData = responseJSON as? [String: Any]
                else {
                    completion(false, ["Данные не получены"], nil)
                    return
            }
            let errors = responseData["errors"] as? [String]
            if errors != nil {
                completion(false, errors, nil)
            }
            guard
                let dataNode = responseData["data"] as? [String: Any],
                let projectNode = dataNode["projects"] as? [[String: Any]]
                else {
                    completion(false, ["Некорректные данные"], nil)
                    return
            }
            let projects = projectNode.compactMap(Project.init)
            completion(true, nil, projects)
        }
        task.resume()
    }

    func getUserInfo(withCompletion completion: @escaping (_ success: Bool, _ errors: [String]?, _ account: AccountInfo?) -> Void)
{
        let url = URL(string: "\(ZiroWeb.api)/user/getProfile")!
        let task = URLSession.shared.dataTask(with: url) { (data, response, error) in
            if let error = error {
                completion(false, [error.localizedDescription], nil)
                return
            }
            print(response!)
            if let httpResponse = response as? HTTPURLResponse, httpResponse.statusCode != 200 {
                completion(false, ["Операция завершена неудачно"], nil)
                return
            }
            guard
                let data = data,
                let responseJSON = try? JSONSerialization.jsonObject(with: data, options: []),
                let responseData = responseJSON as? [String: Any]
                else {
                    completion(false, ["Данные не получены"], nil)
                    return
            }
            let errors = responseData["errors"] as? [String]
            if errors != nil {
                completion(false, errors, nil)
            }
            guard
                let dataNode = responseData["data"] as? [String: Any]
                else {
                    completion(false, ["Некорректные данные"], nil)
                    return
            }
            let account = AccountInfo(dataNode)
            completion(true, nil, account)
        }
        task.resume()
    }

    func getUserTasks(withCompletion completion: @escaping (_ success: Bool, _ errors: [String]?, _ projects: [ZTask]?) -> Void) {

        let taskUrl = URL(string: "\(ZiroWeb.api)/task/getCurrentTasks")!
        let task = URLSession.shared.dataTask(with: taskUrl) { (data, response, error) in
            if let error = error {
                completion(false, [error.localizedDescription], nil)
                return
            }
            print(response!)
            if let httpResponse = response as? HTTPURLResponse, httpResponse.statusCode != 200 {
                completion(false, ["Операция завершена неудачно"], nil)
                return
            }
            guard
                let data = data,
                let responseJSON = try? JSONSerialization.jsonObject(with: data, options: []),
                let responseData = responseJSON as? [String: Any]
                else {
                    completion(false, ["Данные не получены"], nil)
                    return
            }
            let errors = responseData["errors"] as? [String]
            if errors != nil {
                completion(false, errors, nil)
            }
            guard
                let dataNode = responseData["data"] as? [String: Any],
                let taskNode = dataNode["tasks"] as? [[String: Any]]
                else {
                    completion(false, ["Некорректные данные"], nil)
                    return
            }
            let tasks = taskNode.compactMap(ZTask.init)
            completion(true, nil, tasks)
        }
        task.resume()
    }

    func getTaskDetail(by taskId: String, withCompletion completion: @escaping (_ success: Bool, _ errors: [String]?, _ account: TaskDetail?) -> Void) {
        let url = URL(string: "\(ZiroWeb.api)/task/getTaskDetails")!

        let json: [String: Any] = ["taskId": taskId]
        let jsonData = try? JSONSerialization.data(withJSONObject: json)
        var request = URLRequest(url: url)
        request.httpMethod = "POST"
        request.setValue("application/json; charset=utf-8", forHTTPHeaderField: "Content-Type")
        request.httpBody = jsonData

        let task = URLSession.shared.dataTask(with: request) { (data, response, error) in
            if let error = error {
                completion(false, [error.localizedDescription], nil)
                return
            }
            print(response!)
            if let httpResponse = response as? HTTPURLResponse, httpResponse.statusCode != 200 {
                completion(false, ["Операция завершена неудачно"], nil)
                return
            }
            guard
                let data = data,
                let responseJSON = try? JSONSerialization.jsonObject(with: data, options: []),
                let responseData = responseJSON as? [String: Any]
                else {
```

```
                completion(false, ["Данные не получены"], nil)
                return
            }
            let errors = responseData["errors"] as? [String]
            if errors != nil {
                completion(false, errors, nil)
            }
            guard
                let dataNode = responseData["data"] as? [String: Any]
                else {
                    completion(false, ["Некорректные данные"], nil)
                    return
            }
            let taskDetails = TaskDetail(dataNode)
            completion(true, nil, taskDetails)
        }
        task.resume()
    }

    func addComment(for taskId: String, withText text: String, withCompletion completion: @escaping (_ success: Bool, _ errors: [String]?, _ comment: Comment?) -> Void) {
        let comment = Comment(["text": text, "leavingDate": "11.05.2019"])
        completion(true, nil, comment)
    }
}
```

Класс LoginViewController

```
import UIKit

class LoginViewController: UIViewController {

    @IBOutlet weak var loginView: UIView!
    @IBOutlet weak var loginTextBox: UITextField!
    @IBOutlet weak var passwordBox: UITextField!

    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view, typically from a nib.

    }

    override func viewDidLayoutSubviews() {
        super.viewDidLayoutSubviews()
        loginView.dropShadow(color: .black, opacity: 1, offSet: CGSize(width: -1, height: 1), radius: 3, scale: true)
    }

    @IBAction func signIn() {
        guard let login = loginTextBox.text, login.count > 0 else {
            handle(errors: ["Email должен быть заполнен"])
            return
        }
        guard let password = passwordBox.text, password.count > 0 else {
            handle(errors: ["Пароль должен быть заполнен"])
            return
        }
        ZiroWeb().signIn(withEmail: login, andPassword: password) { (success, errors) in
            if success {
                DispatchQueue.main.async {
                    self.performSegue(withIdentifier: "Main", sender: self)
                }
            } else if let errors = errors {
                self.handle(errors: errors)
            }
        }
    }
}
```

Класс TasksViewController

```
import UIKit

class TasksViewController: UIViewController {

    let ziroWeb = ZiroWeb()
    var tasks: [ZTask]?
    var filteredTasks: [ZTask]?

    var taskSource: [ZTask]? {
        return filteredTasks ?? tasks
    }

    private var selectedTask: ZTask?

    @IBOutlet weak var tableView: UITableView!
    private let refreshControl = UIRefreshControl()

    let statusColors = [
        0: UIColor(red: 76/255.0, green: 82/255.0, blue: 91/255.0, alpha: 1),
        1: UIColor(red: 8/255.0, green: 168/255.0, blue: 99/255.0, alpha: 1),
        2: UIColor(red: 237/255.0, green: 230/255.0, blue: 23/255.0, alpha: 1),
        3: UIColor(red: 232/255.0, green: 144/255.0, blue: 4/255.0, alpha: 1),
        4: UIColor(red: 15/255.0, green: 68/255.0, blue: 191/255.0, alpha: 1),
    ]

    let priorColors = [
        0: UIColor(red: 244/255.0, green: 199/255.0, blue: 0/255.0, alpha: 1),
        1: UIColor(red: 249/255.0, green: 197/255.0, blue: 99/255.0, alpha: 1),
        2: UIColor(red: 247/255.0, green: 161/255.0, blue: 2/255.0, alpha: 1),
        3: UIColor(red: 247/255.0, green: 92/255.0, blue: 2/255.0, alpha: 1),
        4: UIColor(red: 247/255.0, green: 14/255.0, blue: 2/255.0, alpha: 1),
    ]

    private var selectedSorting = 0

    @IBAction func sortSegmentChanged(_ sender: UISegmentedControl) {
        filteredTasks = nil
        self.selectedSorting = sender.selectedSegmentIndex
        if selectedSorting == 1 {
            filteredTasks = tasks?.filter({$0.priorityNum > 1})
        }
        tableView.reloadData()
    }

    override func viewDidLoad() {
        super.viewDidLoad()
        tableView.refreshControl = refreshControl
        // Configure Refresh Control
        refreshControl.addTarget(self, action: #selector(refreshData(_:)), for: .valueChanged)
        refreshControl.attributedTitle = NSAttributedString(string: "Загружаю данные...")
        // Do any additional setup after loading the view.
        ziroWeb.getUserTasks { [weak self] (success, errors, tasks) in
            guard let self = self else { return }
            if success, let tasks = tasks, tasks.count > 0 {
                self.tasks = tasks
                self.tasks?.sort(by: { (t1, t2) -> Bool in
                    return t1.priorityNum > t2.priorityNum
                })
                DispatchQueue.main.async {
                    self.tableView.reloadData()
                }
            } else if let errors = errors {
                self.handle(errors: errors)
            }
        }
    }

    override func prepare(for segue: UIStoryboardSegue, sender: Any?) {
        if segue.identifier == "TaskDetails" {
            guard let detailController = segue.destination as? TaskDetailViewController,
                let task = selectedTask else { return }
            detailController.taskId = task.id
        }
    }

    @objc private func refreshData(_ sender: Any) {
        // Fetch Weather Data
        filteredTasks = nil
        tasks = nil

        ziroWeb.getUserTasks { [weak self] (success, errors, tasks) in
            guard let self = self else { return }
            DispatchQueue.main.async {
                self.refreshControl.endRefreshing()
            }
            if success, let tasks = tasks, tasks.count > 0 {
                self.tasks = tasks
                self.tasks?.sort(by: { (t1, t2) -> Bool in
                    return t1.priorityNum > t2.priorityNum
                })
                self.filteredTasks = nil
                if self.selectedSorting == 1 {
                    self.filteredTasks = self.tasks?.filter({$0.priorityNum > 1})
                }
                DispatchQueue.main.async {
                    self.tableView.reloadData()
                }
            } else if let errors = errors {
                self.handle(errors: errors)
            }
        }
    }
}

extension TasksViewController: UITableViewDataSource {
    func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
        return taskSource?.count ?? 0
    }

    func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {
        let cell = tableView.dequeueReusableCell(withIdentifier: "taskCell", for: indexPath)
        if
            let statusView = cell.viewWithTag(100),
            let numLabel = cell.viewWithTag(101) as? UILabel,
            let priorityLabel = cell.viewWithTag(102) as? UILabel,
            let descriptionLabel = cell.viewWithTag(103) as? UILabel,
            let task = self.taskSource?[indexPath.row] {
            statusView.layer.cornerRadius = statusView.frame.height / 4.0
            statusView.backgroundColor = self.statusColors[task.statusNum]
            numLabel.text = task.number
            priorityLabel.text = task.priority
            priorityLabel.textColor = self.priorColors[task.priorityNum] ?? UIColor.black
            descriptionLabel.text = task.title
        }

        return cell
    }
}

extension TasksViewController: UITableViewDelegate {
    func tableView(_ tableView: UITableView, willSelectRowAt indexPath: IndexPath) -> IndexPath? {
        selectedTask = nil
        guard let tasks = taskSource, indexPath.row >= 0, indexPath.row < tasks.count else { return nil }
        selectedTask = tasks[indexPath.row]
        return indexPath
    }

    func tableView(_ tableView: UITableView, didSelectRowAt indexPath: IndexPath) {
        tableView.deselectRow(at: indexPath, animated: true)
    }
}
```

Класс ProjectsViewController

```
import UIKit

class ProjectsViewController: UIViewController {

    let ziroWeb = ZiroWeb()
    var projects: [Project]?
    var filteredProjects: [Project]?
```

```
private var selectedProject: Project?

var projectSource: [Project]? {
    return filteredProjects ?? projects
}

@IBOutlet weak var tableView: UITableView!

private let refreshControl = UIRefreshControl()
private var searchText: String?
private var selectedSorting = 0

override func viewDidLoad() {
    super.viewDidLoad()
    tableView.refreshControl = refreshControl
    // Configure Refresh Control
    refreshControl.addTarget(self, action: #selector(refreshData(_:)), for: .valueChanged)
    refreshControl.attributedTitle = NSAttributedString(string: "Загружаю данные...")
    // Do any additional setup after loading the view.
    ziroWeb.getUserProjects { [weak self] (success, errors, projects) in
        guard let self = self else { return }
        if success, let projects = projects, projects.count > 0 {
            self.projects = projects
            self.projects?.sort(by: { (p1, p2) -> Bool in
                return p1.name <= p2.name
            })
            DispatchQueue.main.async {
                self.tableView.reloadData()
            }
        } else if let errors = errors {
            self.handle(errors: errors)
        }
    }
}

override func prepare(for segue: UIStoryboardSegue, sender: Any?) {
    if segue.identifier == "ProjectDetails" {
        guard let detailController = segue.destination as? ProjectDetailViewController,
        let project = selectedProject else { return }
        detailController.project = project
    }
}

@IBAction func segmentChanged(_ sender: UISegmentedControl) {
    filteredProjects = nil
    self.selectedSorting = sender.selectedSegmentIndex
    projects?.sort(by: { (p1, p2) -> Bool in
        return sender.selectedSegmentIndex == 0 ? p1.name <= p2.name : p1.projectIndex >= p2.projectIndex
    })
    if let text = self.searchText {
        filteredProjects = projects?.filter({$0.name.contains(text)})
    }
    tableView.reloadData()
}

@objc private func refreshData(_ sender: Any) {
        // Fetch Weather Data
        filteredProjects = nil
        projects = nil

        ziroWeb.getUserProjects { [weak self] (success, errors, projects) in
            guard let self = self else { return }
            DispatchQueue.main.async {
                self.refreshControl.endRefreshing()
            }
            if success, let projects = projects, projects.count > 0 {
                self.projects = projects
                self.projects?.sort(by: { (p1, p2) -> Bool in
                    return self.selectedSorting == 0 ? p1.name <= p2.name : p1.projectIndex >= p2.projectIndex
                })
                if let text = self.searchText {
                    self.filteredProjects = self.projects?.filter({$0.name.contains(text)})
                }
                DispatchQueue.main.async {
                    self.tableView.reloadData()
                }
            } else if let errors = errors {
                self.handle(errors: errors)
            }
        }
    }
}

extension ProjectsViewController: UITableViewDataSource {
    func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
        return projectSource?.count ?? 0
    }

    func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {
        let cell = tableView.dequeueReusableCell(withIdentifier: "ProjectCell", for: indexPath)
        if let project = projectSource?[indexPath.row] {
            cell.textLabel?.text = project.name
            let projectStatus = "Запланированно: \(project.todoTaskCount) задач, в процессе: \(project.inProgressTaskCount)"
            cell.detailTextLabel?.text = projectStatus
        }
        return cell
    }
}

extension ProjectsViewController: UITableViewDelegate {

    func tableView(_ tableView: UITableView, willSelectRowAt indexPath: IndexPath) -> IndexPath? {
        selectedProject = nil
        guard let projects = projectSource, indexPath.row >= 0, indexPath.row < projects.count else { return nil }
        selectedProject = projects[indexPath.row]
        return indexPath
    }

    func tableView(_ tableView: UITableView, didSelectRowAt indexPath: IndexPath) {
        tableView.deselectRow(at: indexPath, animated: true)
    }
}
```

```
extension ProjectsViewController: UISearchBarDelegate {
    func searchBarSearchButtonClicked(_ searchBar: UISearchBar) {
        guard let text = searchBar.text, text.count > 2 else { return }
        searchText = text
        filteredProjects = projects?.filter({$0.name.contains(text)})
        self.tableView.reloadData()
    }

    func searchBar(_ searchBar: UISearchBar, textDidChange searchText: String) {
        if searchText == "" {
            self.searchText = nil
            filteredProjects = nil
            self.tableView.reloadData()
        }
    }
}
```

Класс ProjectDetailViewController

```
import UIKit

class ProjectDetailViewController: UITableViewController {

    var project: Project!

    @IBOutlet weak var nameLabel: UILabel!
    @IBOutlet weak var shortNameLabel: UILabel!
    @IBOutlet weak var descriptionLabel: UILabel!
    @IBOutlet weak var toDoLabel: UILabel!
    @IBOutlet weak var inProgressLabel: UILabel!

    override func viewDidLoad() {
        super.viewDidLoad()

        nameLabel.text = project.name
        shortNameLabel.text = project.shortName
        descriptionLabel.text = project.description
        toDoLabel.text = "\(project.todoTaskCount) задач"
        inProgressLabel.text = "\(project.inProgressTaskCount) задач"
    }

    // MARK: - Table view data source

    override func tableView(_ tableView: UITableView, didSelectRowAt indexPath: IndexPath) {
        if indexPath.section == 1 {
            dismiss(animated: true, completion: nil)
        } else {
            if let cell = tableView.cellForRow(at: indexPath), cell.tag == 100 {
                let alert = UIAlertController(title: nil, message: descriptionLabel.text, preferredStyle: .alert)
                alert.addAction(UIAlertAction(title: "OK", style: .cancel, handler: nil))
                present(alert, animated: true, completion: nil)
            }
            tableView.deselectRow(at: indexPath, animated: true)
        }
    }
}
```

Класс AccountViewController

```
import UIKit

class AccountViewController: UITableViewController {

    @IBOutlet weak var emailLabel: UILabel!
    @IBOutlet weak var firstNameLabel: UILabel!
    @IBOutlet weak var lastNameLabel: UILabel!
    @IBOutlet weak var skypeLabel: UILabel!
    @IBOutlet weak var phoneLabel: UILabel!
    @IBOutlet weak var dobLabel: UILabel!

    let ziroWeb = ZiroWeb()

    override func viewDidLoad() {
        super.viewDidLoad()
        ziroWeb.getUserInfo { [weak self] (success, errors, account) in
            DispatchQueue.main.async {
                guard let self = self else { return }
                if success, let account = account {
                    self.emailLabel.text = account.email
                    self.firstNameLabel.text = account.firstName
                    self.lastNameLabel.text = account.lastName
                    self.skypeLabel.text = account.skype
                    self.phoneLabel.text = account.phoneNumber
                    self.dobLabel.text = account.dateOfBirth
                } else if let errors = errors {
                    self.handle(errors: errors)
                }
            }
        }
    }
}
```

Класс TaskDetailViewController

```
import UIKit

class TaskDetailViewController: UITableViewController {

    var taskId: String!
    let ziroWeb = ZiroWeb()

    @IBOutlet weak var numLabel: UILabel!
    @IBOutlet weak var titleLabel: UILabel!
    @IBOutlet weak var typeLabel: UILabel!
    @IBOutlet weak var detailLabel: UILabel!
    @IBOutlet weak var controlLabel: UILabel!
    @IBOutlet weak var dateLabel: UILabel!
    @IBOutlet weak var estimationLabel: UILabel!
```

```
@IBOutlet weak var spentLabel: UILabel!
@IBOutlet weak var projectLabel: UILabel!
@IBOutlet weak var commentsLabel: UILabel!
@IBOutlet weak var logLabel: UILabel!

private var projectId: String = ""
private var taskProject: Project?
private var logs: [LogItem] = []
private var comments: [Comment] = []

override func viewDidLoad() {
    super.viewDidLoad()

    ziroWeb.getTaskDetail(by: taskId) { [weak self] (success, errors, taskDetails) in
        guard let self = self else { return }
        if success, let task = taskDetails {
            self.projectId = task.projectId
            self.logs = task.logs
            self.comments = task.comments
            DispatchQueue.main.async {
                self.numLabel.text = task.number
                self.titleLabel.text = task.title
                self.typeLabel.text = task.type
                self.detailLabel.text = task.description
                self.controlLabel.text = task.priority
                self.dateLabel.text = task.creationDate
                self.estimationLabel.text = "\(task.estimatedTime)"
                self.spentLabel.text = "\(task.spentTime)"
                self.projectLabel.text = task.projectName
                self.commentsLabel.text = "\(task.comments.count)"
                self.logLabel.text = "\(task.logs.count)"
            }
        } else if let errors = errors {
            self.handle(errors: errors)
        }
    }
}

override func prepare(for segue: UIStoryboardSegue, sender: Any?) {
    if segue.identifier == "ProjectDetails",
        let vc = segue.destination as? ProjectDetailViewController,
        let project = self.taskProject {
        vc.project = project
    } else if segue.identifier == "Logs",
        let vc = segue.destination as? LogsViewController {
        vc.logs = self.logs
    } else if segue.identifier == "Comments",
        let vc = segue.destination as? CommentsViewController {
        vc.taskId = self.taskId
        vc.comments = self.comments
    }
}

override func tableView(_ tableView: UITableView, didSelectRowAt indexPath: IndexPath) {
    if let cell = tableView.cellForRow(at: indexPath) {
        switch cell.tag {
            case 100: alert(withDetail: titleLabel.text)
            case 101: alert(withDetail: detailLabel.text)
            case 300:
                ziroWeb.getUserProjects { (success, errors, projects) in
                    if success, let projects = projects {
                        self.taskProject = projects.first(where: {$0.id == self.projectId})
                        if self.taskProject != nil {
                            DispatchQueue.main.async {
                                self.performSegue(withIdentifier: "ProjectDetails", sender: self)
                            }
                        }
                    } else if let errors = errors {
                        self.handle(errors: errors)
                    }
                }
            case 600: dismiss(animated: true, completion: nil)
            default: break
            }
        }
        tableView.deselectRow(at: indexPath, animated: true)
    }

    private func alert(withDetail detailText: String?) {
        let alert = UIAlertController(title: nil, message: detailText, preferredStyle: .alert)
        alert.addAction(UIAlertAction(title: "OK", style: .cancel, handler: nil))
        present(alert, animated: true, completion: nil)
    }
}
```

Класс LogsViewController

```
import UIKit

class LogsViewController: UITableViewController {

    var logs: [LogItem]!

    override func viewDidLoad() {
        super.viewDidLoad()
    }

    // MARK: - Table view data source

    override func numberOfSections(in tableView: UITableView) -> Int {
        // #warning Incomplete implementation, return the number of sections
        return 2
    }

    override func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
        // #warning Incomplete implementation, return the number of rows
        return section == 0 ? 1 : logs.count
    }


    override func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {
        if indexPath.section == 0 {
            let closeCell = tableView.dequeueReusableCell(withIdentifier: "CloseCell", for: indexPath)
            return closeCell
        }
        let cell = tableView.dequeueReusableCell(withIdentifier: "LogCell", for: indexPath)
        if
            let authorLabel = cell.viewWithTag(100) as? UILabel,
            let dateLabel = cell.viewWithTag(101) as? UILabel,
            let spentLabel = cell.viewWithTag(102) as? UILabel,
            let textLabel = cell.viewWithTag(103) as? UILabel {
            let log = self.logs[indexPath.row]
            authorLabel.text = log.authorName
            dateLabel.text = log.date
            spentLabel.text = "Затратил часов: \(log.spentTimeHours)"
            textLabel.text = log.text
        }

        return cell
    }

    override func tableView(_ tableView: UITableView, heightForRowAt indexPath: IndexPath) -> CGFloat {
        return indexPath.section == 0 ? 40.0 : 120.0
    }

    override func tableView(_ tableView: UITableView, didSelectRowAt indexPath: IndexPath) {
        tableView.deselectRow(at: indexPath, animated: true)
        if indexPath.section == 0 {
            self.dismiss(animated: true, completion: nil)
        }
    }
}
```

Класс CommentsViewController

```
import UIKit

class CommentsViewController: UIViewController {

    var comments: [Comment]!
    var taskId: String!
    let ziroWeb = ZiroWeb()

    @IBOutlet weak var tableView: UITableView!
    @IBOutlet weak var navBar: UINavigationItem!

    override func viewDidLoad() {
        super.viewDidLoad()
        let closeButton = UIBarButtonItem(title: "Назад",  style: .plain, target: self, action: #selector(close))
        let addButton = UIBarButtonItem(title: "Добавить",  style: .plain, target: self, action: #selector(add))
        navBar.leftBarButtonItem = addButton
        navBar.rightBarButtonItem = closeButton
    }

    @objc func close() {
        self.dismiss(animated: true, completion: nil)
    }

    @objc func add() {
        let alert = UIAlertController(title: "Добавить комментарий", message: "Введите комментарий", preferredStyle: .alert)
        alert.addAction(UIAlertAction(title: "Отмена", style: .cancel, handler: nil))
        alert.addAction(UIAlertAction(title: "Сохранить", style: .default, handler: { _ in
            guard let text = alert.textFields?[0].text, text.count > 0 else {
                self.handle(errors: ["Комментарий не должен быть пустым"])
                return
            }
            self.ziroWeb.addComment(for: self.taskId, withText: text, withCompletion: { (success, errors, comment) in
                if success, let comment = comment {
                    self.comments.insert(comment, at: 0)
                    DispatchQueue.main.async {
                        self.tableView.reloadData()
                    }
                } else if let errors = errors {
                    self.handle(errors: errors)
                }
            })
        }))
        alert.addTextField(configurationHandler: nil)
        present(alert, animated: true, completion: nil)
    }
}

extension CommentsViewController: UITableViewDataSource {
    func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
        return comments.count
    }

    func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {
        let cell = tableView.dequeueReusableCell(withIdentifier: "CommentCell", for: indexPath)
        if
            let authorLabel = cell.viewWithTag(100) as? UILabel,
            let dateLabel = cell.viewWithTag(101) as? UILabel,
            let textLabel = cell.viewWithTag(102) as? UILabel {
            let comment = self.comments[indexPath.row]
            authorLabel.text = comment.authorName
            dateLabel.text = comment.date
            textLabel.text = comment.text
        }

        return cell
    }
}

extension CommentsViewController: UITableViewDelegate {
    func tableView(_ tableView: UITableView, didSelectRowAt indexPath: IndexPath) {
        tableView.deselectRow(at: indexPath, animated: true)
    }
}
```