

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ

РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА ПО ДИСЦИПЛИНЕ
«ПРОЕКТНАЯ ДЕЯТЕЛЬНОСТЬ»

на тему:

Куратор проекта: _____ / _____, _____ /
подпись _____ ФИО, уч. звание и степень

Студент: _____ / _____, _____ /
подпись _____ ФИО, группа

Студент: _____ / _____, _____ /
подпись _____ ФИО, группа

Студент: _____ / _____, _____ /
подпись _____ ФИО, группа

Москва, 2020

ЗАДАНИЕ НА ВЫПОЛНЕНИЕ ПО ПРОЕКТУ

Тема: мобильное приложение на операционной системе iOS для сервиса по изучению русского языка как иностранному Una

Исходный код мобильного приложения: <https://github.com/Nelmeris/Una-iOS>

Собранное приложение на мобильном устройстве можно получить у презентующих.

Назначение: использование мобильного приложения в качестве тренажера и замены веб-сервиса для изучения русского языка студентами, преподавателями или просто людьми, желающими самостоятельно изучить русский язык.

Состав проекта:

- Куфаев Артем Александрович - капитан и главный разработчик iOS
- Сметанина Александра Александровна - UI-дизайнер
- Давтаев Артур Русланович - UI/UX-дизайнер

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	4
СОДЕРЖАТЕЛЬНЫЕ ГЛАВЫ	5
ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ	5
ПОСТАНОВКА ЗАДАЧИ, ОПРЕДЕЛЕНИЕ ЦЕЛЕЙ РАБОТЫ	5
ОБЗОР МЕТОДОВ И АЛГОРИТМОВ РЕШЕНИЯ ЗАДАЧ	6
ВЫБОР И ОБОСНОВАНИЕ МЕТОДА ДОСТИЖЕНИЯ ЦЕЛЕЙ	12
АРХИВ ПРОТОТИПОВ	32
ТЕХНИЧЕСКАЯ РЕАЛИЗАЦИЯ ПРОДУКТА	33
ЗАКЛЮЧЕНИЕ	35
СПИСОК ИСТОЧНИКОВ	36

ВВЕДЕНИЕ

Наша тема - мобильное приложение на операционной системе iOS для сервиса обучения русскому языку как иностранному Юна.

Заказчикам требовался сервис по обучению русскому языку с примером работы хотя бы одного курса по выданной теме с реализацией одного типа решения тренажера. Мы продолжаем свою работу, начатую в прошлом семестре и берем на себя новые задачи

Проект актуален в связи с отсутствием качественных сервисов подобной темы в странах СНГ или где-либо еще (со слов заказчика), которым могли бы воспользоваться люди, желающие изучить русский язык, или преподаватели, подбирающие материал и сервисы для обучения своих студентов.

Мобильные приложения по изучению иностранного языка популярны и наиболее удобны к использованию, нежели использование веб-сервиса через мобильный браузер устройства, чему, как минимум, способствует возможность контроля UI/UX-дизайна на мобильной версии сервиса. Помимо этого, мобильное приложение может предоставить пользователю ряд преимуществ - уведомления или напоминания о уроках, локальная база данных, позволяющая создать Offline-режим или хотя-бы повышенную загрузку и скорость работы

Источник: <https://habr.com/ru/post/168843/>

Данное приложение подойдет как обычным студентам, так и людям, желающим, по разным причинам, самостоятельно изучить русский язык в разных уголках мира.

СОДЕРЖАТЕЛЬНЫЕ ГЛАВЫ

ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

В настоящее время наиболее популярными мобильными ОС являются Android компании Google, iOS - Apple и Windows Phone компании Microsoft. Вторую позицию занимает операционная система iOS. По данным Яндекс.Метрики с 1 апреля 2015 года по 19 мая 2019 года эта ОС используется 29.06% пользователей в России <https://radar.yandex.ru/mobile?tablePeriod=all&period=all>. За последние 4 года доля iOS в России постепенно падает. Так, 1 апреля 2015 года ее доля составляла 35.95%, а 1 марта 2019 - уже 24.83%.

ПОСТАНОВКА ЗАДАЧИ, ОПРЕДЕЛЕНИЕ ЦЕЛЕЙ РАБОТЫ

Главной задачей данного проекта было создать iOS приложение для сайта Юна, которое сможет работать Offline. В нем должна быть основа для создания и переноса с веб версии тренажеров для обучения и тренировки русского языка, для людей, у которых русский язык не является основным или вообще не изучался. Приложение должно быть понятным, удобным в использовании, приятным глазу и запоминающимся.

ОБЗОР МЕТОДОВ И АЛГОРИТМОВ РЕШЕНИЯ ЗАДАЧ

Стартовая конфигурация приложения вынесена в отдельный класс:

```

import UIKit

class Start {

    static public let shared = Start()
    private init() {
        BaseUIOperation.queue.maxConcurrentOperationCount = 1
        BaseDatabaseOperation.queue.maxConcurrentOperationCount = 1
        BaseBackendOperation.queue.maxConcurrentOperationCount = 1
        let coreDataURL =
            BaseCoreDataService.persistentContainer.persistentStoreDescriptions.first?.url?.absoluteString
        print("CoreData URL: \(coreDataURL ?? "NULL")")
        // AuthManager.shared.logout()
        // UserDefaults.standard.removeObject(forKey: isWelcomeKey)
        // clearDatabase()
    }

    private func clearDatabase() {
        let context = BaseCoreDataService.persistentContainer.viewContext
        let userCD = UserCoreDataService(context: context)
        let lessonCD = LessonCoreDataService(context: context)
        let lessonPartCD = LessonPartCoreDataService(context: context)
        let lessonTaskCD = LessonTaskCoreDataService(context: context)
        do {
            try userCD.removeAll()
            try lessonCD.removeAll()
            try lessonPartCD.removeAll()
            try lessonTaskCD.removeAll()
        } catch {
            print(error)
        }
    }

    private let welcomeIds = (storyboardName: "Welcome", vcId: "Page")
    private let authIds = (storyboardName: "Authorization", vcId: "Auth")
    private let studyIds = (storyboardName: "Study", vcId: "Study")
    private let isWelcomeKey = "isWelcome"

    public func configureWindow(_ window: UIWindow) {
        if #available(iOS 13.0, *) {
            window.overrideUserInterfaceStyle = .light
        }
    }

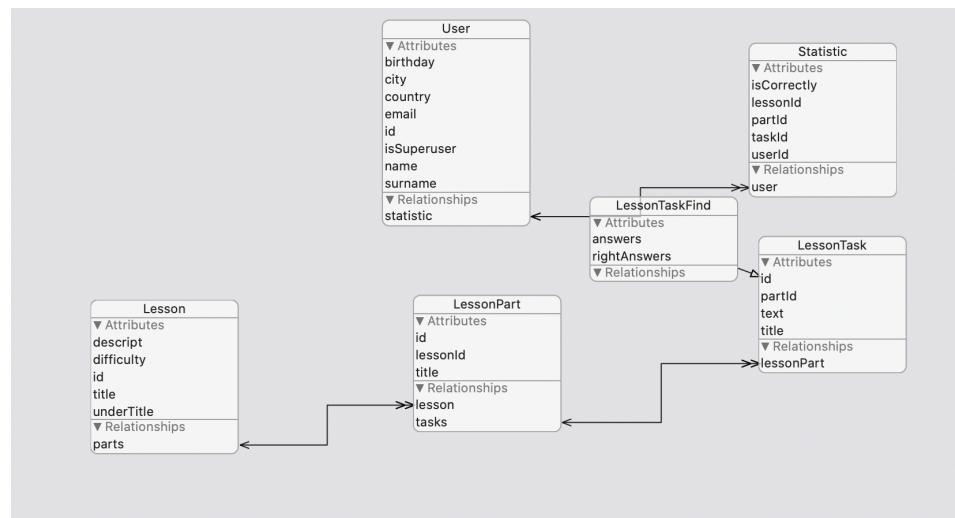
    public func configureNavBarAppearance() {
        let navigationBarAppearence = UINavigationBar.appearance()
        navigationBarAppearence.isTranslucent = false
        navigationBarAppearence.tintColor = UIColor(named: "MainColor")
        navigationBarAppearence.barTintColor = .white
        navigationBarAppearence.titleTextAttributes = [NSAttributedString.Key.foregroundColor: UIColor(named: "MainColor")!]
    }

    public func configureBarButtonItemAppearance() {
        let BarButtonItemAppearence = UIBarButtonItem.appearance()
        BarButtonItemAppearence.setBackButtonTitlePositionAdjustment(UIOffset(horizontal: -1000.0, vertical: 0.0), for: .default)
    }

    public func getRootController() -> UIViewController {
        let userDefaults = UserDefaults.standard
        if !userDefaults.bool(forKey: isWelcomeKey) { // If need present WelcomePageViewController
            return UIStoryboard(name: welcomeIds.storyboardName, bundle: nil).instantiateViewController(withIdentifier: welcomeIds.vcId)
        } else {
            if !AuthManager.shared.isAuthenticated() { // If need authorization
                let storyboard = UIStoryboard(name: authIds.storyboardName, bundle: nil)
                let authVC = storyboard.instantiateViewController(withIdentifier: authIds.vcId)
                let navControl = UINavigationController(rootViewController: authVC)
                navControl.modalPresentationStyle = .fullScreen
                return navControl
            } else {
                let storyboard = UIStoryboard(name: studyIds.storyboardName, bundle: nil)
                let studyVC = storyboard.instantiateViewController(withIdentifier: studyIds.vcId)
                let navControl = UINavigationController(rootViewController: studyVC)
                studyVC.modalPresentationStyle = .fullScreen
                return navControl
            }
        }
    }
}

```

Структура локальной базы данных:



Пример сервиса локальной базы данных

```

class UserCoreDataService: BaseCoreDataService {

    private let entityName = "User"

    func load() throws -> [User] {
        let request: NSFetchedRequest<User> = NSFetchedRequest(entityName: entityName)
        request.sortDescriptors = [ NSSortDescriptor(key: "id", ascending: false) ]
        return try context.fetch(request)
    }

    func load(email: String) throws -> User? {
        let request: NSFetchedRequest<User> = NSFetchedRequest(entityName: entityName)
        request.predicate = NSPredicate(format: "email = %@", email)
        return try context.fetch(request).first
    }

    enum SaveResult {
        case success(User)
        case failure(Error)
    }

    func save(authUser: UnaAuthUser, profile: UnaUserProfile, completion: @escaping (SaveResult) -> ()) throws {
        let user = try load(email: authUser.email) ?? User(context: context)
        user.id = Int64(authUser.id)
        user.email = authUser.email
        user.name = authUser.firstName
        user.surname = authUser.lastName
        user.city = profile.city
        user.country = profile.country
        if let birthday = profile.date {
            user.birthday = UserCoreDataService.dateFormatter.date(from: birthday)
        }
        saveContext {
            if let error = $0 {
                completion(.failure(error))
            } else {
                completion(.success(user))
            }
        }
    }

    static var dateFormatter: DateFormatter = {
        let dateFormatter = DateFormatter()
        dateFormatter.dateFormat = "dd.MM.yyyy"
        return dateFormatter
    }()

    func removeAll() throws {
        let data = try self.load()
        data.forEach { self.context.delete($0) }
    }
}
  
```

Пример сервиса работы с удаленной базой данных:

```

9 import Foundation
10
11 protocol UnaDBServiceLessonsFactory {
12     func getLessons(complete: @escaping ([UnaLesson]) -> ()) throws
13     func getLessonParts(for id: Int, complete: @escaping ([UnaLessonPart]) -> ()) throws
14 }
15
16 extension UnaDBService: UnaDBServiceLessonsFactory {
17
18     func getLessons(complete: @escaping ([UnaLesson]) -> ()) throws {
19         try self.request(with: (getLessonsSQL(), [])) { values in
20             var lessons: [UnaLesson] = []
21             try values.forEach { try lessons.append(UnaLesson(from: $0)) }
22             complete(lessons)
23         }
24     }
25
26     private func getLessonsSQL() -> String {
27         let lessonsTable = "lessons_lesson"
28         return "SELECT * FROM \(lessonsTable);"
29     }
30
31     func getLessonParts(for id: Int, complete: @escaping ([UnaLessonPart]) -> ()) throws {
32         try self.request(with: (getLessonPartsSQL(), [String(id)])) { values in
33             var parts: [UnaLessonPart] = []
34             try values.forEach { try parts.append(UnaLessonPart(from: $0)) }
35             complete(parts)
36         }
37     }
38
39     private func getLessonPartsSQL() -> String {
40         let lessonPartsTable = "lessons_lessonpart"
41         return "SELECT * FROM \(lessonPartsTable) WHERE lesson_id = $1"
42     }
43
44 }
```

Для реализации API сервисов с помощью Moya достаточно реализовать класс-наследник `Moya.TargetType`:

```

9 import Moya
10
11 enum UnaAPI {
12     case login(String, String)
13 }
14
15 extension UnaAPI : TargetType {
16
17     var baseURL: URL {
18         return URL(string: "http://5.23.55.140/api/")!
19     }
20
21     var path: String {
22         switch self {
23             case .login:
24                 return "token/"
25         }
26     }
27
28     var method: Method {
29         switch self {
30             case .login:
31                 return .post
32         }
33     }
34
35     var sampleData: Data {
36         return Data()
37     }
38
39     var task: Task {
40         switch self {
41             case .login(let email, let password):
42                 return .requestParameters(parameters: ["username": email, "password": password], encoding: JSONEncoding.default)
43         }
44     }
45
46     var headers: [String : String]? {
47         return ["Content-type": "application/json; charset=UTF-8"]
48     }
49
50 }
```

После этого очень просто создавать под-сервисы для различных модулей, к примеру, для авторизации:

```

import Foundation
import Moya
import SwiftyJSON

class UnaNetworkManager: UnaNetworkable {

    public static let shared = UnaNetworkManager()

    private let errorParser: UnaNetworkErrorParser
    private init() {
        errorParser = UnaNetworkErrorParser()
    }

    var provider = MoyaProvider<UnaAPI>()

    func login(email: String, password: String, completion: @escaping (AccessToken?, Error?) -> ()) {
        provider.request(.login(email, password), errorParser: errorParser, objectModel: AccessToken.self, success: { (response) in
            completion(response, nil)
        }) { (error) in
            completion(nil, error)
        }
    }
}

```

После написания сервисов по управлению сырьими данными с БД и сервером нужно назначить работу этих данных в приложении соответствующим потокам. Для этого я использую систему GCD и Operation. Сначала опишем базовую асинхронную операцию. С помощью потомков подобного класса можно регулировать количество выполняемых одновременно операций, отслеживать их выполнение, завершение, ошибки и т.д.. Далее создаю соответствующие операции для каждого действия с данными. Такой подход также инкапсулирует в себя большое количество кода, что предварительно берет на себя соответствующую ответственность.

```

9 import Foundation
10
11 class AsyncOperation: Operation {
12
13     private var _executing = false
14     private var _finished = false
15
16     let commonQueue = OperationQueue()
17
18     override var isAsynchronous: Bool {
19         return true
20     }
21
22     override var isExecuting: Bool {
23         return _executing
24     }
25     override var isFinished: Bool {
26         return _finished
27     }
28
29     override func start() {
30         guard !isCancelled else {
31             finish()
32             return
33         }
34         willChangeValue(forKey: "isExecuting")
35         _executing = true
36         didChangeValue(forKey: "isExecuting")
37         main()
38     }
39
40     override func main() {
41         fatalError("Should be overriden")
42     }
43
44     func finish() {
45         willChangeValue(forKey: "isFinished")
46         _finished = true
47         didChangeValue(forKey: "isFinished")
48     }
49
50 }
51

```

Пример операции, загружающей данные пользователя из сети:

```

9 import Foundation
10
11 enum LoadUserBackendOperationResult {
12     case success(UnaAuthUser, UnaUserProfile)
13     case notFound
14     case failure(Error)
15 }
16
17 class LoadUserBackendOperation: BaseBackendOperation {
18
19     private(set) var result: LoadUserBackendOperationResult? { didSet { finish() } }
20     private let email: String
21
22     init(userEmail: String) {
23         self.email = userEmail
24         super.init()
25     }
26
27     override func main() {
28         guard !self.isCancelled else { return }
29         do {
30             try UnaDBService.shared.getUser(with: email) { user in
31                 guard let user = user else {
32                     self.result = .notFound
33                     return
34                 }
35                 do {
36                     try UnaDBService.shared.getUserProfile(with: user.id) { profile in
37                         guard let profile = profile else {
38                             self.result = .notFound
39                             return
40                         }
41                         self.result = .success(user, profile)
42                     }
43                 } catch {
44                     self.result = .failure(error)
45                 }
46             }
47         } catch {
48             self.result = .failure(error)
49         }
50     }
51 }
52 }
```

Как можно видеть, операция манипулирует лишь сервисами по работе с бэкэндом, что создаёт слой между сервисами по сырым данным и менеджерами данных для отображения. Последним шагом будет создание соответствующих DataManager'ов. Пример менеджера работы с данными уроков:

```

9 import Foundation
10
11 final class LessonsDataManager: BaseDataManager {
12
13     static let `default` = LessonsDataManager()
14
15     enum GetLessonsResult {
16         case success([Lesson])
17         case failure(Error)
18     }
19
20     func getLessons(completion: @escaping (GetLessonsResult) -> ()) {
21         let loadOperation = LoadLessonsUIOperation(context: context, backendQueue: backendQueue, dbQueue: dbQueue)
22         loadOperation.loadFromDatabase.completionBlock = {
23             let result = loadOperation.loadFromDatabase.result!
24             switch result {
25                 case .success(let lessons):
26                     completion(.success(lessons))
27                 case .failure(let error):
28                     completion(.failure(error))
29             }
30         }
31         loadOperation.completionBlock = {
32             let result = loadOperation.result!
33             switch result {
34                 case .success(let lessons):
35                     completion(.success(lessons))
36                 case .notFound: break
37                 case .failure(let error):
38                     completion(.failure(error))
39             }
40         }
41         uiQueue.addOperation(loadOperation)
42     }
}

```

После этого можно сказать, что все слои обработки данных и ядро были описаны, теперь остается применить созданные менеджеры для отображения данных на экране. Суть менеджеров данных в том, что при их использовании можно не беспокоиться о том, как он работает и смело можно забыть про все слои ниже

В архитектуре MVP для отображения данных используются: Presenters, ViewModels, ViewControllers, Views:

- Presenter загружает данные и подготавливает их в соответствующую форму для дальнейшего отображения
- ViewModel служит структурой, в которую Presenter парсит данные, пришедшие из DataManager

ViewController запрашивает данные у Presenter и, используя ViewModels, настраивает соответствующие View на себе

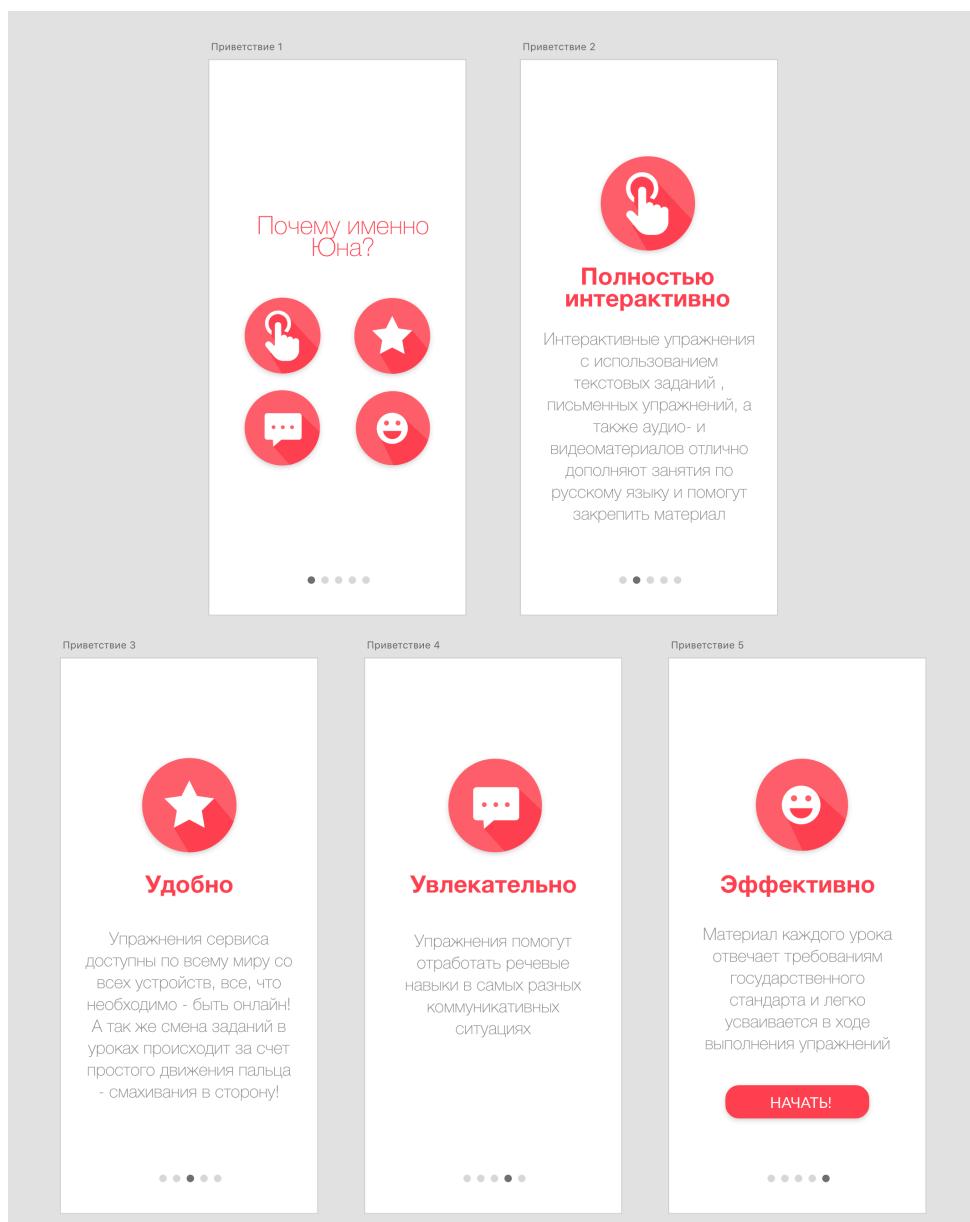
View являются графическими контейнерами данных, которые отображают их на экране устройства надлежащим видом

Помимо всех этих основных структур, моделей, сервисов и менеджеров в приложении, в поддержку им, были созданы вспомогающие классы и использованы паттерны проектирования

ВЫБОР И ОБОСНОВАНИЕ МЕТОДА ДОСТИЖЕНИЯ ЦЕЛЕЙ

Так как это является вторым томом, а точнее семестром работы нашей команды над этим проектом было решено для пущего интереса переделать всё. Поэтому сначала в это секции будут видны идеи и макеты не только новой версии приложения, но так же и старой, дабы показать прогресс и этапы креативного смещения проекта.

Итак, прототипирование. Далее пойдут вещи по тому порядку, по которому пользователь будет воспринимать. То есть, к примеру, первое, что он увидит перед экраном регистрации/входа - окна приветствия. При первом входе в приложение пользователю показывается основная информация о приложении и приветствие (в прошлой версии этого не было):



Приветственные страницы

Иконки взяты с сайта «Una» (создана еще иконка для страницы «Полностью интерактивно») и описание тоже, добавлена информация про пользование жестами в нашем приложении. В конце есть кнопка «Начать», при нажатии на которую открывается окно входа

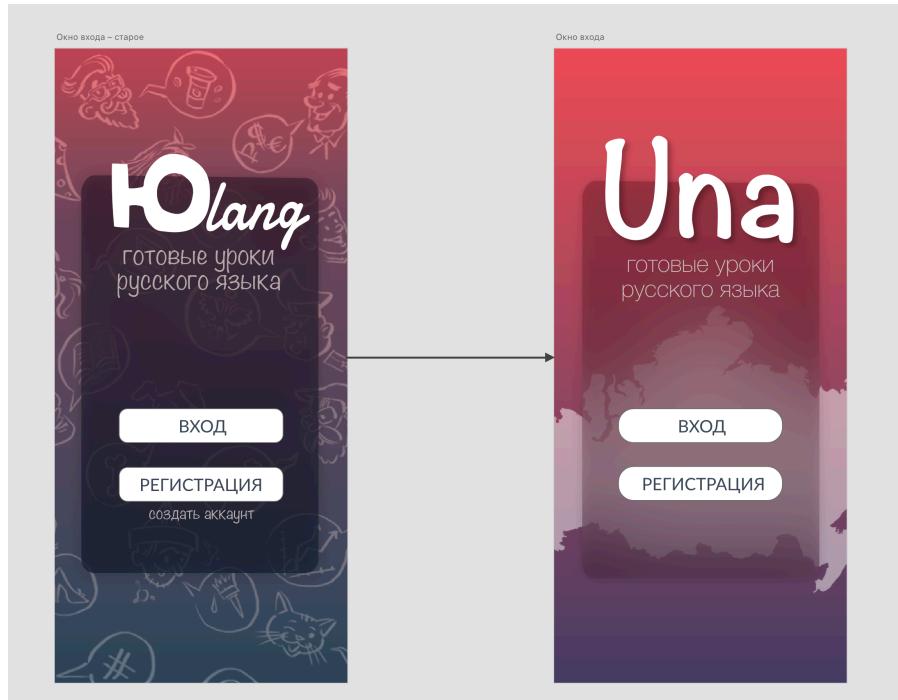
Далее после нажатия кнопки «Начать!» открывается окно входа. Итак, чему нужно придерживаться при создании меню входа:

1. Нужно определить ценность входа для пользователя.

Регистрация - последнее, что пользователи хотят делать. Как правило, пользователи не хотят регистрироваться, пока не увидят в этом ценность. Надо чётко определять ценность предложения по заполнению формы. Сообщение должно быть ясным и "острым".

2. Чётко показать, куда нажимать, чтобы войти.

Когда пользователь заходит в новое приложение, надо четко показать ему, где зарегистрироваться или войти



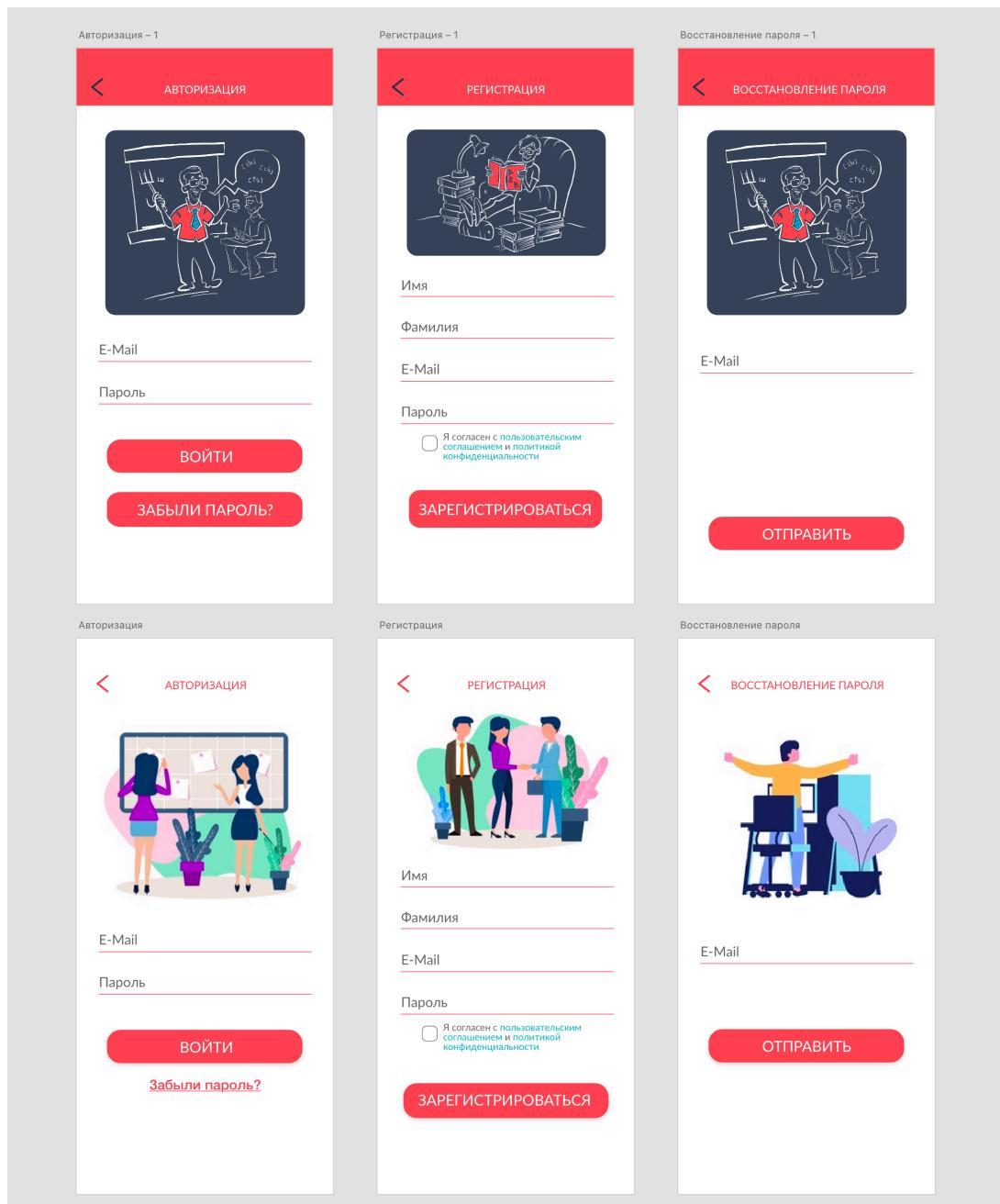
Старое и новое окно входа

Теперь перечислим причины, по которым мы решили изменить окно входа и за счет чего мы это сделали:

- Смена названия приложения. Изначальное название было «YouLang», теперь же «Юна» или же «Una»

- Из-за смены главной темы основного сайта, картинки на заднем фоне нашей заставки стали неактуальными, т.к. данный стиль больше не используется на «Una»
- Картинки еще были убраны из-за слишком большого их количества, что мешала пользователю сосредоточиться на чем-то одном, взгляд разбегался
- Теперь же на заднем фоне более приятные цвета (более яркий розовый и синий ближе к фиолетовому, нежели серо-синему, как изначально) и фрагмент карты России, взятый с основного сайта Una, чтобы соответствовать ему по стилю. Цвета соответствуют основным цветам приложения
- Изменен шрифт текста, чтобы не выбиваться из основного стиля приложения, т.к. старый текст больше нигде не использовался, что делало его каким-то лишним
- Убрана подпись под кнопкой «регистрация», чтобы не вводить пользователя в заблуждение, т.к. создавалось ощущение, что это интерактивная надпись, при нажатии на которую будет создаваться новый аккаунт, при наличии кнопки «регистрация» это мешало
- Темный прямоугольник за надписями и кнопками был освещен, чтобы он не сильно выбивался на фоне всего остального и не чернил все основные цвета
- По итогу было убрано много всего лишнего, осталось только все необходимое

На данном окне есть 2 кнопки - «Вход» и «Регистрация». Они тоже подверглись небольшим изменениям



Авторизация/регистрация/восстановление пароля - сверху старые окна, снизу новые

Что изменилось:

- Из-за смены главной стилистики сайта «Una», пришлось изменить и картинки на этих окнах, т.к. они были взяты с сайта «YouLang» и не подходили под дальнейшую стилистику приложения
- Кнопка «Забыли пароль?» в окне «Авторизация» была заменена на подчеркнутую надпись под кнопкой «Войти»
- NavigationBar сверху был на протяжении всего приложения был кораллового цвета, но сейчас для общей стилистики приложения был убран фон (т.е. просто осталось стандартный белый фон) и текстом кораллово цвета

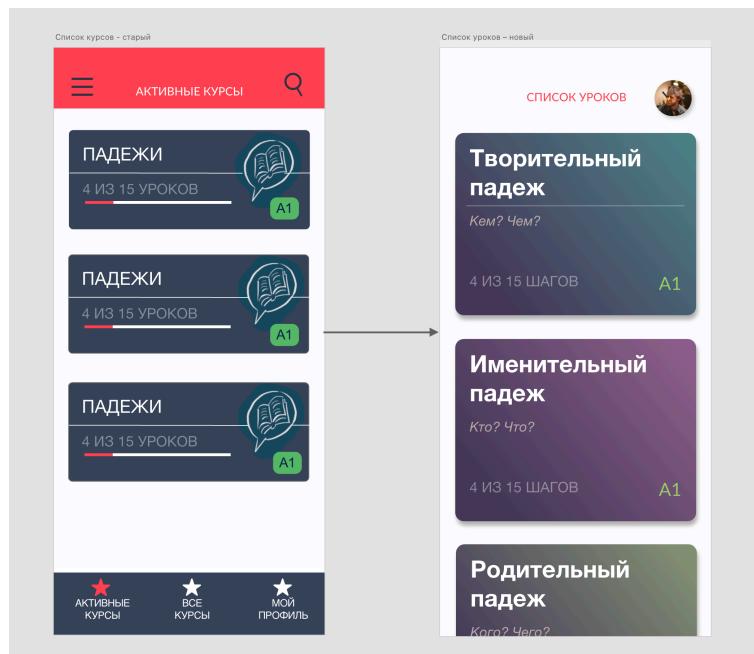
- Изображения теперь заменены векторными картинками (в дальнейшем будут заменены картинками, нарисованными в нашей команде)
- Теперь кнопки хорошо выделяются и в окне ничего не мешает концентрации на главных вещах

Далее, полгода назад шло одно из главнейших элементов любого приложения, сайта: главное меню, dashboard, с помощью которого будет осуществляться навигация по приложению. По старому ТЗ нам надо расположить на dashboard следующий элемент: список курсов и прочие приятности. Вот наш бывший дашборд:



Разделы меню

Однако жизнь внесла свои корректизы. Дашборда больше нет. В связи с новым курсом, который выбрала обновлённая Юна, в дашборде пропала необходимость, ведь всё что должно быть, это список уроков и профиль. Поэтому был выбран максимально упрощенный, лаконичный и честный дизайн:



Старое и новое окно списка уроков

Основные изменения коснулись и самих уроков. Раньше были курсы, в которых уже были уроки, и показанное выше окно называлось «Активные курсы», теперь оно имеет название

«Список укоров». Т.к. отпала надобность вкладок «все курсы» и «активные курсы», они были заменены одним окном «Список уроков»

Изменения, которые коснулись основных пунктов приведенного выше окна:

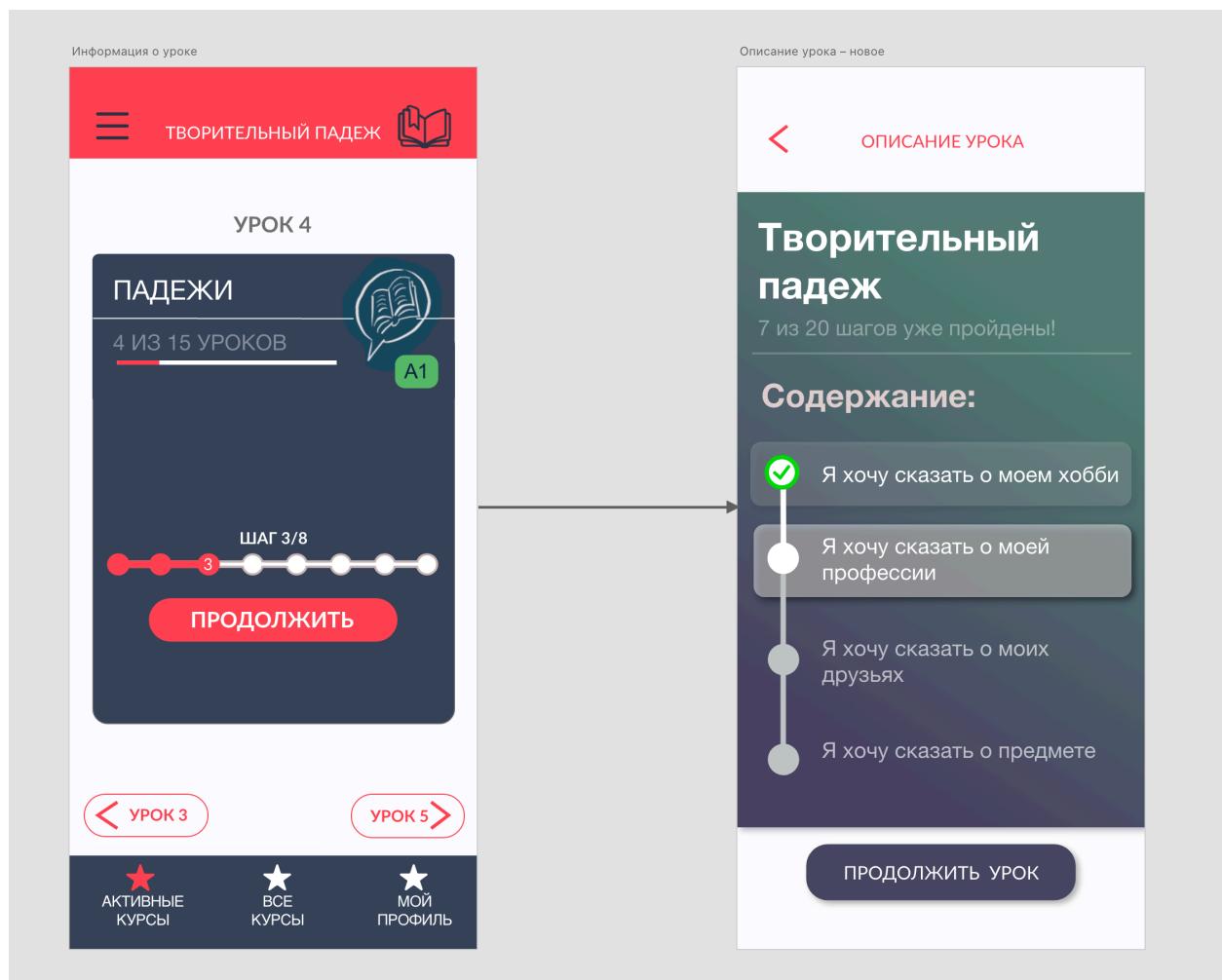
- Поменялись карточки уроков (в предыдущей версии карточки курсов). Они стали больше и разных цветов
- В новой версии каждому уроку соответствует индивидуальный градиент, который далее будет использоваться в самих заданиях урока. Данное решение помогает пользователю отличать уроки друг от друга не только по названию
- После названия написаны пару слов об уроке, в данном случае к уроку «Творительный падеж» написано описание «Кем? Чем?», что дает краткую и понятную информацию об уроке
- Далее, написана информация о пройденных шагах в уроке, чтобы пользователь понимал, какую часть урока он еще не закончил
- Справа снизу же написан уровень урока (от A1 до C2), который отражает его сложность. Теперь на эту информацию уделяется не такой сильный акцент, как в предыдущей версии, где данная информация находится в цветном прямоугольнике. Теперь же это просто текст, который с одной стороны заметен

Весь UI/UX дизайн выполнен согласно Apple Human Interface Guidelines, согласно которым интерфейс должен соответствовать, и у нас он соответствует следующим 3-м постулатам:

- **Чистота.** Во всей системе текст должен быть разборчив в любом размере, значки четкие и понятные, украшения тонкие и уместные, дабы создавать четкий фокус на функциональности приложения. Негативное пространство, цвет, шрифты, графика и элементы интерфейса должны тонко выделять важный контент и передавать интерактивность.
- **Deference.** Текущее движение и четкий, красивый интерфейс помогают людям понимать контент и взаимодействовать с ним, никогда не соревнуясь с ним. Контент обычно занимает весь экран, а прозрачность и размытость часто намекают на большее. Минимальное использование рамок и теней делает интерфейс легким и воздушным, обеспечивая при этом функционалу первостепенное значение.

- **Глубина.** Четкие визуальные слои и реалистичные движения должны передавать иерархию и облегчать понимание.

В приложении достаточно сильно изменилось промежуточное окно между самими заданиями в уроке



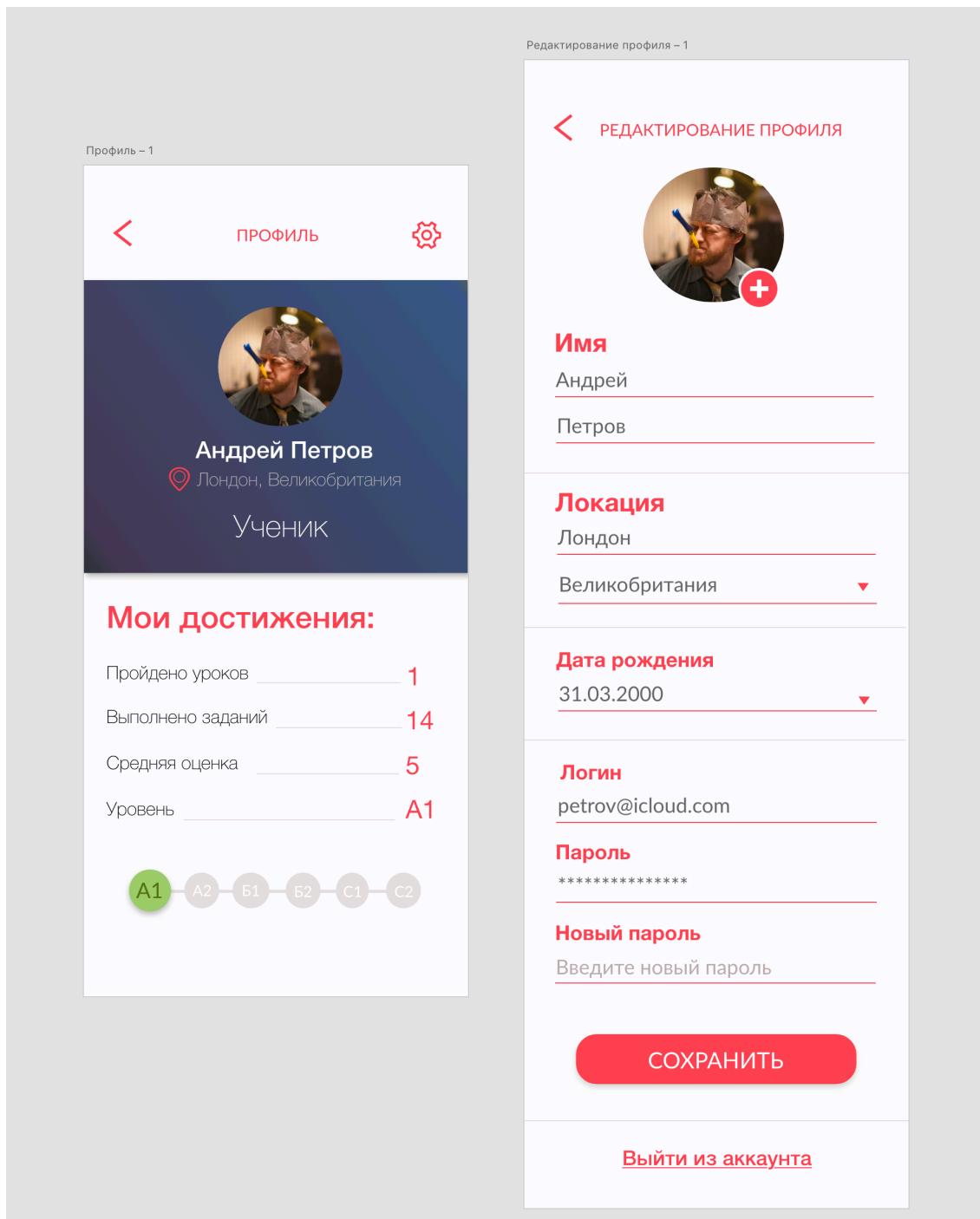
Окно описания урока - слева старое, справа новое

Что и почему было изменено:

- Опять же, из-за отсутствия дашборда, освобождается достаточно много места и сверху не давит «шапка» кораллового яркого цвета
- Структура уроков была изменена (это будет разбираться чуть позже), из-за чего появились подразделы в уроках. В данном случае (в окнах выше), содержатся 4 подраздела в уроке, а именно:
 - «Я хочу сказать о моем хобби»
 - «Я хочу сказать о моей профессии»

- «Я хочу сказать о моих друзьях»
- «Я хочу сказать о предмете»
- Ниже названия урока пишется количество пройденных шагов. В отличие от предыдущей версии, была убрана шкала количества, которая показывала те же самые данные повторно, из-за этого и была лишней
- В старой версии можно было перемещаться между соседними уроками, т.к. они все содержались в одном курсе. Т.к. курсов больше нет, а есть только отдельные уроки, в этом окне их пролистывать нет смысла
- Кнопка «Продолжить урок» сделана сине-фиолетовым цветом, в отличие от прошлой версии - коралловой. Так, она не сильно бросается в глаза, но при этом она достаточно заметна
- После прохождения одного или более подраздела, появляется возможность вернуться к той части урока и перепройти её. Для этого нужно нажать на нужную часть, после чего она станет выделена (как в пункте «Я хочу сказать о моей профессии» на окнах выше) и на кнопку «Продолжить урок»

Из окна «Список уроков» так же можно, помимо самого урока, зайти в профиль пользователя, нажав на аватарку пользователя справа сверху



Окно профиля и редактирования профиля

В окне профиля отображаются:

- Аватар
- Локация
- Статус пользователя (в данном случае - ученик)
- Количество пройденных уроков
- Количество выполненных заданий
- Средняя оценка пользователя (от 2 до 5)

- Уровень пользователя (от A1 до C2)
- Иллюстрация уровня сложности для понимания того, как выглядит шкала оценивания сложности

Далее же, аватар, локацию, дату рождения, пароль и почту можно будет менять в окне «Редактирование профиля», где еще можно выйти из аккаунта, чтобы в дальнейшем зайти за другого пользователя

Переходя к самим заданиям урока, после нажатия на кнопку «Продолжить урок» в окне «Описание урока», открывается окно с заданием. Заданий в уроке может быть разное количество, они разделяются на подразделы, после прохождения каждого подводятся итоги с подсчётом количества правильно и неправильно решенных заданий

Было реализовано 5 различных типа заданий:

1. Перетаскивание

В заданиях с "перетаскиванием" пользователю необходимо перетащить нужное слово в нужное, заранее выделенное место в тексте. Из-за возможного большого количества слов была реализована удобная и интуитивная горизонтальная прокрутка вставляемых в текст слов. С помощью этого решения освободилось больше места для самого текста, в котором, при необходимости тоже реализована прокрутка

2. Задание с выбором из списка (одно значение)

Первоначально в этом задании задумывалась одновременное отображение нескольких однотипных заданий с единственным выбором, однако по причине нехватки места и неудобного выбора из-за небольшого размера получаемого текста было принято решение оставить только одно задание на один экран, для упрощения и удобства пользования.

3. Выделение в тексте

Пользователю будет предоставлен текст, в котором ему понадобится «выделять» нужные по заданию слова. «Выделение» будет происходить при нажатии пользователем на одно из выделенных слов, после чего оно будет выделено белым цветом с коралловой рамкой. При повторном нажатии на слово выделение будет снято

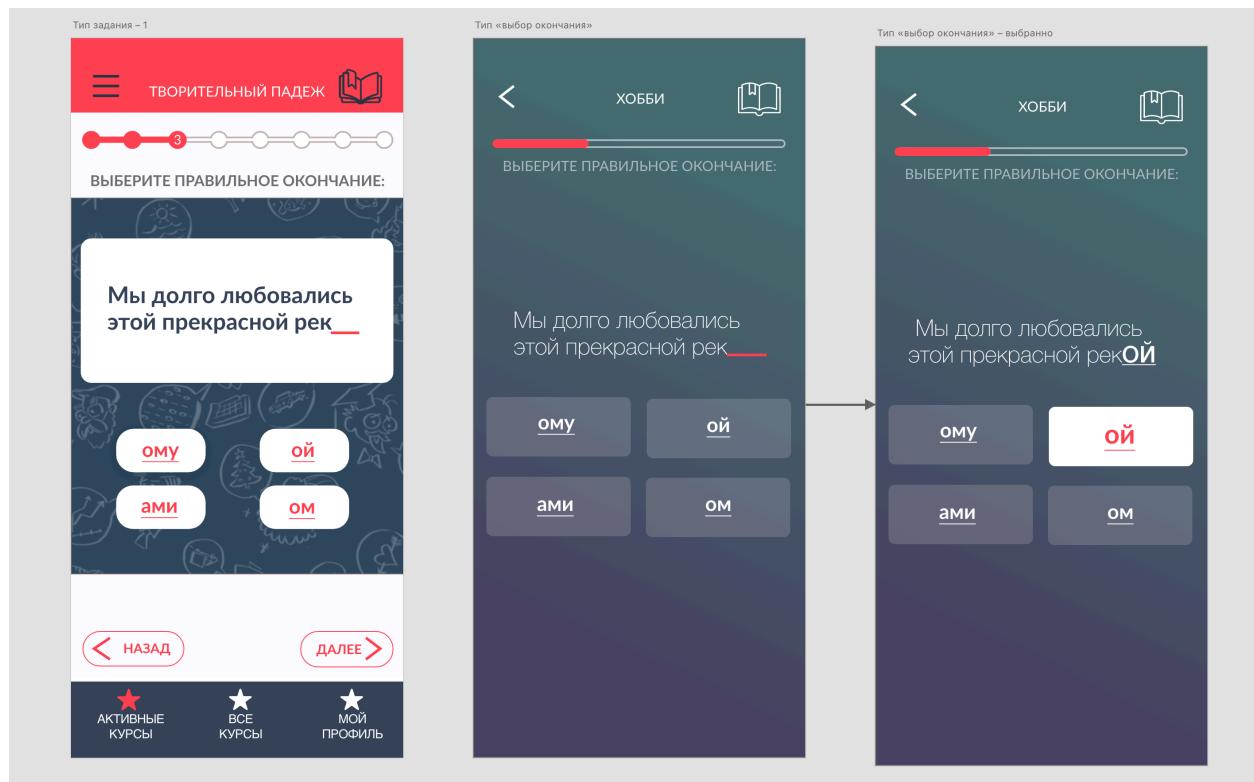
4. Задание с вводом с клавиатуры окончания

В некоторых заданиях пользователю будет необходимо ввести значения с клавиатуры. Пустые места для ввода помечены на экране, при нажатии на которых открывается встроенная в телефон клавиатура для ввода данных.

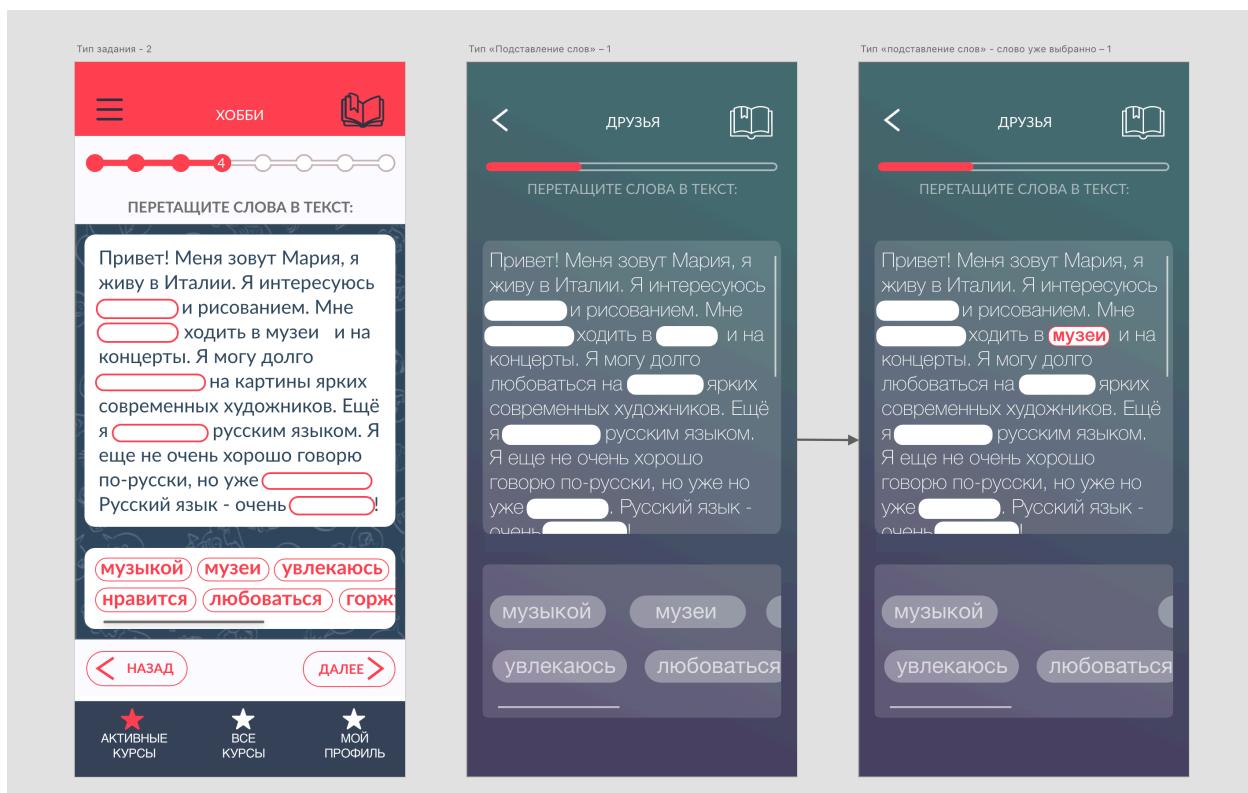
5. Задание с вводом с клавиатуры слова

Так же, как и в предыдущем варианте, пользователю нужно будет вписать в поле слово с клавиатурой, слово же это берется из скобочек после поля, после чего нужно будет поставить это слово в нужную форму

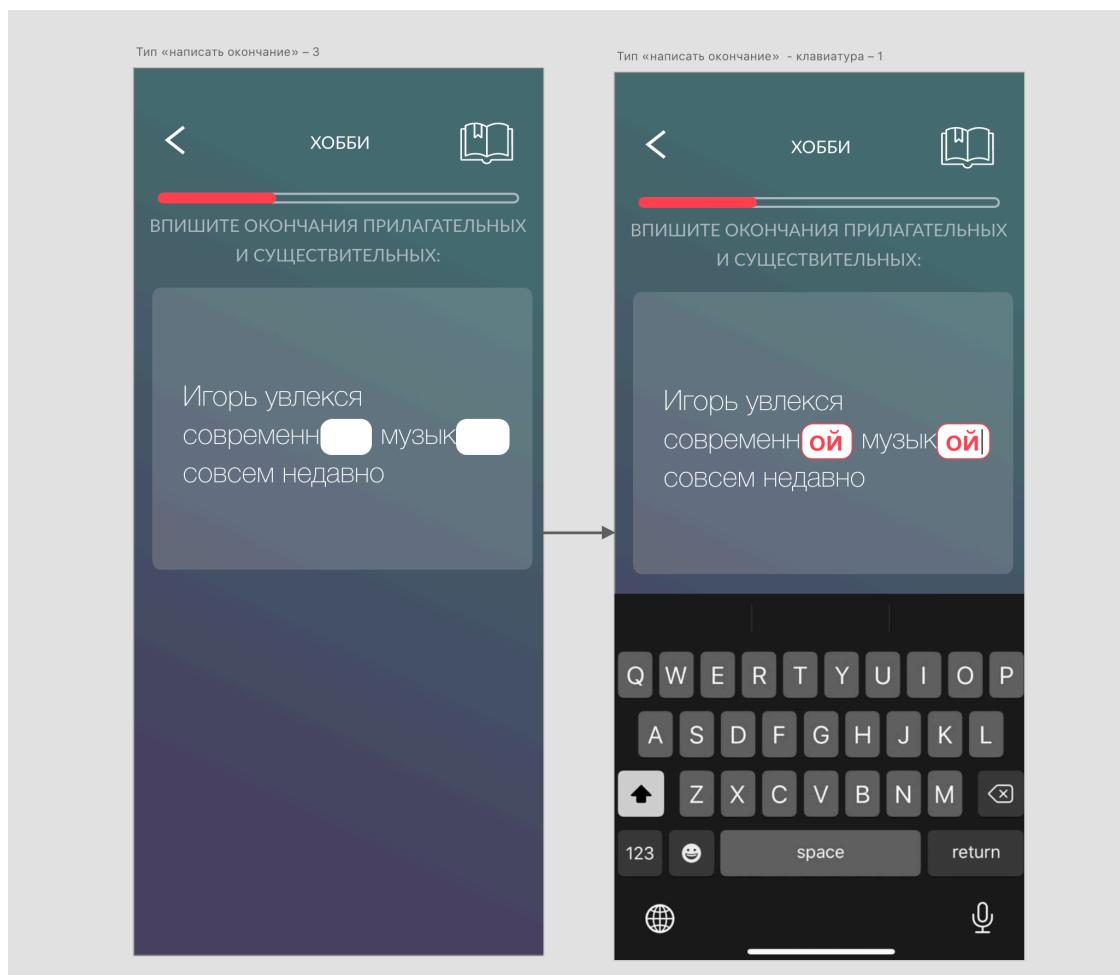
Далее будут показаны сами варианты заданий, первое окно будет заданием со старым дизайном, второе с новым, третье же с ходом выполнения задания (если окна 2, значит, в старом дизайне данного типа задания не было)



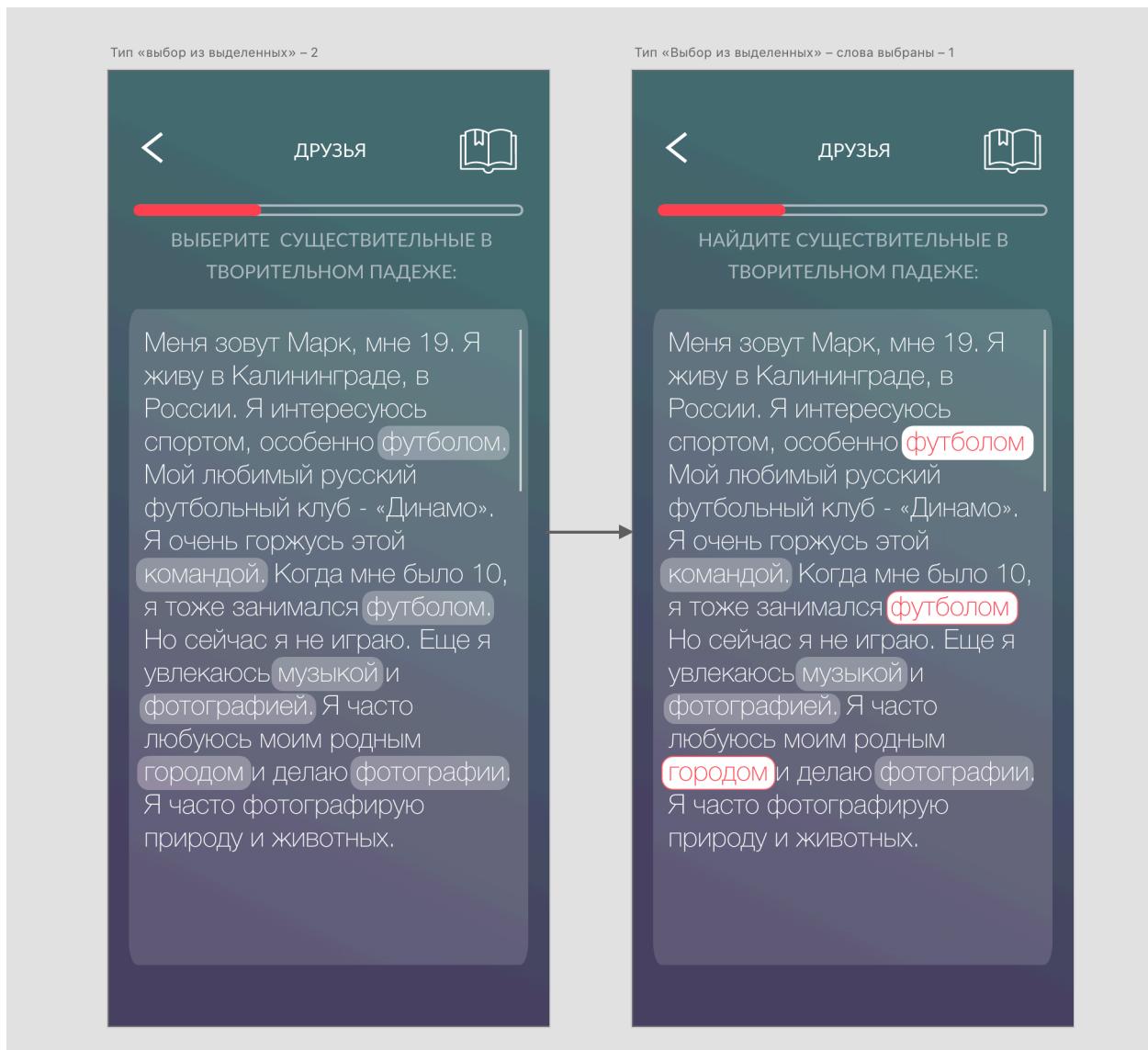
Тип задания «Выбор окончания»



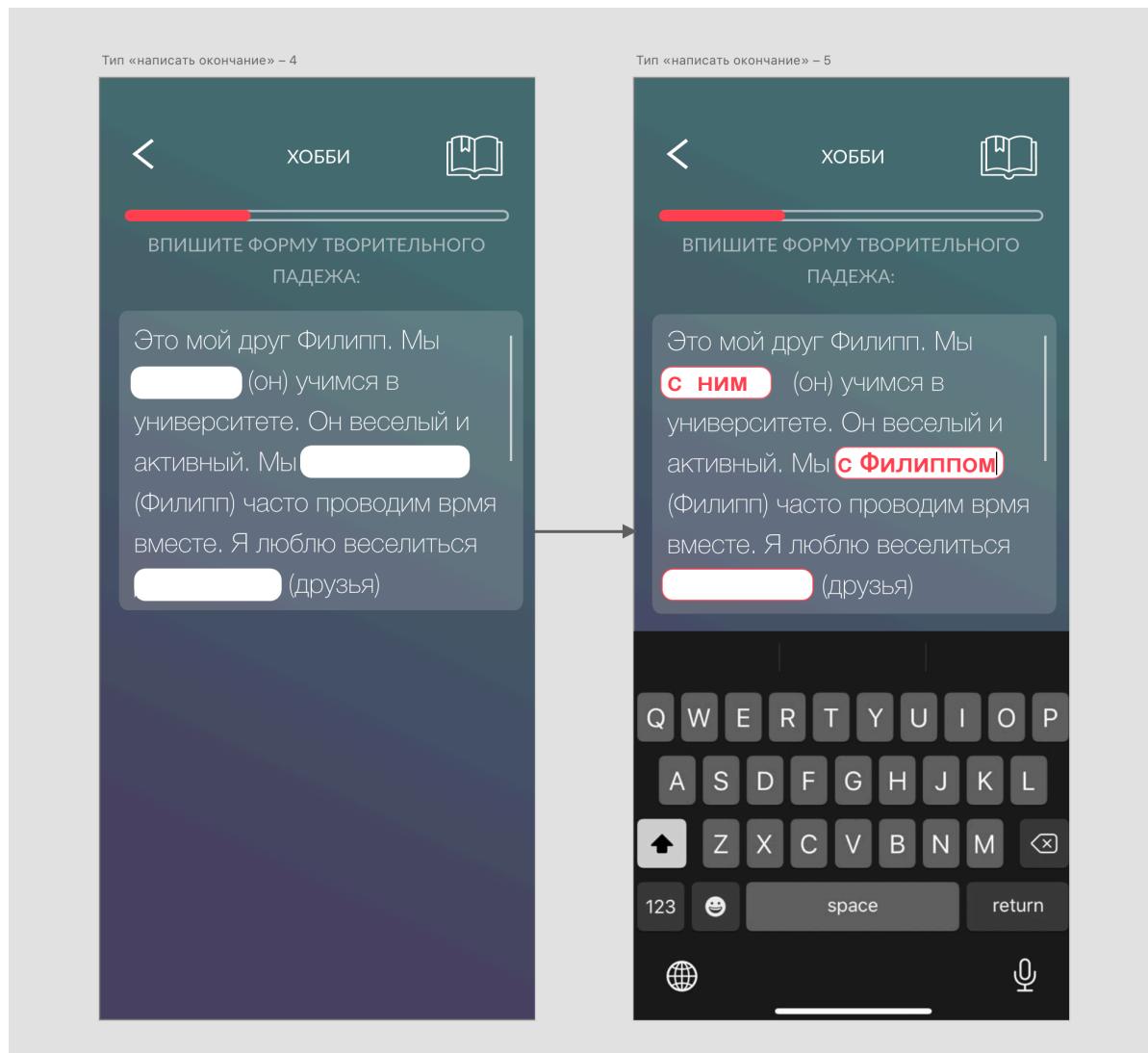
Тип задания «Вставить слова»



Тип задания «Вписать окончания»

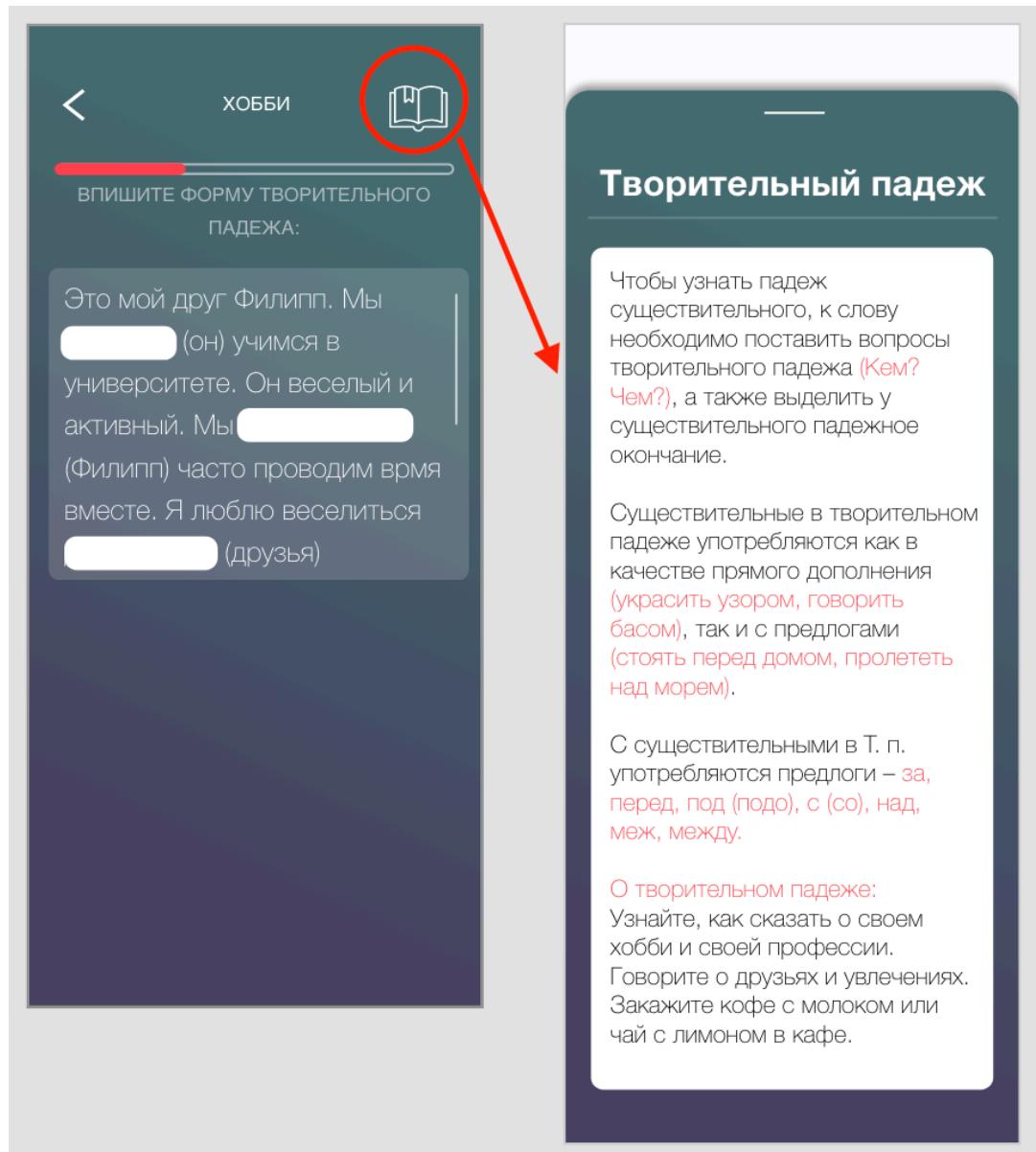


Тип задания «Выберите слова»



Тип задания «Впишите слова из скобок в правильной форме»

Сверху справа у каждого задания есть иконка книги, что является теорией. При нажатии на нее снизу выезжает окошко с текстом, которое можно после прочтения смахнуть вниз



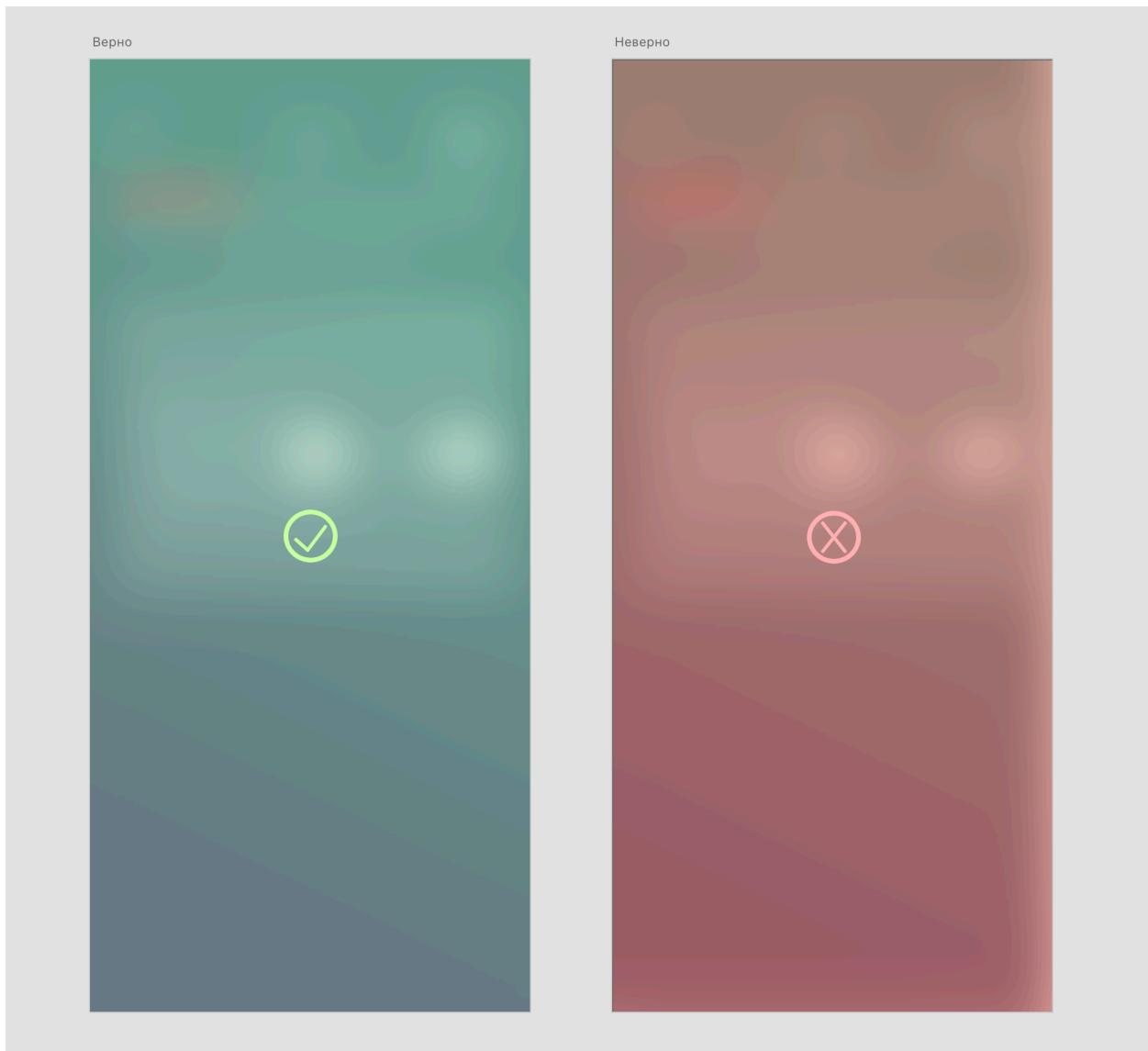
Окно теории

Все тренажеры в макетах были разработаны так, чтобы пользователю было удобно их использовать:

- Кнопки для перемещения были заменены интуитивным свайпом, информация о котором была написана в приветственных окнах приложения
- Шрифт большой, легко читается, никакой лишней информации, кроме задания на экране нет
- Блоки, которые нужно перемещать, тоже достаточно большие, их удобно передвигать и попадать по ним пальцем
- Места для изменения слов, вставки частей теста специально выделены белым цветом для того, чтобы пользователь лучше понимал, то с этими местами нужно взаимодействовать

- Сверху есть шкала прохождения этапов урока, она заполняется по мере прохождения упражнений, с помощью чего можно с легкостью отследить, сколько было пройдено и сколько осталось еще пройти
- Под шкалой для каждого шага написано описание того, что в задании нужно сделать

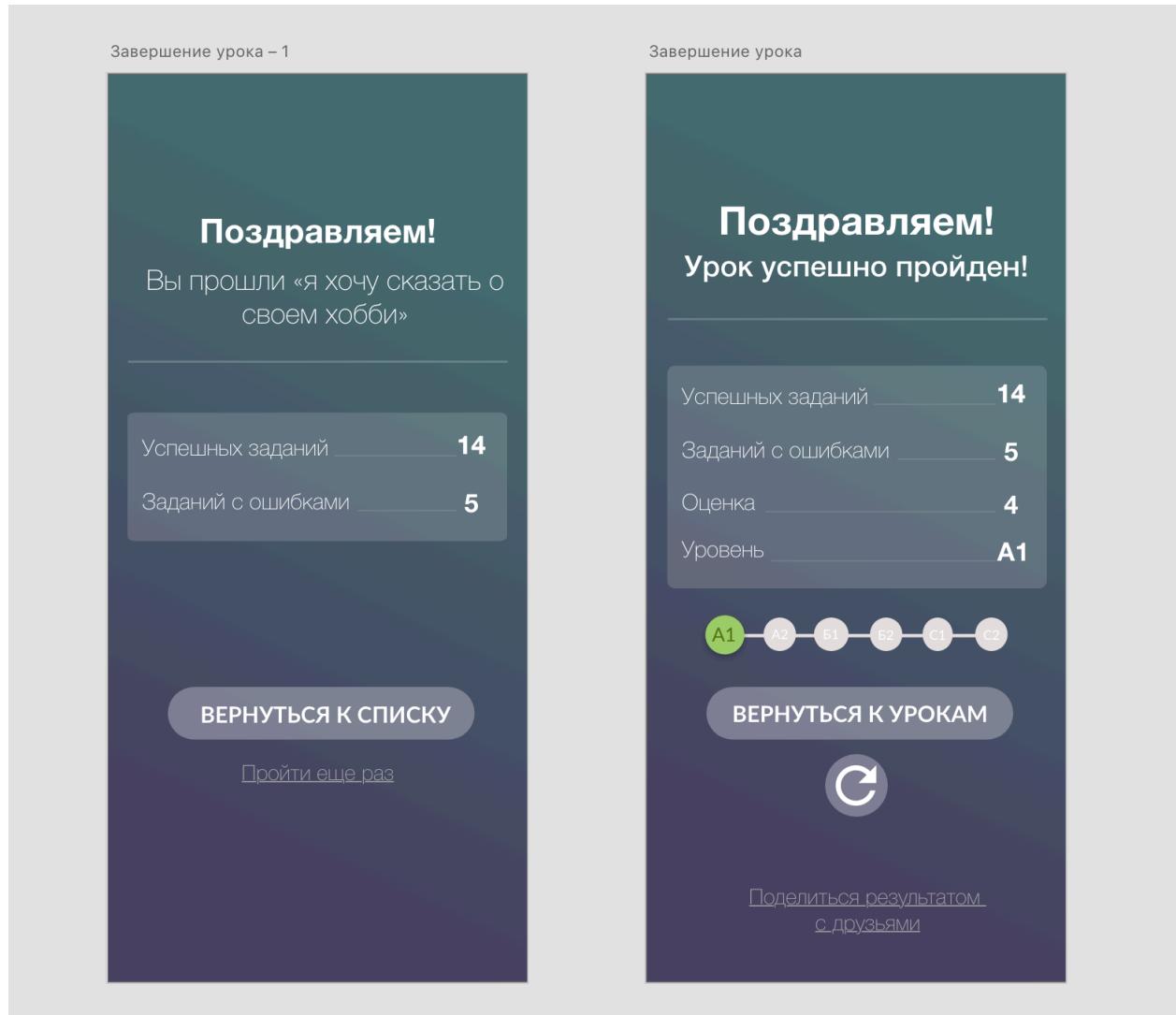
После свайпа заполненного задания на экране появляются окна со значком либо правильного, либо неправильного выполнения



Окно проверки задания

Далее, после прохождения части урока, например, «Я хочу сказать о моем хобби» появляется промежуточное окно с результатами, где подсчитывается количество правильных и неправильных ответов. Снизу есть кнопка «Вернуться к списку» и надпись

«Пройти еще раз». Посли же окончания самого урока, например, «Творительный падеж» появляется окно, где подводятся результаты по всех частям урока, выводится информация об оценке, которую получил за этот урок пользователь, также есть оценка сложности урока и визуализация уровней сложности



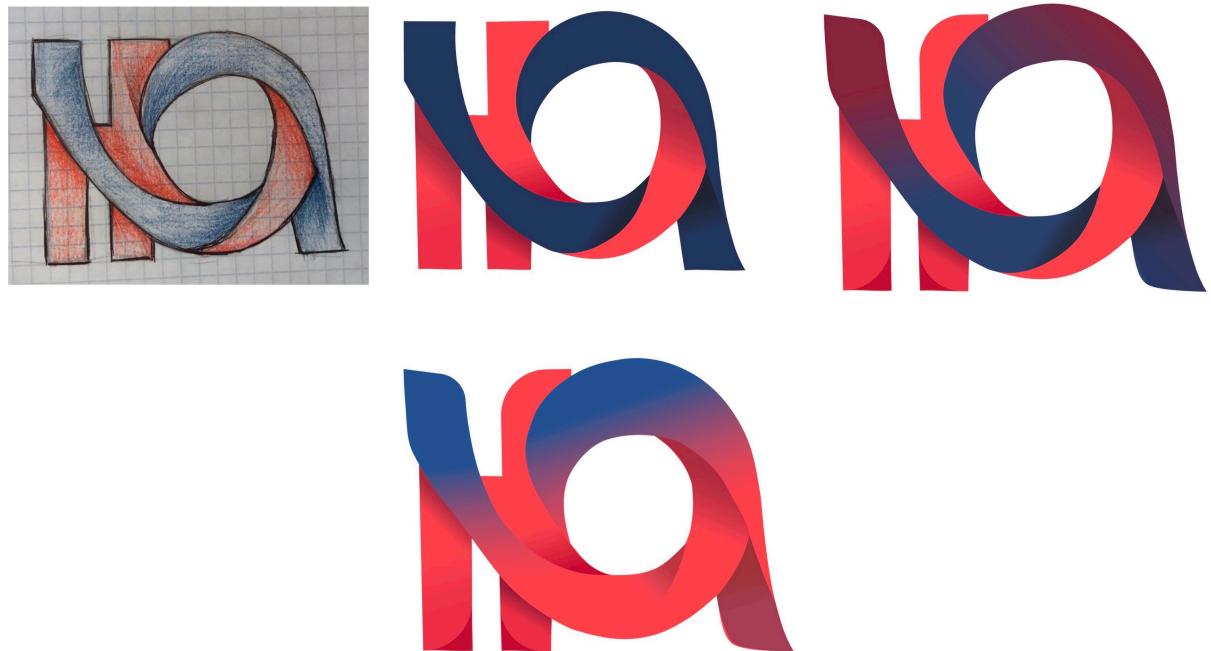
Окна окончания части урока и самого урока в целом

Теперь про логотип. Из-за смены названия приложения, нужно было изменить и логотип. Ниже представлен старый вариант логотипа



Старый логотип

Далее будут представлены этапы создания логотипа от наброска до конечного результата



Этапы создания логотипа

Данный логотип отражает название приложения «Юна», т.к. содержит в себе все эти буквы. Цвета были выбраны под основные цвета приложений, так же основой был градиент, как и во всем приложении в целом

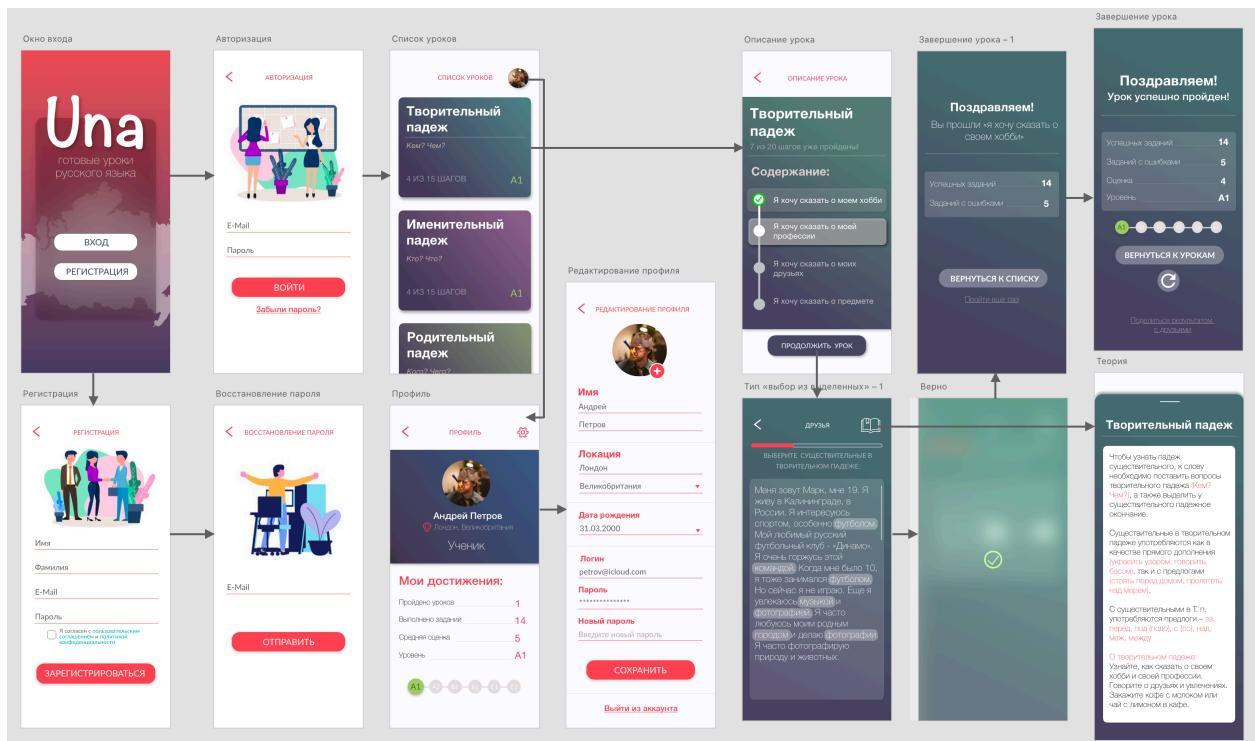


Задумка логотипа

По итогу, ниже представлена вся схема работы приложения (взаимодействия окон), сначала с неисправлена версией, а затем с исправленной



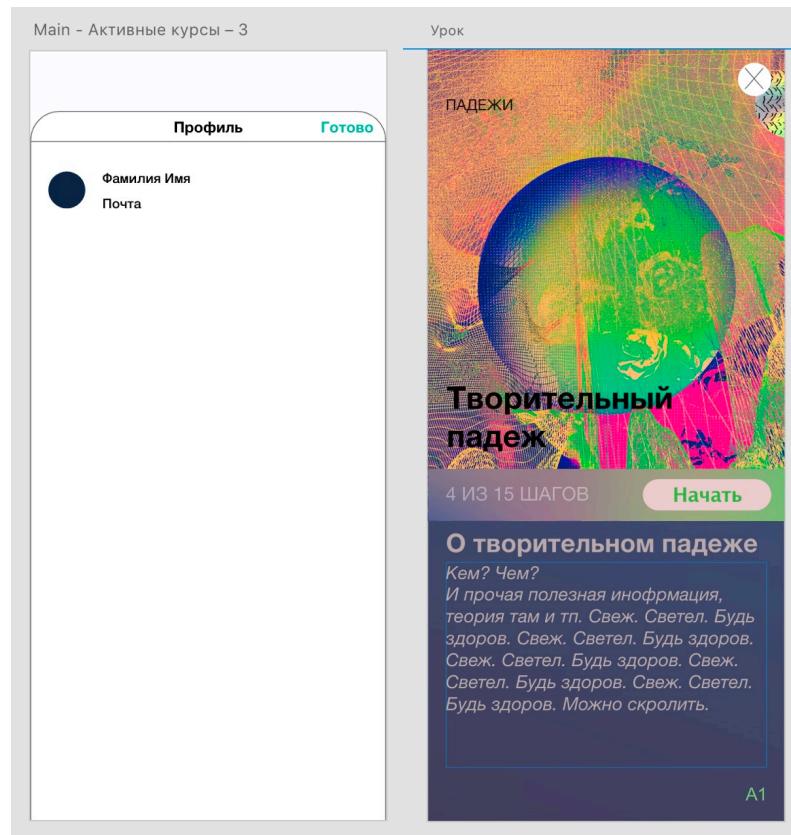
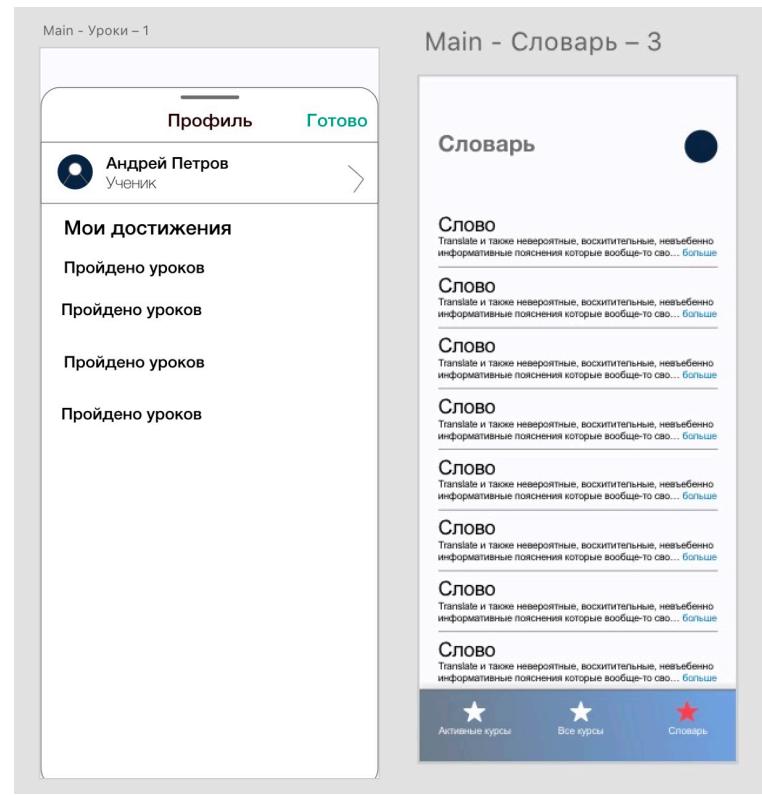
Схема работы приложения

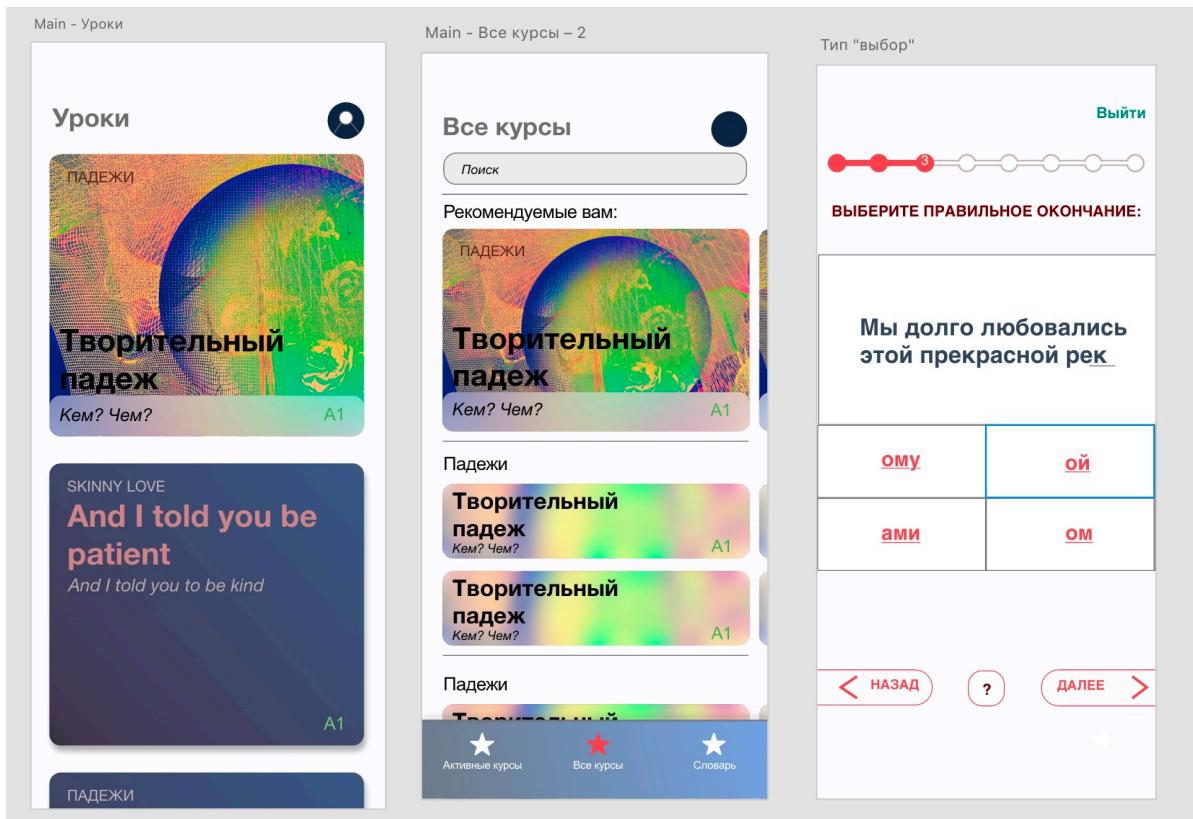


Описание последовательности окон:

- Окно входа
- Авторизация, которая подразделяется на:
 - Окно регистрации
 - Окно восстановления пароля
- Затем идёт список уроков, в котором выведены все доступные пользователю уроки для прохождения
- Откроется окно с информацией о данном уроке, после чего можно нажать на нужную часть урока и кнопку «Продолжить урок»
- Откроются сами задания
- После свайпа показывается информация о том, верно или нет было решено задания
- После прохождения части урока открывается окно с результатами
- После нажатия на кнопку возвращаемся в окно «Описание урока»
- Нажимаем на кнопку «Продолжить урок»
- Так продолжать до того момента, как не будет пройден весь урок
- При необходимости может быть открыта теория, которую можно убрать свайпом вниз
- После прохождения всего урока открывается информация о его прохождении в целом
- После этого можно вернуться на главный экран списка уроков или перепройти данный урок еще раз

АРХИВ ПРОТОТИПОВ





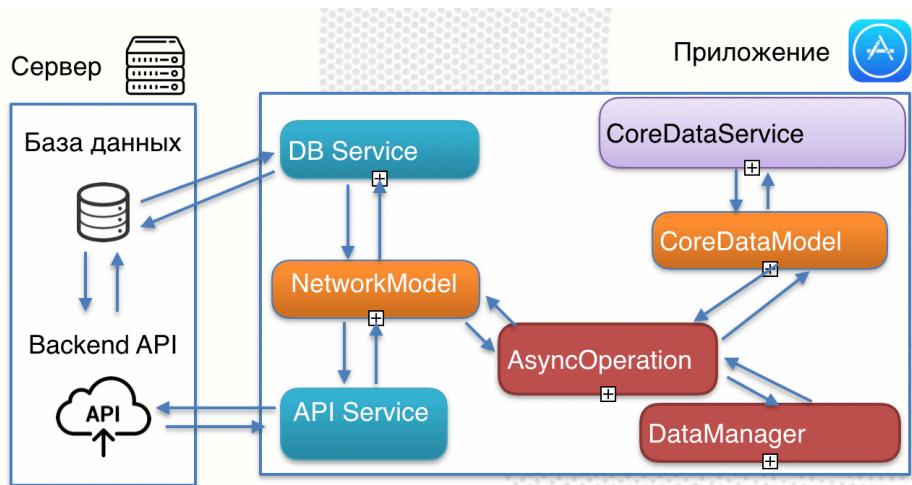
ТЕХНИЧЕСКАЯ РЕАЛИЗАЦИЯ ПРОДУКТА

В качестве децентрализованного хранилища исходного кода был использован Git, а в качестве удаленного репозитория - GitHub

Мобильное приложение реализовано на базе операционной системы iOS на языке Swift 5.0. Структура проекта имеет свою целостность и единый стиль. Весь код написан по принципам ООП и архитектурному паттерну MVP. При реализации интерфейса был использован AutoLayout для адаптивной верстки приложения на различные устройства iPhone.

Для загрузки данных по API с сервера Una создан сервис APIService с поддержкой фреймворка Mouya

Для загрузки данных из базы данных Una создан сервис DBService с поддержкой фреймворка PostgresClientKit



Архитектурная схема обработки данных до DataManager

Для кэширования и работы данных через CoreData созданы сервисы CoreData

Для асинхронной обработки данных во всем приложении созданы AsyncOperation'ы

Для удобной работы с данными созданы DataManager'ы

При разработке использовались новые системные библиотеки, такие как:

- CoreData - база данных для сохранения постоянных данных приложения в автономном режиме и кэширования временных данных
- Grand Central Dispatch - технология, предназначенная для управления многопоточностью

Помимо этого, использовались новые сторонние библиотеки:

- Moya - для упрощенной работы с RESTfull API
- PostgresClientKit - для работы с SQL-запросами к Postgres базам данных (в данном случае БД Юны)

ЗАКЛЮЧЕНИЕ

В результате проделанной работы было разработано мобильное приложение на операционной системе iOS для сервиса обучения русскому языку как иностранному Una. Кроме того, в ходе создания данного приложения были получены практические навыки в разработке и изучены множество инструментов для создания мобильных приложений.

В рамках данной работы были получены следующие результаты:

- Было разработано множество вариантов макетов для приложения
- Был изменен основной дизайн
- Было сформировано основное ядро работы с данными во всем приложении
- Заглушки, реализованные через Mock-Server, были заменены на реально существующие база данных и API сервиса Una
- Было настроено кэширование данных
- Была настроена работа с многопоточностью; работа с данными разбита по потокам
- Приложение было переписано на новый язык Swift 5.0 и добавлена поддержка iOS 13
- Была перестроена архитектура приложения на MVP (Model-View-Presenter)
- Макеты приложения были проверены на функциональность на реальном устройстве и при помощи сторонних пользователей
- Научились работать с OBS и Final Cut Pro

И, в заключение хотелось бы отметить, что мобильные приложения сегодня не могут пробиться и завоевать наибольшую популярность у потребителей на одной лишь идее. Необходима не только продуманная стратегия продвижения, но и, что еще важно, грамотная реализация. Да, с использованием в совокупности различных PR-технологий мобильная программа может достичь настоящего успеха, но грамотная реализация поможет остаться идее релевантной, на плаву на долгое время. И даже для брендовых приложений пусть и является простым привлечение покупателей за счет узнаваемости своего бренда, но удерживать их интерес они не могут только лишь применяя грамотную стратегию, но и годами полируя свою идею до блеска. Если продолжать грамотно продумывать данное мобильное приложение, то мы сможем помочь все большему количеству людей, которые хотят изучить русский язык.

СПИСОК ИСТОЧНИКОВ

1. Акулов, И.Б. SwiftBook [Электронный ресурс] / И.Б. Акулов. — Электрон. текстовые дан. — Режим доступа: <https://swiftbook.ru/contents/doc/>, свободный. — Русская документация Apple Swift Programming Language
2. GeekUniversity iOS [Электронный ресурс] /. — Электрон. текстовые дан. — Режим доступа: https://geekbrains.ru/geek_university/ios. — Онлайн-университет iOS-разработки
3. Core Data + Swift для самых маленьких: необходимый минимум [Электронный ресурс] / angryscorp. — Электрон. текстовые дан. — 2016. — Режим доступа: <https://habr.com/ru/post/258953/>, свободный
4. Core Data в деталях [Электронный ресурс] / IrixV. — Электрон. текстовые дан. — 2019. — Режим доступа: <https://habr.com/ru/post/436510/>, свободный
5. Human Interface Guidelines [Электронный ресурс] / Apple Inc.. — Электрон. текстовые дан. — Режим доступа: <https://developer.apple.com/design/human-interface-guidelines/ios/overview/themes/>, свободный
6. Структура iOS-проекта [Электронный ресурс] / GeekBrains. — Электрон. текстовые дан. — Режим доступа: <https://uploads.hb.cldmail.ru/asset/1140133/attachment/3ee6f7229be0cd14a234e7892ada285e.pdf>
7. UIKit [Электронный ресурс] / Apple Inc.. — Электрон. текстовые дан. — Режим доступа: <https://developer.apple.com/documentation/uikit>, свободный
8. Foundation | Apple Developer Documentation [Электронный ресурс] / Apple Inc.. — Электрон. текстовые дан. — Режим доступа: <https://developer.apple.com/documentation/foundation>, свободный
9. Акулов, И.Б. Туториалы - SwiftBook [Электронный ресурс] / И.Б. Акулов. — Электрон. текстовые дан. — Режим доступа: <https://swiftbook.ru/tutorials/>, свободный
10. Swift | Apple Developer Documentation [Электронный ресурс] / Apple Inc.. — Электрон. текстовые дан. — Режим доступа: <https://developer.apple.com/documentation/swift>, свободный
11. Не заставляйте меня думать /Стив Круг; пер. с англ. М.А. Райтмана. — 3-е. — Москва: Издательство "Э", 2018. — 256с.

12. Stack Overflow [Электронный ресурс] /. — Электрон. текстовые дан. — Режим доступа: <https://stackoverflow.com>, свободный. — Создание сообществ, где каждый, кто кодирует, может учиться и делиться своими знаниями
13. Perfect - REST сервер на Swift [Электронный ресурс] / Demtriy. — Электрон. текстовые дан. — habr, 2016. — Режим доступа: <https://habr.com/ru/post/283260/>, свободный
14. Better Routing for IOS Applications with Router in Swift [Электронный ресурс] / Seyhun AKYÜREK. — Электрон. текстовые дан. — Medium, 2016. — Режим доступа: <https://medium.com/swift-programming/better-routing-for-ios-applications-with-router-in-swift-9b47cc8cb4c7>, свободный
15. Погружение в паттерны проектирование /Александр Швец. — 2-е изд., электрон. — Refactoring Guru, 2018.
16. Pro Git /Scott Chacon, Ben Straub. — 2-е изд. — Apress, 2014. — 419с.
17. Манифест Чистого Программиста или краткий конспект книги «Чистый Код» Роберта Мартина [Электронный ресурс] / tanchuev. — Электрон. журн. — habr, 2018. — Режим доступа: <https://habr.com/ru/post/424051/>
18. Нормализация отношений. Шесть нормальных форм [Электронный ресурс] / DevilAngel. — Электрон. журн. — habr, 2015. — Режим доступа: <https://habr.com/ru/post/254773/>, свободный — пер. у Vincent Driessens
19. Moya — как перестать беспокоиться о сетевой части и начать жить [Электронный ресурс] / svyat_reshetnikov. — Электрон. текстовые дан. — habr, 2017. — Режим доступа: <https://habr.com/ru/post/332570/>, свободный
20.  How To Record A Discord Call Using OBS - For FREE [Электронный ресурс] / Dusty Porter. — Электрон. журн. — YouTube, 2018. — Режим доступа: <https://www.youtube.com/watch?v=QJyUW3C1tV0>, свободный
21. How To Export to MP4 MPEG4 in Final Cut Pro | FCPX | 10.3 and above [Электронный ресурс] / Steve Burnside, MBA. — Электрон. журн. — YouTube, 2017. — Режим доступа: <https://www.youtube.com/watch?v=d3K7gvUO5Mg>, свободный
22. Импорт и экспорт видео в Final Cut Pro X [Электронный ресурс] / Apple Siberia, — Электрон. журн. — YouTube, 2017 — Режим доступа: <https://www.youtube.com/watch?v=23GFOE7YP7w>, свободный

Монтаж видео в FCPX. Как плавно ускорить и замедлить видео в Final Cut Pro X? [Электронный ресурс] / Tech Union, — Электрон. журн. — YouTube, 2016 — Режим доступа: <https://www.youtube.com/watch?v=QrEoBV2HgT4>, свободный

23. Final Cut Pro X - Как обрезать видео [Электронный ресурс] / Монтаж и заработка, — Электрон. журн. — YouTube, 2017 — Режим доступа: <https://www.youtube.com/watch?v=kRJ8Kf91fAg>, свободный
24. Архитектурные паттерны в iOS [Электронный ресурс] / КАППАЛОТ. — Электрон. текстовые дан. — habr, 2016. — Режим доступа: <https://habr.com/ru/company/badoo/blog/281162/>, свободный
25. Self-sizing Table View Cells [Электронный ресурс] / Kevin Colligan. — Электрон. текстовые дан. — raywenderlich, 2018. — Режим доступа: <https://www.raywenderlich.com/8549-self-sizing-table-view-cells>, свободный
26. How to convert a hex color to a UIColor [Электронный ресурс] / Paul Hudson. — Электрон. текстовые дан. — hacking with swift, 2019. — Режим доступа: <https://www.hackingwithswift.com/example-code/uicolor/how-to-convert-a-hex-color-to-a-uicolor>, свободный
27. How To Delete Every Record Of A Core Data Entity [Электронный ресурс] / Bart Jacobs. — Электрон. текстовые дан. — cocoa casts, 2016. — Режим доступа: <https://cocoacasts.com/how-to-delete-every-record-of-a-core-data-entity>, свободный
28. Hashable [Электронный ресурс] / Apple Inc.. — Электрон. текстовые дан. — Режим доступа: <https://developer.apple.com/documentation/swift/hashable>, свободный
29. Core Data with Swift 4 for Beginners [Электронный ресурс] / Shashikant Jagtap. — Электрон. текстовые дан. — Medium, 2017. — Режим доступа: <https://medium.com/xcblog/core-data-with-swift-4-for-beginners-1fc067cca707>, свободный
30. The power of switch statements in Swift [Электронный ресурс] / Shashikant Jagtap. — Электрон. текстовые дан. — Swift by Sundell, 2017. — Режим доступа: <https://www.swiftbysundell.com/articles/the-power-of-switch-statements-in-swift/>, свободный
31. iOS 8's built-in virtual keyboards on the iPhone: A visual catalog [Электронный ресурс] / JOEY DEVILLA. — Электрон. текстовые дан. — global nerdy, 2015. — Режим доступа: <http://www.globalnerdy.com/2015/05/04/ios-8s-built-in-virtual-keyboards-on-the-iphone-a-visual-catalog/>, свободный
32. @IBDesignable and @IBInspectable in Swift 3 [Электронный ресурс] / ANANTHA KRISHNAN K G. — Электрон. текстовые дан. — Medium, 2016. — Режим доступа:

- <https://medium.com/anantha-krishnan-k-g/ibdesignable-and-ibinspectable-in-swift-3-702d7dd00ca>, свободный
33. How to use IBInspectable to adjust values in Interface Builder [Электронный ресурс] / Paul Hudson. — Электрон. текстовые дан. — Hacking With Swift, 2019. — Режим доступа: <https://www.hackingwithswift.com/example-code/uikit/how-to-use-ibinspectable-to-adjust-values-in-interface-builder>, свободный
34. IBInspectable / IBDesignable [Электронный ресурс] / Nate Cook. — Электрон. текстовые дан. — NSHipster, 2015. — Режим доступа: <https://nshipster.com/ibinspectable-ibdesignable/>, свободный
35. Building your Custom IBDesignable Controls and Views [Электронный ресурс] / Admin. — Электрон. текстовые дан. — digital leavews, 2015. — Режим доступа: <https://digitalleaves.com/building-your-custom-ibdesignable-controls-and-views/>, свободный
36. Working With IBDesignable [Электронный ресурс] / John Marstall. — Электрон. текстовые дан. — Medium, 2015. — Режим доступа: <https://medium.com/bpxl-craft/working-with-ibdesignable-807739869b36>, свободный
37. How to Create Round Buttons Using @IBDesignable on iOS 12 [Электронный ресурс] / DESIGN. — Электрон. текстовые дан. — Super Easy Apps, 2018. — Режим доступа: [https://blog.supereeasyapps.com/how-to-create-round-buttons-using-ibdesignable-on-ios-11/](https://blog.supereasyapps.com/how-to-create-round-buttons-using-ibdesignable-on-ios-11/), свободный
38. Магия IBDesignable или расширяем функциональность Interface Builder в Xcode [Электронный ресурс] / dante_photo. — Электрон. текстовые дан. — Medium, 2016. — Режим доступа: <https://habr.com/ru/post/274687/>, свободный
39. Fetching Remote Data With Core Data Background Context in iOS App [Электронный ресурс] / Alfian Losari. — Электрон. текстовые дан. — Medium, 2018. — Режим доступа: <https://medium.com/swift2go/fetching-remote-data-with-core-data-background-context-in-ios-app-224dad15ef6c>, свободный
40. API на Swift за пять минут. Лекция в Яндексе [Электронный ресурс] / Leono. — Электрон. текстовые дан. — Habr, 2017. — Режим доступа: <https://habr.com/ru/company/yandex/blog/339572/>, свободный
41. Token Based Authentication for Django Rest Framework [Электронный ресурс] / Shubham Bansal. — Электрон. текстовые дан. — Medium, 2018. — Режим доступа: <https://medium.com/quick-code/token-based-authentication-for-django-rest-framework-44586a9a56fb>, свободный

42. Getting Started with Moya [Электронный ресурс] / Malcolm Kumwenda. — Электрон. текстовые дан. — Medium, 2018. — Режим доступа: <https://medium.com/flawless-app-stories/getting-started-with-moya-f559c406e990>, свободный
43. Абстракция сетевого слоя с применением «стратегий» [Электронный ресурс] / iWheelBuy. — Электрон. текстовые дан. — Medium, 2017. — Режим доступа: <https://habr.com/ru/post/338380/>, свободный
44. The Mysterious Case of the Status Bar [Электронный ресурс] / craiggrummitt. — Электрон. текстовые дан. — Medium, 2018. — Режим доступа: <https://medium.com/@craiggrummitt/the-mysterious-case-of-the-status-bar-d9059a327c97>, свободный
45. How to: Create a UIPageViewController [Электронный ресурс] / Elliott Diaz. — Электрон. текстовые дан. — Medium, 2017. — Режим доступа: <https://medium.com/how-to-swift/how-to-create-a-uipageviewcontroller-a948047fb6af>, свободный
46. Disable “Dark Mode” in iOS 13 for your iOS App [Электронный ресурс] / akshay somkuwar. — Электрон. текстовые дан. — Medium, 2019. — Режим доступа: <https://medium.com/@akshay.s.somkuwar/disable-dark-mode-in-ios-13-for-your-ios-app-c025b446d87b>, свободный
47. Dark Mode: Adding support to your app in Swift [Электронный ресурс] / SwiftLee. — Электрон. текстовые дан. — AvanderLee, 2019. — Режим доступа: <https://www.avanderlee.com/swift/dark-mode-support-ios/>, свободный
48. Swift : Factory Pattern [Электронный ресурс] / Jay Mayu. — Электрон. текстовые дан. — Medium, 2018. — Режим доступа: <https://medium.com/@mayooresan/swift-factory-pattern-6f0d7556d862>, свободный
49. How to Find a Substring in a String with Swift [Электронный ресурс] / matt_eaton. — Электрон. текстовые дан. — Agnostic Development, 2017. — Режим доступа: <https://www.agnosticev.com/content/how-find-substring-string-swift>, свободный
50. Изменение коммитов в Git [Электронный ресурс] / limonte. — Электрон. текстовые дан. — Habr, 2013. — Режим доступа: <https://m.habr.com/ru/post/201922/>, свободный