Д. В. Назаров
(ГГУ имени Ф. Скорины, Гомель)
Науч. рук. **А. А. Зайцев**, ст. преподаватель

РАЗРАБОТКА ПРИЛОЖЕНИЯ-МЕССЕНДЖЕРА ДЛЯ ИСПОЛЬЗОВАНИЯ ОРГАНИЗАЦИЯМИ

Проблематика. С учетом современных пандемических реалий перевода сотрудников на удаленную работу, многие, в том числе государственные, организации встают перед вопросом – где сотрудники будут общаться по служебным вопросам?

Самым распространенным ответом является «Slack», однако и это решение имеет ряд проблем.

В 2020м году Slack занимает достаточно большую долю рынка корпоративных мессенджеров, однако всё больше подвергается критике за то, что превращается в подобие Skype for Business. Функционал постоянно растет, однако качество разработки мессенджера падает, дизайн становится всё более перегруженным и непонятным для только что пришедших людей, а новыми функциями никто не пользуется, предпочитая Slack только для текстового общения. В связи с этим, а также высокой ценой, необходимо новое решение, которое будет обладать меньшим функционалом, однако будет дешевле и проще в использовании.

Цель работы. Создать мессенджер, который сможет заменить для организаций такие программные продукты как «Slack» или, например, «Microsoft Teams». Мессенджер будет иметь возможность регистрации и контроля пользователей, а также возможность создания групповых чатов. Получившийся продукт будет иметь открытый исходный код, а также распространяться по лицензии GPL v3, что даёт возможность организациям быстро модернизировать или создавать свои модули для него, и использовать мессенджер, не платя никаких денег создателю.

Для того чтобы создать мессенджер, проанализируем требования к нему:

– Основная функция – текстовое общения;
– Небольшие требования к ресурсам
– Возможность создавать групповые чаты
– Регистрация и контроль пользователей
– Сохранение списка чатов на устройстве

Реализация:

Сохранение сообщений на удаленном сервере. Для того чтобы сообщения появлялись при входе на любом корпоративном устройстве, необходимо хранить их на удаленном сервере. Это позволяет не беспокоиться о переносе данных в переписках при переходе на другое устройство, а значит легко их восстанавливать в случае эксцессов.

Если хранить переписки на собственных серверах для каждой компании – их обслуживание и поддержка обойдется очень дорого, поэтому сервера, так как и хранение информации, доверяют некоторым доверенным провайдерам, таким как, например, Google.

В данном приложении будет использоваться база данных от Firebase, так как Google является корпорацией, чьи сервера используют многие крупные компании, а так как как Google пока что только входит на рынок облачных сервисов, пытаясь потеснить таких гигантов как Microsoft и Amazon, компания очень сильно снижает цены, что повлияет на цену использования приложения, которую можно понизить.

Ещё одним преимуществом использования Firebase является то, что мы можем предложить функцию постмодерации и просмотра сообщений для наших клиентов, напрямую из базы данных.

В нашем случае база данных была спроектирована таким образом, что у каждого диалога есть свой уникальный идентификатор, по которому его могут найти другие участники. Также уникальный идентификатор есть и у каждого сообщения, по которому мы можем его найти. Также хранится время отправки сообщения и автор сообщения.

Сохранение диалогов на устройстве. Для корректной работы приложения нам необходимо хранить диалоги на устройстве, так как в ином случае пользователю придется при каждом использовании опять находить по id и добавляться в них.

Для того чтобы выбрать оптимальный метод хранения диалогов, нам необходимо рассмотреть существующие подходы к оффлайн-хранению информации на android:

– SharedPreferences – постоянное хранилище, используемое приложениями для хранения своих настроек, например. Обычно SharedPreferences и используется для того, чтобы сохранить немногочисленные настройки. Так как в нашем случае все настройки находятся в аккаунте, этот метод мы не используем

– SQLite – встраиваемая база данных с открытым исходным кодом, поддерживающая все возможности SQL. Является основной базой данных для android-приложений.

Так как хранить нам необходимо информацию о диалогах, число которых неограниченно, а число полей всегда одно и тоже, и они всегда заполнены – оптимальным методом хранения данных в данном случае является SQLite.Для работы с встраиваемой базой данных Google рекомендует использовать Room, что мы и будем делать.

Регистрация и вход пользователей. Для приложения необходима регистрация, так как это базовый механизм разграничения прав пользователей, также дающий им возможность показать свою личность используя некоторые функции приложения.

Firebase дает нам возможность регистрировать пользователей тремя разными способами:

– Регистрация через аккаунт Google

– Регистрация через электронную почту

– Регистрация через телефон

Также это дает нам преимущества легкой аутентификации пользователей, так как все их данные сохраняются непосредственно на сервере.

Проверка зарегистрирован ли пользователь производится в DialogsActivity, до того, как произведется рендер остальной части активити, что не дает возможности начать пользоваться мессенджером без регистрации.

DialogsActivity

При проектировании интерфейса использовались некоторые методики изученного пользовательского опыта. После изучения тепловых карт пользования экраном смартфонов, были приняты некоторые решения по части пользовательского опыта в дизайне. Кнопки, которые должны нажиматься часто должны располагаться внизу. Кнопки, которые должны нажиматься нечасто должны располагаться сверху.

Из-за этого, во время создания интерфейса экрана чатов, кнопки создания и присоединения к чатам были расположены сверху экрана – чтобы пользователю приходилось тянуться к ним, что не допускает случайных нажатий.

Кнопки создания и подключения чатов располагаются сверху, а диалоги располагаются под ними – из-за того, что к диалогам необходим постоянный доступ.

Стоит заметить, что список чатов содержит удобные цветовые метки, которые не отвлекают от сути, но позволяют чату отличаться от других.

Также в каждом чате можно увидеть идентификатор чата, который позволяет передать его другим сотрудникам, и дать им возможность присоединиться к чату.

Список чатов реализуется с помощью RecyclerView, и, чтобы его реализовать, мы должны создать следующие компоненты:

– RecyclerView, который мы должны добавить в layout нашего экрана;

– layout для каждой строки списка;

– адаптер, который содержит данные и связывает их со списком.

ChatActivity

В данном активити будет проходить большая часть времени использования приложения. Суть его состоит в том, что пользователю нужен экран, где было бы видно сообщения, отправленные в чате. Дополнительным требованием является то, что нужно видеть не только сообщения, помещающиеся на одном экране. Также сообщения должны обновляться в режиме реального времени у всех участников беседы, одновременно. Для удобства пользователей были придуманы некоторые особенности, улучшающие удобство использования приложения:

– показ времени отсылки сообщения;

– показ отправителя для каждого из сообщений;

– кнопка быстрого выхода;

– показ идентификатора чата;

– показ имени чата.

Показ информации для юзера, такой как название чата, мы осуществляем с помощью обыкновенных textView. Информацию для этих полей мы передаем в данное активити с помощью механизма Extras, котоырй позволяет передавать параметры в открываемое с помощью Intent активити.

Как мы можем видеть, в данном активити тоже необходим некий список сообщений, который необходимо реализовать через адаптер.

Для решения этой задачи был использован компонент, который бы расставлял сообщения по разным сторонам. Для такой задачи использовался ListView, с адаптером FirebaseListAdapter.

В обычном режиме, все элементы view, которое выступает элементом списка, имеют одинаковые свойства во всем, кроме текста, однако в нашем случае, необходимо было модифицировать их так, чтобы у разных отправителей сообщения показывались с разной стороны. Это достигается путем того, что мы меняем свойство Gravity у view в том случае, когда автор сообщения – непосредственно пользо-

ватель данного инстанса приложения. FirebaseListAdapter с этой модификацией появляется в методе displayChat.

А. Д. Нечай
(ГГУ имени Ф. Скорины, Гомель)
Науч. рук. **Е. А. Ружицкая**, канд. физ.-мат. наук, доцент

РАЗРАБОТКА WEB-ПРИЛОЖЕНИЯ ПО ПРОДАЖЕ КОМПЛЕКТУЮЩИХ ДЛЯ ПЕРСОНАЛЬНОГО КОМПЬЮТЕРА

У каждого человека рано или поздно возникает потребность в новом компьютере. Это может происходить по нескольким причинам: компьютер устарел в плане своих технических возможностей в связи с постоянным развитием технологий и увеличение требований различных приложений к аппаратным вычислительным скоростям ПК или получить первый опыт пользования компьютером. Большинство покупателей выбирает именно ноутбуки, т. к. они не требуют настройки, сборки и проверки работы. При этом, стационарные компьютеры в большинстве случаев будут дешевле и производительнее. Разработанное приложение призвано устранить данную проблему.

Посетитель сайта имеет возможность ознакомиться с каталогом магазина, ценами на комплектующие и другие оказываемые услуги. Можно оставить свой контактный номер телефона с целью получения консультации или оформления заказа.

Пользователь приложения имеет возможность выбрать определенный набор комплектующих и сделать заказ. Сборкой компьютера занимается компания. Этот вариант подойдет для более продвинутых пользователей. Для обычного покупателя, который не интересуется в текущих трендах в мире комплектующих, сделать выбор в сторону определенных компонентов системы вызывает определенные трудности. В этом случае, компания может разместить на сайте уже готовые решения, на которые можно очень быстро оформить заказ без каких-либо проблем.

В приложении реализована возможность просмотра познавательных и актуальных роликов из различных ресурсов на различного рода темы, связанные с новинками информационных технологий. Так же есть возможность просмотра последних новостей в мире компьютерных технологий.