МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»**
Кафедра математической кибернетики и компьютерных наук

СОЗДАНИЕ МОБИЛЬНОГО ПРИЛОЖЕНИЯ С ПОМОЩЬЮ ЯЗЫКА SWIFT

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

Студентки 4 курса 411 группы
направления 02.03.02 — Фундаментальная информатика и информационные технологии
факультета КНиИТ
Панченко Анастасии Ильиничны

Научный руководитель
доцент, к. ф.-м. н. _____ А. С. Иванов

Заведующий кафедрой
к. ф.-м. н. _____ А. С. Иванов

Саратов 2020

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В современном мире всё больше растёт потребность в совершенствовании мобильных технологий и программного обеспечения. Потребность в общении с людьми является движущей силой к новым разработкам и совершенствованию уже имеющихся. Общение в многочисленных социальных сетях является виртуальным общением, а в последнее время у людей возникает интерес в личном, живом общении, основанном, например, на общих интересах, увлечениях и хобби.

Поскольку тема «живого общения» в последнее время становится всё более популярной, пришла мысль разработать мобильное приложение для поиска людей по интересам, в котором делается упор на общение в реальной жизни.

Актуальность работы заключается в том, что пользователи мобильного приложения смогут осуществить возможность найти единомышленников и установить контакты в сфере своих увлечений, назначая встречи в «живом пространстве», что вполне в духе реального времени, когда люди устали от виртуального мира и стремятся к получению неповторимой атмосферы реального общения.

Целью настоящей работы является создание iOS приложения, предназначенного для установления и развития контактов между пользователями в соответствии с их интересами и увлечениями в реальном пространстве.

Для достижения цели должны быть решены следующие задачи:

- Изучить язык Swift.
- Ознакомиться с инструментами разработки для iOS приложений.
- Ознакомиться с библиотекой UIKit.
- Разработать iOS-приложение для встречи и общения людей в реальной жизни.

Работа состоит из введения, двух глав, заключения, списка из 20 источников и трёх приложений.

В главе «Описание используемых средств и технологий» описывается основная идея приложения и его минимальный функционал. Рассматривается выбор ОС для приложения и выбор облачной базы данных, описываются особенности языка Swift. Рассматривается платформа Firebase для разработки мобильных приложений, библиотека для создания интерфейса UIKit и про-

грамма Interface Builder.

В главе «Разработка мобильного приложения» рассматривается внутренняя структура программы, объясняется работа наиболее важных классов. Демонстрируется хранение зарегистрированных пользователей в Firebase, описывается структура хранения данных в базе. Также описывается и иллюстрируется подробное взаимодействие пользователя с интерфейсом приложения.

1 Описание используемых средств и технологий

В данном разделе рассмотрим платформу для разработки мобильных приложений Firebase и программу для создания интерфейса Interface Builder.

1.1 Firebase

В данной работе используются следующие составляющие Firebase:

— Служба авторизации Firebase Authentication.
— Хранение данных в Cloud Firestore [1].
— Хранение изображений в Cloud Storage.

Функциональные составляющие Firebase позволяют программе получать быстрый доступ к данным в реальном времени без потери скорости соединения и обмена информацией и при любом изменении синхронизировать и отправлять на подключенное устройство обновления за доли секунды [2].

В данной работе для реализации чата в реальном времени и карты с отображением меток в реальном времени используются асинхронные слушатели для отслеживания обновления данных, хранящихся в коллекциях и документах.

Асинхронный слушатель базы данных сообщает пользователю о хранящихся данных. Этот привязанный слушатель активируется один раз в начале для первичного набора данных и потом срабатывает каждый раз при изменении данных [3].

При использовании Firebase можно отметить такие положительные моменты как:

— Простое получение и чтение данных;
— Использование базы в реальном времени;
— Несложное подключение и использование SDK;
— Подробная и понятная официальная документация;
— Мгновенное обновление на подключённом устройстве при изменении данных.

Всё это делает Firebase удобным для использования при разработке приложения.

1.2 Interface Builder

Interface Builder — это приложение от компании Apple, содержащее в себе весь набор инструментов для создания графических интерфейсов в составе

IDE Xcode [4].

Interface Builder значительно облегчает разработку пользовательского интерфейса. Используя его, можно создать различные элементы управления без написания кода, например:

— кнопки, ползунки, текстовые поля и т.д.;
— варианты различных экранов;
— создание связей и переходов между экранами и т.д.

К плюсам Interface Builder можно отнести то, что при использовании этого сервиса можно наглядно соединять и располагать составные элементы в программе, устанавливать стили, шрифты и т.д., не прописывая код для каждого элемента на экране. Interface Builder облегчает работу с табличными типами и ячейками таблиц. Можно создавать таблицы практически полностью без написания кода. Благодаря всему этому можно очень легко и быстро создать желаемый интерфейс приложения [5].

Однако есть и минусы. Возможности Interface Builder не безграничны, и если интерфейс приложения становится слишком сложным, то его придётся доделывать в коде. Так, например, не получится реализовать анимацию с помощью этой программы. Также разработка интерфейса данным способом подходит только для индивидуальных разработчиков, командная работа в этом сервисе становится крайне затруднительной и проблемной.

В программе используются мобильные инструменты предварительной визуализации — сториборды. Они имеют ряд достоинств:

— полностью виден сценарий разрабатываемого приложения и взаимосвязи между его экранами;
— сториборды могут описывать переходы между различными окнами (переходы называются segues [6]), которые создаются путём соединения двух экранов прямо в сториборде. Благодаря segues, разработчик пишет меньше кода для пользовательского интерфейса [7].

2 Разработка мобильного приложения

В качестве практического задания в данной работе было разработано мобильное приложение для iOS, предназначенное для установления и развития контактов между пользователями в соответствии с их интересами и увлечениями в «живом пространстве», а именно в рамках представленного города (Саратов).

В приложении существует панель вкладок с 3 основными экранами:

1. *Профиль* — основная информация о пользователе;
2. *Сообщения* — список диалогов с другими пользователями;
3. *Поиск друзей по городу* — отображение меток пользователей на карте города в реальном времени.

При нажатии на метку возможно два действия:

1. Просмотр профиля пользователя;
2. Переход на экран чата с выбранным человеком.

Более подробно интерфейс будет рассмотрен в разделе «Взаимодействие с приложением».

Сначала рассмотрим внутреннее строение данного мобильного приложения.

2.1 Приложение с точки зрения программиста

Рассмотрим более подробно внутреннюю структуру программы.

Файлы программы имеют два формата: .swift и .storyboard. Swift-файлы содержат исходный код, написанный языке с одноимённым названием. Сториборд — это визуальное представление пользовательского интерфейса приложения, отображающее экраны и связи между этими экранами. Сториборд позволяет подключать вид экрана (View) к контроллеру вида (View Controller), а также управлять передачей данных между контроллерами вида [8].

Перечислим все файлы с форматом .storyboard:

1. *Entry* — представляет экраны входа и регистрации.
2. *TabBar* — видовое представление панели с тремя вкладками, которая имеет связи с другими сторибордами в виде ссылок.
3. *DialogList* — файл, который содержит экран со списком диалогов пользователей и экран чата, а также ссылку на экран просмотра профиля.
4. *UserProfile* — файл, который содержит экран профиля пользователя и

экран редактирования профиля.

5. *Map* — файл, который содержит экран с картой, экран создания метки на карте и экран просмотра профиля, а также ссылку на экран чата с конкретным пользователем.

Рассмотрим важные классы программы более подробно.

2.1.1 Database

Класс, содержащий основные функции для работы с базой данных. Перечислим основные методы класса:

— `createUser(_ email : String, _ password : String, completion : @escaping (Error?) -> Void)` — метод, который регистрирует нового пользователя в Firebase.

— `signIn(_ email : String, _ password : String, completion : @escaping (Error?) -> Void)` — метод авторизации пользователя.

— `signOut() -> String?` — метод для выхода пользователя из аккаунта.

— `writeProfileData(userId : String, userData : [String : Any], completion : @escaping (Error?) -> Void)` — метод записи данных профиля пользователя в базу.

— `readProfileData(userId : String, completion : @escaping (User?, Error?) -> Void)` — метод чтения данных профиля пользователя из базы. Данный метод имеет асинхронный слушатель, который срабатывает при изменении документа пользователя в базе.

— `downloadImage(urlString : String, completion : @escaping (UIImage?) -> Void)` — загрузка изображения пользователя из Cloud Storage.

— `sendPin(city : String, email : String, pin : [String : Any], completion: @escaping (Error?) -> Void)` — отправка информации о созданной метке в базу данных.

2.1.2 UserProfileViewController

Класс, управляющий экраном профиля пользователя. Перечислим основные методы класса:

— `viewDidLoad()` — метод, загружающий данные профиля пользователя.

— `editPhoto(_ sender: UIBarButtonItem)` — метод, позволяющий пользователю загрузить фотографию из галереи телефона. Данный метод срабатывает при нажатии иконки камеры в верхнем левом углу экрана профиля пользователя.
— `signOut(_ sender: Any)` — метод, позволяющий пользователю выйти из аккаунта. Данный метод срабатывает при нажатии иконки выхода в верхнем правом углу экрана профиля пользователя.

2.1.3 EditProfileViewController

Класс, управляющий экраном редактирования профиля. Методы класса:
— `updateUserData(_ sender: UIButton)` — метод, отправляющий изменённые данные профиля пользователя в базу. Данный метод срабатывает при нажатии кнопки «Сохранить» на экране редактирования профиля.
— `viewWillAppear(_ animated: Bool)` — метод, отображающий текущую информацию профиля в текстовых полях для редактирования.

2.1.4 DialogsViewController

Класс, управляющий экраном списка диалогов. Перечислим основные методы класса:
— `prepare(for segue: UIStoryboardSegue, sender: Any?)` — метод, передающий данные выбранного диалога на экран чата. Данный метод срабатывает при нажатии на выбранную ячейку из таблицы диалогов.
— `viewWillAppear(_ animated: Bool)` — метод, получающий данные о диалогах пользователя.
— `tableView(_ tableView: UITableView,`
`cellForRowAt indexPath: IndexPath) -> UITableViewCell` — метод, заполняющий ячейку таблицы диалогов данными.

2.1.5 Dialogs

Класс, загружающий диалоги пользователя из базы данных. Метод `loadDialogs(fromUser : String, completion : @escaping () -> Void)` загружает данные каждого диалога из базы. Данный метод имеет асинхронный слушатель, который срабатывает при изменении документов коллекции Dialogs в базе данных.

2.1.6 ChatViewController

Класс, управляющий экраном чата. Перечислим основные методы класса:

- `viewWillAppear(_ animated: Bool)` — метод, загружающий сообщения переписки на экран.
- `sendMessage(text : String, toUser : String, completion : @escaping (Error?) -> Void)` — метод, отправляющий сообщение пользователю в базу данных.
- `collectionView(_ collectionView: UICollectionView, cellForItemAt indexPath: IndexPath) -> UICollectionViewCell` — метод, который заполняет данными ячейку таблицы чата и задаёт внешний вид облака сообщения.

2.1.7 MapViewController

Класс, управляющий экраном с картой. Перечислим основные методы класса:

- `longPressGesture(_ sender: UILongPressGestureRecognizer)` — метод, ставящий на карту метку-указатель в указанной точке и отображающий адрес метки. Данный метод срабатывает при распознавании жеста длительного нажатия на карту.
- `viewWillAppear(_ animated: Bool)` — метод, загружающий метки пользователей на карту.
- `deleteWishPin(_ sender: UIBarButtonItem)` — метод удаления метки пользователя с карты. Данный метод срабатывает при нажатии на иконку удаления метки в верхнем правом углу экрана с картой.
- `mapView(_ mapView: MKMapView, didSelect view: MKAnnotationView)` — метод отображения информации о выбранной метке на карте во всплывающем окне. Данный метод срабатывает при нажатии на метку.
- `swipeDownAddress(gesture: UISwipeGestureRecognizer) -> Void` — метод, скрывающий всплывающие окна. Данный метод срабатывает при распознавании жеста смахивания вниз.
- `prepare(for segue: UIStoryboardSegue, sender: Any?)` — метод, передающий данные на экран создания метки, а также на экран просмотра

профиля выбранного пользователя. Данный метод срабатывает при нажатии кнопок «Создать метку» и «Посмотреть профиль» на экране с картой.

2.1.8 ActionListScreen

Класс, управляющий экраном создания метки на карте. Основные методы класса:

— `createPin(_ sender: UIButton)` — метод, создающий новую метку на карте. Данный метод срабатывает при нажатии кнопки «Создать» на экране создания метки.

— `tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell` — метод, который заполняет ячейки таблицы выбора формата встречи информацией на экране создания метки.

2.1.9 Структура базы данных приложения

Для хранения данных в Cloud Firestore используются документы и коллекции. Документ — это запись, содержащая поля. Документы хранятся в коллекции. Коллекция может хранить документы с разным набором значений. Также документ может содержать вложенные коллекции [9]. Рассмотрим подробнее структуру хранения данных для нашего мобильного приложения.

В базе данных хранится две коллекции:

1. *cities* — хранение информации о городе;
2. *userInfo* — хранение информации о зарегистрированных пользователях.

Рассмотрим каждую коллекцию подробнее. На данный момент в коллекции cities хранится один документ, который использует название города в качестве идентификатора документа.

Данный документ имеет вложенную коллекцию *MapPins*. В данной коллекции хранятся документы, которые содержат в себе информацию о добавленной на карту метке. Каждый документ использует email пользователя в качестве идентификатора. Документ содержит следующие поля:

— Населённый пункт и административный район заданной метки.

— Подробный адрес метки в рамках населённого пункта.

— Email пользователя.

— Имя.

— Фамилия.
— URL ссылка на изображение пользователя.
— Координаты метки.
— Желаемый формат встречи при знакомстве.

В коллекции *userInfo* хранятся документы, которые содержат информацию о зарегистрированном пользователе. Каждый документ использует email пользователя в качестве идентификатора. Документ содержит следующие поля:

— Имя.
— Фамилия.
— Город.
— Возраст.
— URL ссылка на изображение пользователя.
— Пол.
— Основная информация о пользователе.
— Хобби и интересы.
— Любимые книги.
— Любимые фильмы.
— Любимая музыка.

Также документ может иметь различное число вложенных коллекций. Вложенные коллекции с названием электронной почты хранят все сообщения переписки между текущим пользователем и пользователем с соответствующим названием коллекции. В таких коллекциях хранятся документы с заданными полями:

— Email отправителя сообщения.
— Email получателя сообщения.
— Текст сообщения.
— Возраст.
— Дата и время отправки сообщения.

Также есть второй вид вложенной коллекции, которая называется *Dialogs*. Она представляет собой заголовки диалогов и содержит документы с электронной почтой в качестве идентификатора. В каждом документе содежатся следующие поля:

— Email отправителя сообщения.

— Email получателя сообщения.
— Текст последнего сообщения.
— Дата и время отправки последнего сообщения.
— Имя и фамилия пользователя, с которым ведётся переписка.
— URL ссылка на изображение пользователя, с которым ведётся переписка.

2.2 Взаимодействие с приложением

В данном разделе опишем подробное взаимодействие пользователя с приложением.

При первом запуске приложения мы попадаем на экран входа. При нажатии на кнопку «Зарегистрироваться» появляется соответствующий экран, где предлагается ввести свой email и пароль.

После авторизации происходит переход на домашнюю страницу профиля. Здесь можно заполнить данные своего профиля, нажав на кнопку «Редактировать», а также загрузить фотографию при нажатии на иконку камеры в верхнем левом углу. После этого откроется галерея для выбора фотографии. Чтобы выйти из профиля, нужно нажать на иконку выхода, расположенную в правом верхнем углу экрана.

Нажимая на вкладки на панели внизу, мы можем переключаться на экраны профиля, сообщений и карты. При переходе на вкладку с изображением окна диалога мы попадём в список диалогов, где будут отображаться переписки с другими пользователями.

Чтобы начать общение, нужно перейти на вкладку карты. Вкладка представляет собой карту города, на которой отображаются метки других пользователей. При длительном нажатии на выбранное место карты на ней появляется метка с указателем, а также всплывает небольшое окно с адресом выбранной точки. Внизу данного окна можно нажать на кнопку «Поставить метку» для добавления своей метки на карту. После этого появится окно с таблицей, в которой перечислены различные варианты занятий при знакомстве. После выбора формата встречи нажимаем на кнопку «Создать», и попадаем обратно на карту, на которой теперь создана наша метка.

Если нам мешают эти небольшие всплывающие окна, их можно скрыть с помощью жеста, смахнув окно вниз. Если мы захотим поставить вторую метку, то сначала нужно будет удалить предыдущую, иначе программа выдаст сообщение с данным предупреждением, после чего вернёт нас на экран карты.

Для того чтобы удалить метку, нужно нажать на иконку перечёркнутой метки в правом вернем углу. После подтверждения удаления метка пользователя исчезнет с карты.

При нажатии на метку всплывает небольшое окно с информацией о пользователе: на нём отображается миниатюра фотографии, имя пользователя, желаемый формат встречи при знакомстве и адрес текущей метки. Также внизу этого окна можно нажать на кнопку «Посмотреть профиль», после чего мы попадём на экран просмотра профиля выбранного пользователя. При прокрутке экрана можно изучить профиль подробнее, или нажать на кнопку «Написать», чтобы перейти в чат с выбранным пользователем.

После отправки первого сообщения в чате можно выйти из вкладки карты и перейти на вкладку сообщений: на экране появится новый диалог с пользователем.

Если мы получим сообщение от другого человека, то диалог также создастся во вкладке сообщений. Нажав на выбранный диалог, мы можем перейти в чат и ответить на сообщение. Также в верхнем правом углу можно нажать на иконку профиля и перейти на страницу профиля пользователя.

ЗАКЛЮЧЕНИЕ

Целью данной работы было создание мобильного приложения для операционной системы iOS, предназначенного для установления и развития контактов между пользователями в соответствии с их интересами и увлечениями в «живом пространстве», а именно в рамках населенного пункта (г. Саратов).

В процессе написания работы были выполнены следующие задачи:

- Изучен язык Swift.
- Осуществлено ознакомление с инструментами разработки для iOS приложений.
- Осуществлено ознакомление с библиотекой UIKit.
- Разработано iOS-приложение для встречи и общения людей в реальной жизни.

Важным инструментом для разработки программы послужил язык программирования Swift. Коммуникативную основу мобильного приложения создала встроенная карта AppleMaps от компании Apple, на которой база данных Cloud Firestore может отображать данные в режиме реального времени. При создании модели приложения был использован облачный сервис — Firebase, сочетающий в себе такие функции как: служба аутентификации, база данных в реальном времени, хранение файлов. Благодаря библиотеке UIKit и программе InterfaceBuider был создан простой, красивый и удобный интерфейс приложения.

В результате было создано мобильное приложение с удобным и понятным пользовательским интерфейсом для встречи и общения людей в реальной жизни.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Cloud Firestore — Firebase. [Электронный ресурс]. — URL: https://firebase.google.com/docs/firestore?authuser=0 (Дата обращения 20.05.2020). Загл. с экр. Яз. рус.

2 Что такое Cloud Firestore? [Электронный ресурс]. — URL: https://ua-blog.com/\T2A\cyrch\T2A\cyrt\T2A\cyro-\T2A\cyrt\T2A\cyra\T2A\cyrk\T2A\cyro\T2A\cyre-cloud-firestore/ (Дата обращения 20.05.2020). Загл. с экр. Яз. рус.

3 Get realtime updates with Cloud Firestore. [Электронный ресурс]. — URL: https://firebase.google.com/docs/firestore/query-data/listen (Дата обращения 20.05.2020). Загл. с экр. Яз. рус.

4 *Шахова, И. С.* Практикум по курсу «Введение в разработку мобильных приложений». iOS / И. С. Шахова, И. Р. Залялов. — Казань: Казанский университет, 2019.

5 Знакомство с Interface Builder. [Электронный ресурс]. — URL: https://habr.com/ru/post/30553/ (Дата обращения 20.05.2020). Загл. с экр. Яз. рус.

6 Особенность Segue в Swift [Электронный ресурс]. — URL: http://bzhizn3.beget.tech/osobennost-segue-v-swift/ (Дата обращения 21.05.2020). Загл. с экр. Яз. рус.

7 iOS Storyboards: анализ плюсов и минусов. [Электронный ресурс]. — URL: https://habr.com/ru/company/mobileup/blog/456086/ (Дата обращения 23.05.2020). Загл. с экр. Яз. рус.

8 Проектирование с помощью сторибордов. [Электронный ресурс]. — URL: https://habr.com/ru/post/152375/ (Дата обращения 20.05.2020). Загл. с экр. Яз. рус.

9 Cloud Firestore — это просто. [Электронный ресурс]. — URL: https://habr.com/ru/post/447640/ (Дата обращения 21.05.2020). Загл. с экр. Яз. рус.