Федеральное государственное образовательное бюджетное учреждение
высшего образования
**«Финансовый университет при Правительстве
Российской Федерации»**

Департамент анализа данных и машинного обучения

Курсовая работа по дисциплине
«Технологии разработки приложений для мобильных устройств»
на тему:

«Мобильное приложение для создания шаблонов электронных писем»

Выполнила:
Студентка ПИ20-6
Бадмаева Б. Б.

Научный руководитель:
к.т.н., доцент,
Крахмалев О. Н.

Москва
2022

Оглавление

ВВЕДЕНИЕ

В современном мире, для любого человека, предпринимателя, организации очень важно поддерживать связь друг с другом. В наше время функционирует огромное число мессенджеров и социальных сетей, но при этом, до сих пор пользуется спросом электронная почта. Пусть первые и выиграли в соревновании за более удобное общение, но кто бы что ни говорил, почта все ещё необходима человеку.

В первое время электронная почта тоже использовалась для коммуникации и общения. Одно из главных ее преимуществ, открытый домен, позволяет отправлять сообщения между ящиками разных почтовых сервисов, например с *@gmail.com* на *@yandex.ru*, в то время как пользователь *Twitter* не сможет найти и связаться на своей платформе с человеком, зарегистрированным только в *Facebook*. Благодаря этому, можно без проблем установить связь со всеми, кто имеет электронный почтовый ящик.

С течением времени, общение между людьми плавно перетекло в социальные сети и мессенджеры, а почта стала использоваться больше в рабочих целях. Большая часть документов, теперь отправляется и хранится на почтовых ящиках: чеки, билеты, счета, справки. Передача важной информации для сотрудников организации, учащихся учебного заведения, пользователей какой-либо платформы практически всегда производится с помощью электронных писем. Все же, невозможно представить мир без существования электронной почты.

Таким образом, мы убеждаемся, что применение сервисов электронной почты и по сей день актуально. Для облегчения данной задачи используют различные конструкторы электронных писем, которые позволяют сокращать время, затрачиваемое на процесс создания сообщения. Такие приложения бывают очень удобны и полезны для многих пользователей, и особенно для тех, кто использует сервисы электронной почты на постоянной основе. Целью данной работы и будет разработка такого конструктора писем.

ПОСТАНОВКА ЗАДАЧИ

Реализация проекта будет осуществляться с использованием фреймворка для разработки кроссплатформенных приложений *React Native* и платформы *Expo Snack*, которая позволит сделать процесс разработки намного быстрее и удобнее.

Главная цель данного проекта – разработка мобильного приложения для создания шаблонов электронных писем. Для достижения цели нужно будет реализовать несколько задач: создание необходимых компонентов и процессов для функционирования приложения, разработка нескольких экранов проекта, навигации между ними, дизайна и расположения элементов интерфейса для взаимодействия с пользователем. Также, для правильного построения шаблонов нужно определить стандартную структуру электронного сообщения. Используя типовое строение письма, необходимо будет реализовать требуемые алгоритмы для обработки данных, структурирования их в соответствующем порядке.

Таким образом, будем считать, что задача проекта – создание программы, состоящей из собственно разработанных компонентов, нескольких экранов навигации, элементов дизайна и сочетающей в себе различные функции, позволяющие конструировать шаблоны электронных писем, выстроенные по стандартной заданной структуре.

ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

Исследуемой предметной областью в данном проекте выступает электронное письмо. Разрабатываемое приложение будет работать непосредственно с конструкцией такого письма. Определение стандартной структуры сообщений, выявление общего шаблона электронных посланий необходимо для разработки алгоритмов, формирующих реализуемую программу.

Непосредственно в сервисе электронной почты, в соответствующих полях заполняются такие данные, как электронный адрес получателя, тема письма, прикреплённые файлы и т. д. Наша область обработки – тело письма, т. е. его основное содержание. В общих чертах, этот раздел содержит вступление, основную часть и заключение. Вступление подразумевает краткое обоснование причины обращения и часто выступает как отклик на предыдущее полученное письмо (подтверждение получения, благодарность за ответ). В основной части подробнее раскрывается тема письма, описываются главные вопросы и положения, а в заключении могут располагаться выводы, возможные решения, просьбы. Будем использовать общую схему с небольшими дополнениями: отдельно вынесем приветствие, включающее в себя обращение и представление, и прощание, которые может содержать какие-либо формулы вежливости, подпись. Тогда, наша структура будет выглядеть следующим образом:

1) Приветствие
2) Вступление
3) Основная часть
4) Заключение
5) Прощание

Каждый из элементов принятой структуры будет называться разделом, а цифра, под которой он указан – кодом раздела. Реализуем возможность заполнения не всех разделов, ведь это не всегда обязательно или необходимо.

Для создания шаблонов будут созданы два варианта их формирования. Первый заключается в конструировании сообщения с помощью выбора одной фразы из списка предложенных для каждого раздела. То есть, будет разработан список стандартных фраз для приветствия, и пользователь должен будет выбрать среди них один вариант, аналогично и для других разделов. Второй способ будет предполагать ввод пользователем его собственных ответов для

каждого раздела. Так, если клиент не найдет подходящих ему выражений среди предложенных, он сможет указать свои в окне с формами ввода.

АКТУАЛЬНОСТЬ АВТОМАТИЗАЦИИ

Как и говорилось ранее, несмотря на существование большого количества социальных сетей и мессенджеров, электронная почта также остается важным инструментом коммуникации для простых людей, предпринимателей или компаний. В мире каждую минуту отправляется огромное число электронных писем. Их использование обусловлено удобством использования сервисов электронной почты, например, для отправки сообщений совсем не обязательна регистрация на какой-либо единой платформе, ведь пользователи могут общаться друг с другом даже имея разные почтовые домены. Огромное количество информации передается и хранится на почтовых ящиках, потому будет актуальным разработка программы, направленной на работу с электронными письмами, особенно на уменьшение времени при их написании.

Приложение, позволяющее создавать шаблоны электронных писем, дает возможность оперативно формировать сообщения, не затрачивая много времени на то, чтобы придумать, построить и структурировать послание.

РАЗРАБОТКА ТЕХНИЧЕСКОГО ЗАДАНИЯ

Разрабатываемое приложение направлено на работу с электронными письмами, а именно создание их шаблонов. Данная программа предназначена для активных пользователей электронной почты и помогает быстро сформировать сообщение, предлагая клиенту готовую структуру письма, подсказки по составлению, распространенные стандартные выражения на выбор.

Для разработки приложения будет использоваться React Native – фреймворк на основе JavaScript и платформа для разработки Expo Snack.

Благодаря использованию единой кодовой базы React Native позволяет создавать программы сразу для нескольких платформ, тогда мы сможем протестировать приложение и на iOS, и на Android. Expo же представляет собой набор инструментов, библиотек и сервисов, позволяющих просматривать и тестировать проекты прямо во время разработки, что очень упрощает и ускоряет работу над приложением. Во время редактирования кода в веб-браузере, можно одновременно транслировать его в Web или на виртуальном устройстве iOS или Android, а также на собственных девайсах. Также, Expo Snack удобен тем, что в нем не нужно устанавливать какие-либо зависимости, а достаточно добавить их в файл *package.json*.

Для данного проекта можно выделить следующие требования и критерии.

Требования, в соответствии с которыми реализуется проект:

- Разработка нескольких экранов приложения
- Реализация навигации между экранами
- Разработка оригинального дизайна пользовательского интерфейса
- Преобразование текста в готовый шаблон письма

Критерии – показатели, по которым оценивается результат работы:

- Реализация навигации с корректной передачей параметров при необходимости
- Обработка текста в соответствии со структурой письма
- Приложение полезное и интуитивно понятное для пользователя

СОЗДАНИЕ АЛГОРИТМА

Для удобства необходимо организовать простейшую структуру директории проекта. Приложение будет иметь несколько экранов, которые будут храниться в отдельной папке *src*/*screens*/. Переходы между ними будут осуществляться с помощью навигации стека, для этого создадим папку

src/*navigations*/ и в ней файл навигации *AppNavigator.js*. А используемые изображения будут находиться в *src*/*assets*/.

Для создания шаблона электронного письма будут разработаны два способа конструирования: настройка содержания сообщения из предложенных фраз или формирование из своих вариантов. Для обоих методов будет применяться организация структуры письма по выявленной типовой схеме. Для того чтобы получить из набора выбранных или введенных фраз корректный шаблон, будет выполняться структурирование по коду раздела в возрастающем порядке. Он будет передаваться при обновлении используемых данных с каждой формы раздела, то есть, в методе *onPress* при переключении между вариантами выбора и в методе *onChange* при вводе своих ответов.

При переходе на результирующий экран будет выполняться функция подготовки данных к отправке на страницу формирования шаблона, и результат этой функции будет передаваться как параметр. На последнем экране будет приниматься этот параметр, конструироваться в послание и выводиться в специальную форму.

Таким образом, приложение будет состоять из 4 экранов: начальная страница Start.js, окно с выбором фраз из списка Choice.js, окно с вводом данных от пользователя Input.js и страница с результатом Result.js.

В общих чертах, алгоритм будет следующий:

- при запуске приложения пользователь попадает на главный экран, делает выбор между двумя вариантами конструирования сообщения
- на экране с предпочтительной формой составления текста, клиент начинает создавать свой шаблон, выбирая или вводя каждый фрагмент структуры электронного письма
- при любом изменении каждого элемента структуры данные обновляются

- при нажатии на кнопку перехода на следующий экран запускается работа функции подготовки данных и результат отправляется как параметр
- данные из параметра принимаются на последней странице и структурируются по общей заданной схеме электронного письма, формируется шаблон
- на выходе пользователь получает модернизированный текст – готовый шаблон электронного письма, который он может скопировать и вставить в при создании сообщения в своём сервисе электронной почты

ОПИСАНИЕ АЛГОРИТМА

Навигация

В файл AppNavigator.js необходимо импортировать {*CreateAppContainer*} из библиотеки ‘react-navigation’, куда будет помещаться созданный навигатор, а также функцию {*createStackNavigator*} из ‘react-navigation-stack’, использующуюся для формирования навигации стека. Кроме того, должны быть импортированы все экраны приложения, между которыми планируется производить переходы, а также их местоположение в файловой системе.

Навигатор представляет собой стек, где каждый новый экран помещается в самое начало. Первым экраном будет указан модуль Start, специально созданный как начальное окно, и по умолчанию, он будет открываться первым. Далее, при переходах на другие экраны, каждый будет помещаться в начало очереди, а предыдущий спускаться на уровень. Благодаря данному методу порядка реализуется и переход на предыдущий, первоначальный экраны.

Навигатор стека *AppNavigator создаётся* с помощью функции *createStackNavigator*(), при этом необходимо указать заголовки экранов и какой именно модуль будет отображаться под этим заголовком. Также для шапки

приложения были указаны параметры дизайна по умолчанию в свойстве *defaultNavigationOptions*. Для возможности использования навигатора вне данного файла экспортируем созданный компонент, сохранив его в контейнер *AppContainer*.

Файл приложения

В файл App.js импортируется контейнер с навигацией, React для создания пользовательских компонентов, которые затем будут преобразовываться в собственные компоненты для отрисовки на каждой платформе, а также методы для подключения шрифтов из ‘expo-font’ и экран Error, который будет использоваться при ошибке их загрузки.

В начале в классе App подключаются все шрифты, которые будут использоваться в приложении. Для этого помещаем в переменную state начальное состояние *isFontLoaded: false*. В теле функции *componentDidMount*() необходимые шрифты загружаются асинхронно и состояние обновляется. При отрисовке компонентов интерфейса в операторе *return* проверяется, были ли загружены шрифты, и если так, используется импортированный контейнер *AppContainer*, а иначе будет отображаться экран ошибки.

Далее всю работу будет выполнять созданный контейнер приложения, в котором указаны все необходимые экраны и разработана навигация между ними. Компонент *App* в конце также экспортируется, как и все экраны приложения.

Начальный экран

В соответствии с выбранным в стеке навигации порядком, контейнер приложения первым запускает файл *Start.js*.

В код первым делом также импортируются необходимые модули: *React* из библиотеки ‘*react’*, компоненты *View, Text, StyleSheet, Image, TouchableOpacity* из ‘*react-native’*, а также иконки из '*@expo/vector-icons'*. На начальном экране будут располагаться две кнопки для перехода на начало

работы с шаблоном, созданные с помощью *TouchableOpacity*. Первая кнопка «Выбрать из предложенных» перемещает пользователя на экран с выбором данных из стандартных фраз, а вторая - «Создать свой шаблон» производит переход на страницу с формами ввода. Свойство *onPress* отслеживает событие нажатия на кнопку. Пока экран включен в стек навигации, он автоматически наследует полезные свойства объекта navigation, тогда, для каждой кнопки используем в *onPress* функцию *navigate()* указав в скобках нужную страницу для перехода. На одну кнопку добавим иконку[1], просто поместив ее внутри тега *TouchableOpacity*. Иконки будут браться из ресурса Expo, в котором при выборе изображения можно получить необходимый код для импорта в файл и для тега (пример на рис. 1).

Рис. 1

Вариант создания с выбором

На данном экране для создания шаблона пользователю будут представлены на выбор готовые фразы по соответствующим разделам. В код программы после импорта необходимых компонентов будут добавлены списки с возможными фразами для каждой структурной части письма. Для каждого из пяти разделов была создана пока пустая строковая переменная, в которую будет сохраняться текущее выбранное значение с кодом раздела для дальнейшей обработки.

[1] https://icons.expo.fyi/ - ресурс с иконками от Expo

Реализация вывода списка вариантов для каждой структурной части будет осуществляться с помощью формы переключателя *RadioForm* библиотеки *'react-native-simple-radio-button'*. Данный метод в отличие от чек-бокса позволяет выбрать только один элемент из каждого блока, что упрощает проверку корректности структуры письма. Для возможности использования *RadioForm* списки разделов будут реализованы как массивы, состоящие из объектов со свойством *label* – текстовым содержанием, которое отрисовывается по ключу *value*.

В атрибуте *radio_props* компонента *RadioForm* мы указываем наши массивы для вывода списков, а в *initial* – значение по умолчанию, у нас это *false*, так как есть возможность использования не всех разделов письма. Для события *onPress*, которое отслеживает нажатие на переключатель будет выполняться присваивание нового значение соответствующей переменной раздела. При обновлении в переменную будет сохраняться строка состоящая из кода раздела и текстового значения выбранной кнопки, полученного по атрибуту *label*.

При нажатии на кнопку перехода на результирующий экран, будет запускаться работа функции *preparing*(), которая подготавливает данные к отправке на последнюю страницу. При выполнении функции каждый раз будет создаваться пустой массив *selected*, в который будут добавляться отобранные данные. Это сделано для того, чтобы пользователь имел возможность вернуться назад со страницы результата на экран формирования шаблона и составить сообщение выбрав другие данные. Иначе, если бы массив не очищался каждый раз при вызове функции, в нем бы оставались старые значения и новые добавлялись бы к уже существующему тексту. Далее создается массив *variables* в котором будут определены конечные значения переменных, в которых сохранялись изменения при переключении кнопок. Запускается цикл, который проходится по всем элементам массива переменных, проверяет, была ли вообще изменена переменная и добавляет в массив отобранных если это так.

Так как первоначальным значениям переменных разделов была пустая строка, достаточно проверить, больше ли нынешняя длина строки элемента чем 0. В конце с помощью оператора *return* будет возвращаться результат выполнения функции – массив *selected*.

Итак, в методе onPress кнопки перехода на экран с формированием шаблона в переменную result сохраняются обработанные данные и производится переход на следующий экран также с помощью *navigate*(), где в скобках указываются название необходимого модуля и параметры для отправки – результат функции *preparing*().

Также, на экране реализована кнопка для перехода на предыдущий экран, где в методе onPress используется функция навигации goBack().

Вариант создания с вводом

В данном окне пользователь будет создавать шаблоны с помощью собственных введённых данных. Клиенту будет представлена форма заполнения, поделённая на разделы, соответствующие установленной нами структуре электронного письма. Тут также для возможности использования каких-либо компонентов или модулей необходимо указать их в *import*. Для каждого раздела указывается заголовок и добавляется краткое описание и примеры. Так, пользователь быстро пойдёт что к чему, и сможет составить сообщение намного быстрее и правильнее, чем в обычных условиях.

Здесь также будут созданы пустые переменные для каждого раздела, в которые в дальнейшем будут сохраняться любые изменения. Форма заполнения для каждой секции создаётся с помощью компонента *TextInput*. Он имеет атрибут *placeholder*, значение которого будет отображаться внутри поля заполнения, мы расположим тут примеры фраз, которые подойдут для каждого раздела письма. Установим в *multiline* значение *true* для возможности ввода длинных значений и перевода текста на новую строку. Функция *onChangeText*()

отслеживает изменения введённых данных. Так, при обновлении состояния, в соответствующие переменные будут сохраняться код раздела и новый текст.

Формы выбора данных на экранах *Choice* и *Input* отличаются, но функционал практически одинаков. И там, и там отслеживаются события нажатия или изменения и любое новое состояние в каждой секции письма записывается в переменную вместе с кодом раздела.

На данном экране в конце также располагаются кнопки с возвратом на предыдущую страницу и кнопка перехода на окно результата с абсолютно такими же функциями в методах onPress, что и в файле Choice.js. Так что, опустим описание их содержания и работы.

Окно результата

После импорта используемых методов, в классовом компоненте с помощью функции *getParam()* принимаются данные отправленные с предыдущего экрана. Для формирования письма создадим пустую строковую переменную *sample*, в которую будем пошагово добавлять каждую обработанную фразу. Создаем цикл, который будет проходиться по всем элементам массива из данных, выбранных в предшествующем окне. К строке каждый раз с новой строки будет добавляться элемент отсортированного с помощью функции sort() массива. Функция substr() будет обрезать каждую фразу, удаляя первый символ, который служил кодом раздела.

Готовый шаблон выводится в созданном для него контейнере, а по кнопке «Скопировать» полученный текст копируется в буфер обмена с помощью функции *copy()*, использующей API Clipboard и его метод *setString().* Также, пользователю отправляется уведомление об успешном копировании с помощью Alert.

ДИЗАЙН ИНТЕРФЕЙСА

Стили заголовков в данном проекте прописаны в файле навигации, в переменной *stackNavigatorOptions*. Параметр *headerShown* отвечает за видимость шапки со стороны пользователя, *headerTintColor* устанавливает цвет текста заголовка, а в *headerStyle* определим цвет фона.

Стили для каждого экрана будут прописываться в созданном с помощью компонента *StyleSheet* и функции *create()* объекте *styles*. Для прикрепления стиля к определенному элементу интерфейса, необходимо в открывающем теге этого компонента указать соответствующую ссылку на элемент созданной таблицы стилей *styles* в атрибуте *style*.

Start.js

Начальный экран Start.js содержит GIF-изображение, приветственное сообщение пользователю и кнопки для переходов на другие страницы.

Для начала был создан основной контейнер компонент, позволяющий прокручивать экран вверх или вниз. В нем будут размещены все используемые компоненты, а примененные стили будут растягиваться на весь экран. Стиль будет прописан в элементе *styles.container*. В нем будет задаваться цвет фона, а благодаря установленному значению *'center'* в свойстве *justifyContent*, элементы блока будут располагаться по центру по вертикали. Атрибут *flex* определяет, как будет заполняться доступное пространство относительно других элементов. Установлено значение 1 для того, чтобы контейнер и его стиль занимал весь экран.

Также был создан еще один контейнер с функциональными элементами. В его стиле *styles.functional* добавим отступ снизу *marginBottom* и центрирование всех элементов по горизонтали, установив в *alignItems значение 'center*'. Такое же оформление основного и функционального контейнера будет и на экране Result.js.

Расположим на экране *GIF*-изображение с помощью тега *Image* со ссылкой на файл в атрибуте *source*. В стиле *styles.gif* для должного отображения *GIF*-изображения укажем для него ширину и высоту, а также необходимые отступы. Для настройки приветственного сообщения, созданного с помощью компонента *Text* в *styles.paragraph* определим желаемый шрифт, его размер, толщину и цвет, добавим выравнивание по центру, отступ и максимальную ширину надписи.

Стилизация у обеих кнопок TouchableOpacity будет общая. В стиле *styles.btnContainer* в свойстве *flexDirection* выберем значение '*row'* для того, чтобы элементы внутри кнопки выводились по горизонтали, так как для одной из кнопок будет добавляться иконка. Отрисуем границы кнопки указав их радиус углов, ширину и цвет, выровняем внутренние части по горизонтали и вертикали. Также укажем цвет кнопки, размеры (ширину и высоту) и отступы. Стиль текста кнопки укажем в *styles.btnText,* установим цвет, толщину, размер и семейство шрифта. Данная настройка кнопок будет использоваться на всех остальных экранах.

Также расположим в конце экрана небольшую подпись, содержащую иконку копирайт и название приложения. Цвет и размер знака указываются в теге иконки, а стиль для надписи поместим в *styles.sign,* где укажем цвет текста, размер и шрифт. Также, в *styles.signContainer* настроим контейнер подписи, указав в *flexDirection* значение '*row',* для отображения иконки и надписи в строку, выровняв по центру по горизонтали в *justifyContent* и поместив в конец экрана с помощью значения 'flex-start' в свойстве alignItems.

Choice.js

На данную страницу в формы переключателя будут помещены списки фраз для разных разделов. Так как каждая подпись элемента переключателя будет отображаться с новой строки, *ScrollView* – идеальный вариант для главного контейнера. Для каждого раздела письма и переключателя *RadioForm* сформирован отдельный контейнер *View*.

Некоторые стили форм переключателя *RadioForm* прописываются в свойствах тега. Настроим в компоненте такие атрибуты как размер и цвет кнопки, а в *labelStyle* поместим ссылку на элемент *label* таблицы стилей *styles*, где укажем шрифт, размер и цвет текста, отступ, а также ширину надписи, чтобы слишком длинные фразы не обрезались, а переносились на новую строку.

В *styles.container* для *ScrollView* определим интересующий цвет фона, выравнивание *alignItems* и также установим в свойстве *flex* значение 1. Контейнер будет занимать все доступное пространство и его стиль в итоге будет отображен на всей странице даже при прокручивании. Кнопки тут настроены по такому же принципу, что и на предыдущем экране, но здесь их следует поместить в контейнеры *View*, и в *styles.btnView* добавить центрирование по горизонтали.

В *styles.sectionHeader* будем описывать оформление заголовка раздела (компонента *Text*). Каждый новый раздел будет отрисовываться на определенном расстоянии от предыдущего с помощью свойства *marginBottom*. Настроив цвет в *backgroundColor*, он будет выводиться на фоне текста вдоль всей строки, образуя интерфейс как у настоящего раздела. Также установим необходимый шрифт и его размер.

Оформление подписи приложения внизу экрана используем то же, что и на начальном экране, только добавим в *styles.signContainer* отступы и настроим отображение в правом нижнем углу, а не по середине. И с такими же изменениями подпись добавлена и на экраны *Input.js* и *Result.js*.

Input.js

Дизайн данного экрана очень схож с предыдущим, ведь в нем выполняется тот же процесс, но другим способом.

Основной контейнер здесь тоже *ScrollView*, а шаблон письма аналогично формируется по разделам, тогда оформление их заголовков и контейнера с

прокруткой позаимствуем из файла *Choice.js*, изменив только цвет в *styles.sectionHeader*.

Сверху экрана у нас располагается надпись, стиль *styles.paragraph* был использован тот же, что и у приветственного сообщения первого экрана *Start.j*s, с небольшими изменениями в отступах. Для оформления *TextInput* в *styles.inputConteiner* установим шрифт, отступы со всех сторон, высоту формы ввода, толщину и радиус скругления границ, а также цвет для вводимого текста. Для каждого раздела здесь добавляется короткое описание в стиле *style.de*scription которого настроим шрифт, размер текста и отступ. Кнопки оформлены также, как в файле *Choice.js*

Result.js

На экране с результатом работы будут располагаться: изображение, надпись, готовый получившийся шаблон электронного письма, кнопка для копирования сообщения, кнопки для возвращения на предыдущий и главный экраны. Стили для основного и функционального контейнеров View, надписи и кнопок будут заимствованы из файла *Start.js*. Для оформления контейнера шаблона укажем в *styles.result* цвета для заднего фона и границ контейнера, также добавим толщину и радиус скругления границ, отступы, выравнивание по центру и размер. Для текста шаблона в *styles.resultText* установим шрифт, размер и цвет текста.

Error.js

Данный экран не включен в стек навигации и будет показываться только при ошибке загрузки шрифтов. Дизайн контейнера View будет использоваться тот же, что и у главных контейнеров всех экранов навигации. Внутри расположим иконку ошибки и соответствующее сообщение.

РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

Назначение

Приложение направлено на быстрое создание шаблона электронного письма. Было разработано два варианта создания сообщения: первый заключается в отборе пользователем фраз из списка предложенных и второй предполагает ввод собственных. На основе выбранных данных и установленной структуре письма подготавливается и формируется шаблон.

Для пользователя не нужно иметь каких-либо специфических знаний или подготовки. Приложение простое и интуитивно понятное, кнопки имеют соответствующие подписи, а в окне с формами ввода своих данных указываются подсказки и примеры возможных вариантов.

Запуск приложения

Для запуска приложения на мобильном устройстве должно быть установлена программа Expo Go, которая позволяет открывать проекты, разработанные с помощью Expo. Для старта необходимо отсканировать QR-код на рис.2

Рис. 2

Также, можно протестировать программу без установки каких-либо дополнительных инструментов. Для этого нужно перейти по ссылке[2] и открыть проект в браузере, где программа будет постоянно отображаться справа в режиме *web*. Выше будут располагаться кнопки для запуска приложения на

[2] https://snack.expo.dev/@aveabina/bsamples - ссылка на открытие проекта в браузере

виртуальных устройствах *iOS* и *Android*, а выбрав *My Device*, можно получить тот самый QR-код для открытия на своем устройстве.

Рекомендации к применению

После запуска приложения открывается начальная страница, которая содержит приветственное сообщение и кнопки для переходов на окна создания шаблонов. Нажав на кнопку «Выбрать из предложенных», пользователь попадает на экран, где ему предстоит сформировать шаблон из готовых выражений, которые будут отображены на данной странице. Из каждой секции необходимо будет отметить по одной фразе, которая наиболее предпочтительна для клиента, но не обязательно включать все разделы в письмо. Если же была выбрана кнопка «Создать свой шаблон», будет запущено окно с формой ввода данных для создания собственного шаблона. Форма также поделена на разделы, соответствующие стандартной структуре письма, в которые пользователь должен будет внести свои варианты ответов. Также, ему будут предложены подсказки и примеры фраз, которые можно использовать для определенной структурной части. Здесь также нет строгого правила заполнения всех существующих секций. На каждом из экранов создания шаблона находятся кнопка для возврата на предыдущую страницу и кнопка для перехода на страницу с результатом.

При открытии последнего окна на экране отображается обработанный подготовленный текст – готовый шаблон, который пользователь может скопировать и вставить в свой почтовый сервис. Также, тут находятся кнопки для возврата на страницу редактирования шаблона и на начальный экран.

ТЕСТИРОВАНИЕ ПРИЛОЖЕНИЯ

Тестирование на собственном мобильном устройстве iOS представлено на рисунках 3-7:

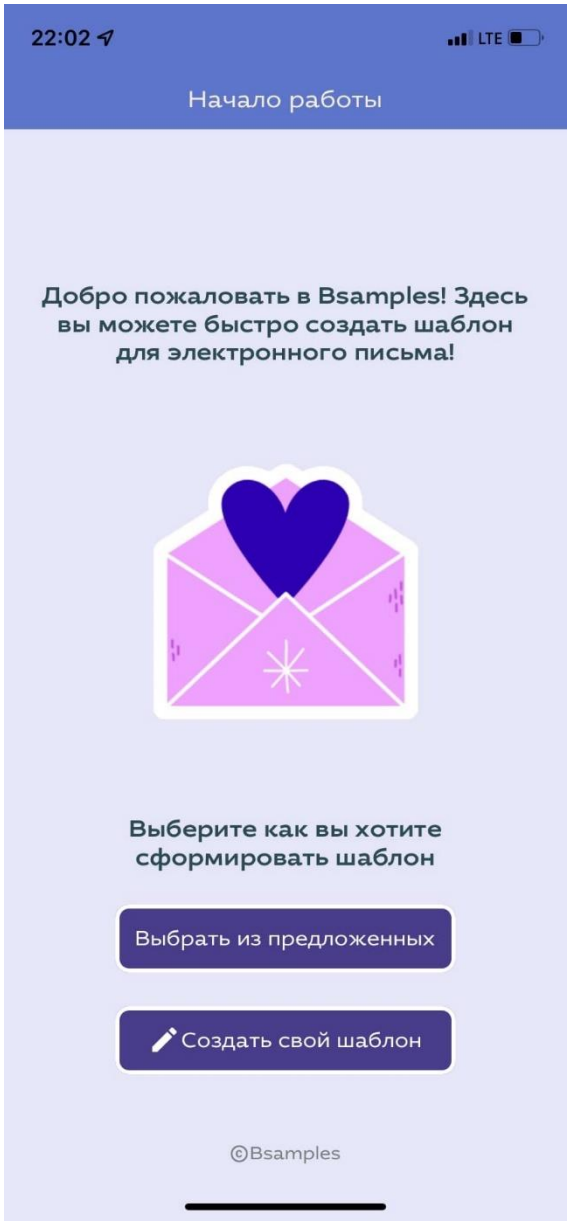
22:02
LTE
Начало работы
Добро пожаловать в Bsamples! Здесь вы можете быстро создать шаблон для электронного письма!
Выберите как вы хотите сформировать шаблон
Выбрать из предложенных
Создать свой шаблон
©Bsamples

Рис. 3

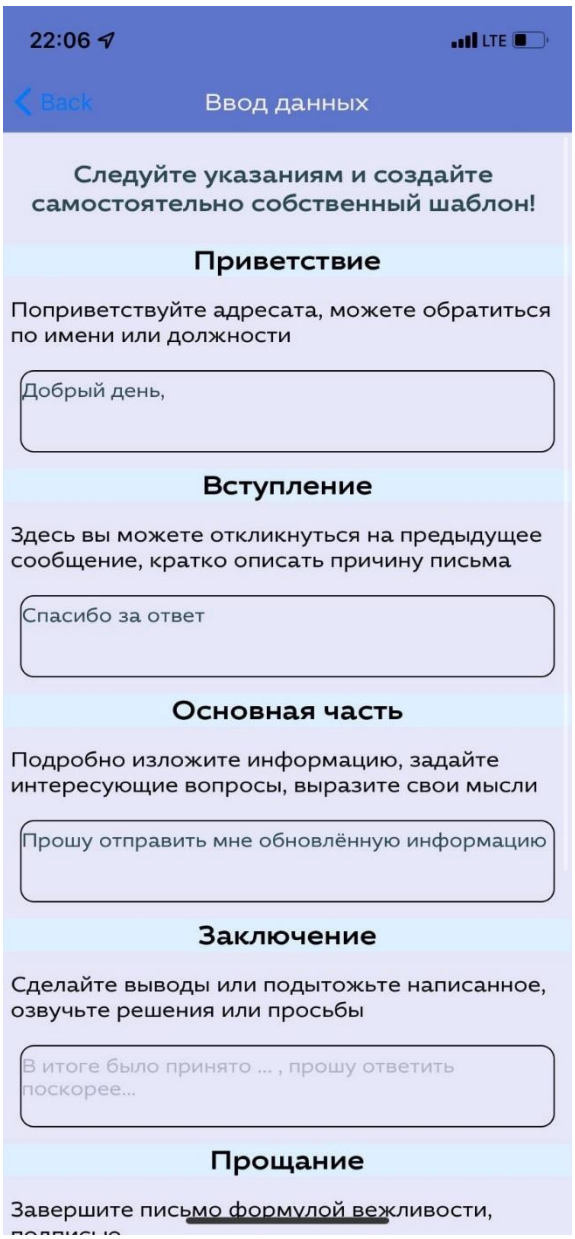
22:06
LTE
Back
Ввод данных
Следуйте указаниям и создайте самостоятельно собственный шаблон!
Приветствие
Поприветствуйте адресата, можете обратиться по имени или должности
Добрый день,
Вступление
Здесь вы можете откликнуться на предыдущее сообщение, кратко описать причину письма
Спасибо за ответ
Основная часть
Подробно изложите информацию, задайте интересующие вопросы, выразите свои мысли
Прошу отправить мне обновлённую информацию
Заключение
Сделайте выводы или подытожьте написанное, озвучьте решения или просьбы
В итоге было принято ... , прошу ответить поскорее...
Прощание
Завершите письмо формулой вежливости,

Рис. 4

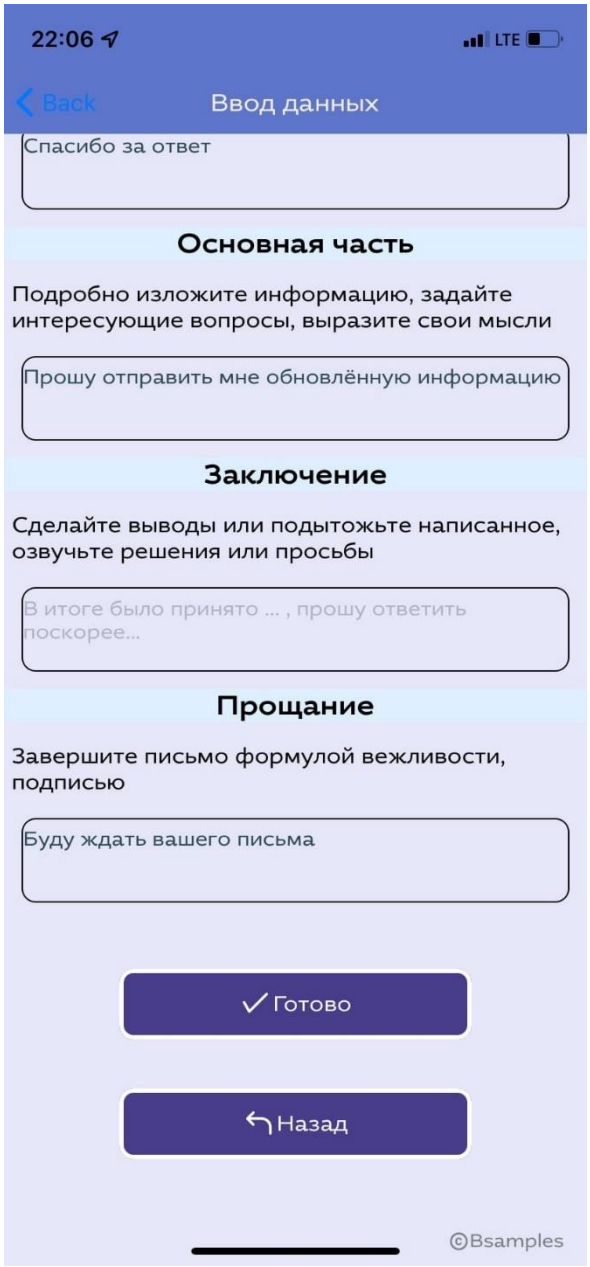
22:06
LTE
Back
Ввод данных
Спасибо за ответ
Основная часть
Подробно изложите информацию, задайте интересующие вопросы, выразите свои мысли
Прошу отправить мне обновлённую информацию
Заключение
Сделайте выводы или подытожьте написанное, озвучьте решения или просьбы
В итоге было принято ... , прошу ответить поскорее...
Прощание
Завершите письмо формулой вежливости, подписью
Буду ждать вашего письма
Готово
Назад
©Bsamples

Рис. 5

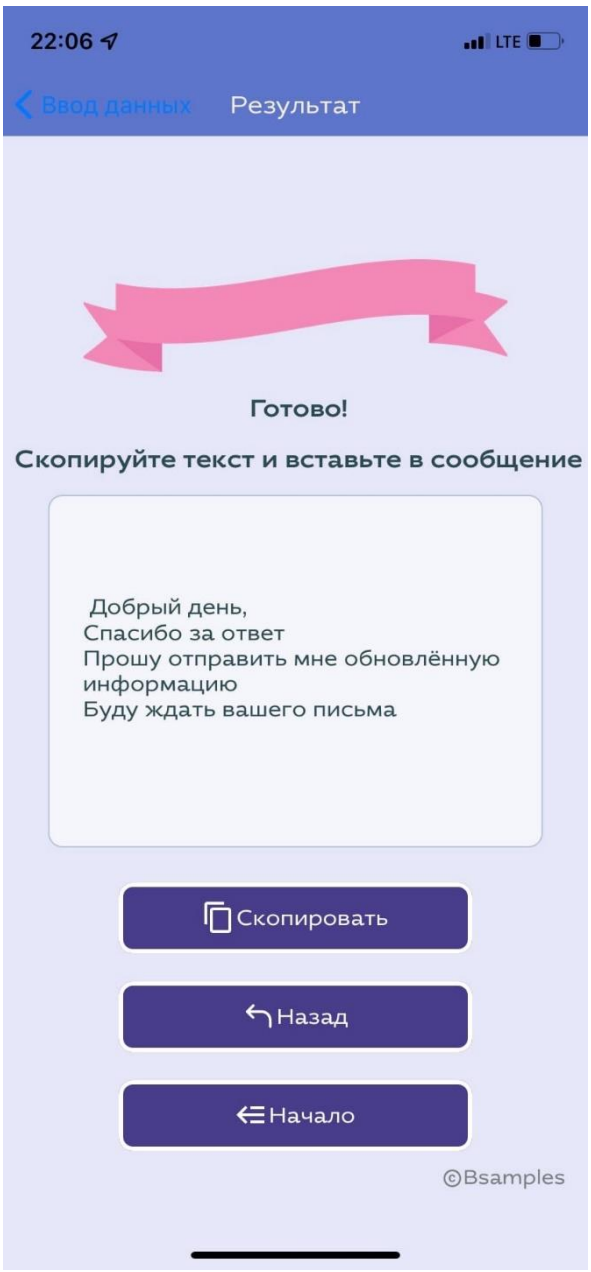
22:06
LTE
Ввод данных
Результат
Готово!
Скопируйте текст и вставьте в сообщение
Добрый день,
Спасибо за ответ
Прошу отправить мне обновлённую информацию
Буду ждать вашего письма
Скопировать
Назад
Начало
©Bsamples

Рис. 6

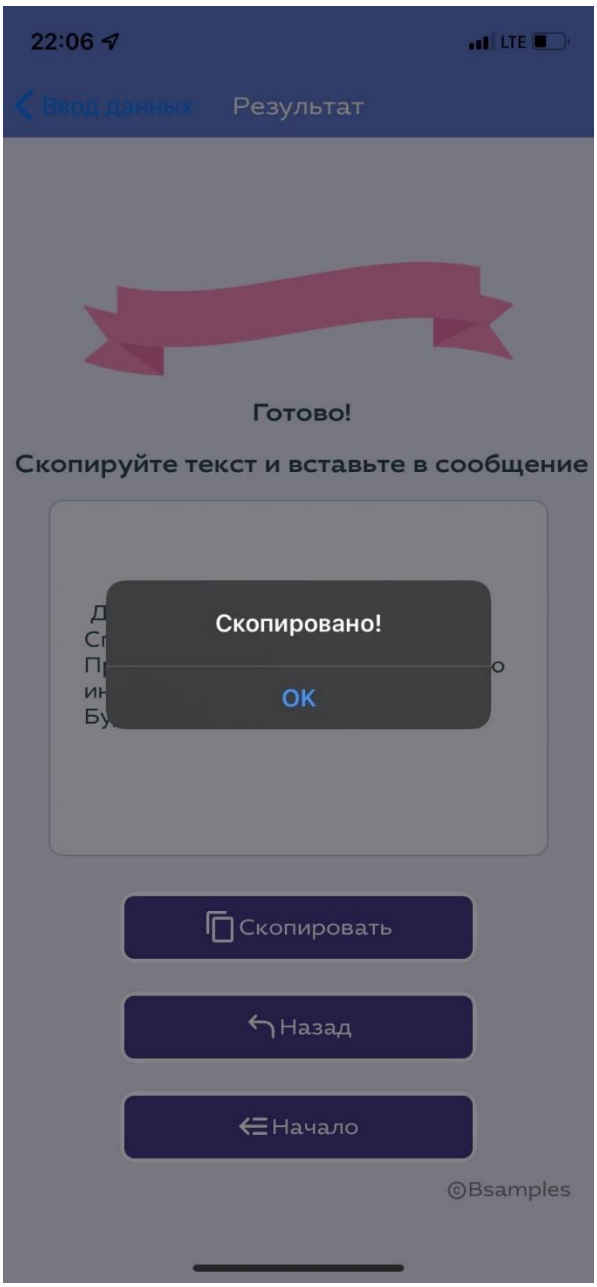

Рис. 7

Тестирование на виртуальном устройстве Android представлено на рисунках 8-12. При ошибке загрузки шрифтов отображается экран, изображенный на рис. 13

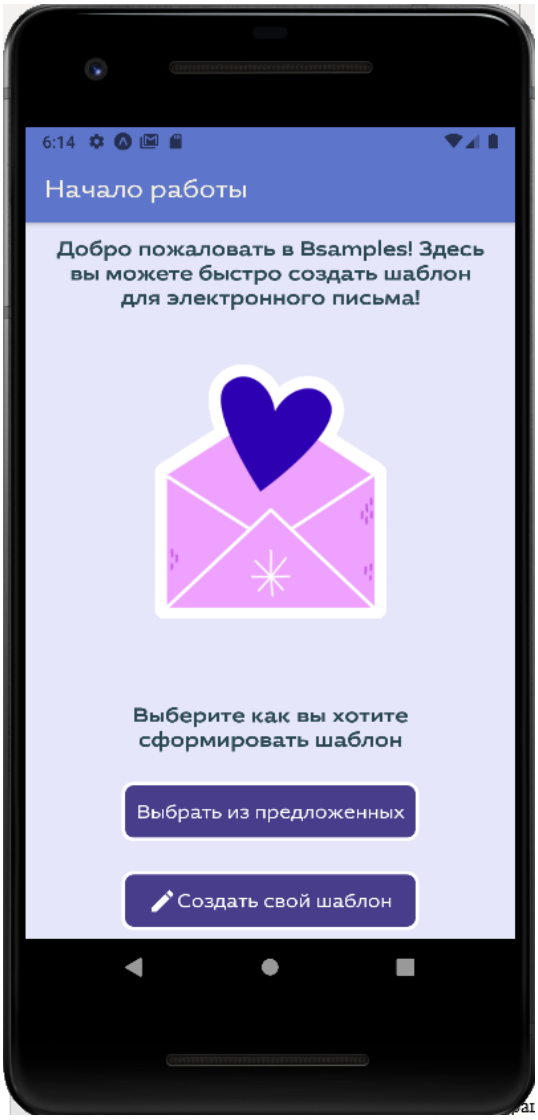
6:14
Начало работы
Добро пожаловать в Bsamples! Здесь вы можете быстро создать шаблон для электронного письма!
Выберите как вы хотите сформировать шаблон
Выбрать из предложенных
Создать свой шаблон

Рис. 8

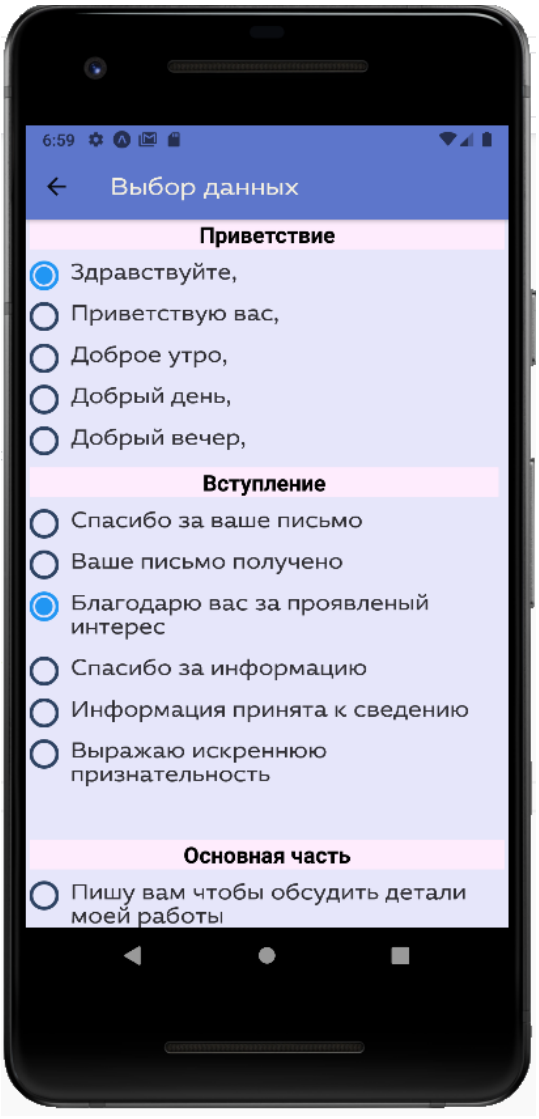
6:59
Выбор данных
Приветствие
Здравствуйте,
Приветствую вас,
Доброе утро,
Добрый день,
Добрый вечер,
Вступление
Спасибо за ваше письмо
Ваше письмо получено
Благодарю вас за проявленый интерес
Спасибо за информацию
Информация принята к сведению
Выражаю искреннюю признательность
Основная часть
Пишу вам чтобы обсудить детали моей работы

Рис. 9

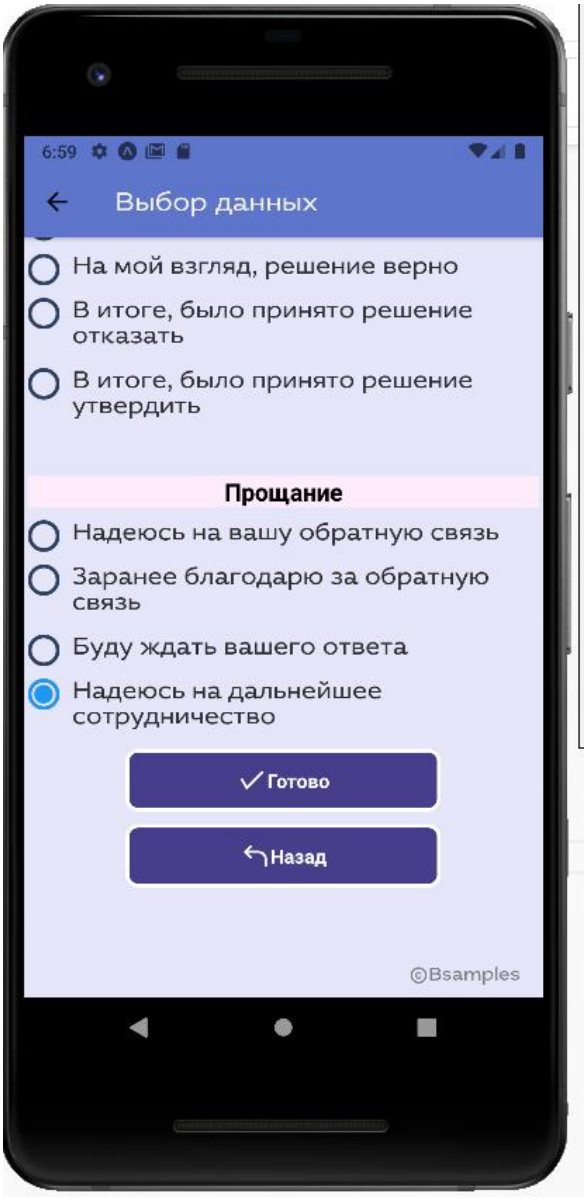
6:59
Выбор данных
На мой взгляд, решение верно
В итоге, было принято решение отказать
В итоге, было принято решение утвердить
Прощание
Надеюсь на вашу обратную связь
Заранее благодарю за обратную связь
Буду ждать вашего ответа
Надеюсь на дальнейшее сотрудничество
Готово
Назад
©Bsamples

Рис. 10

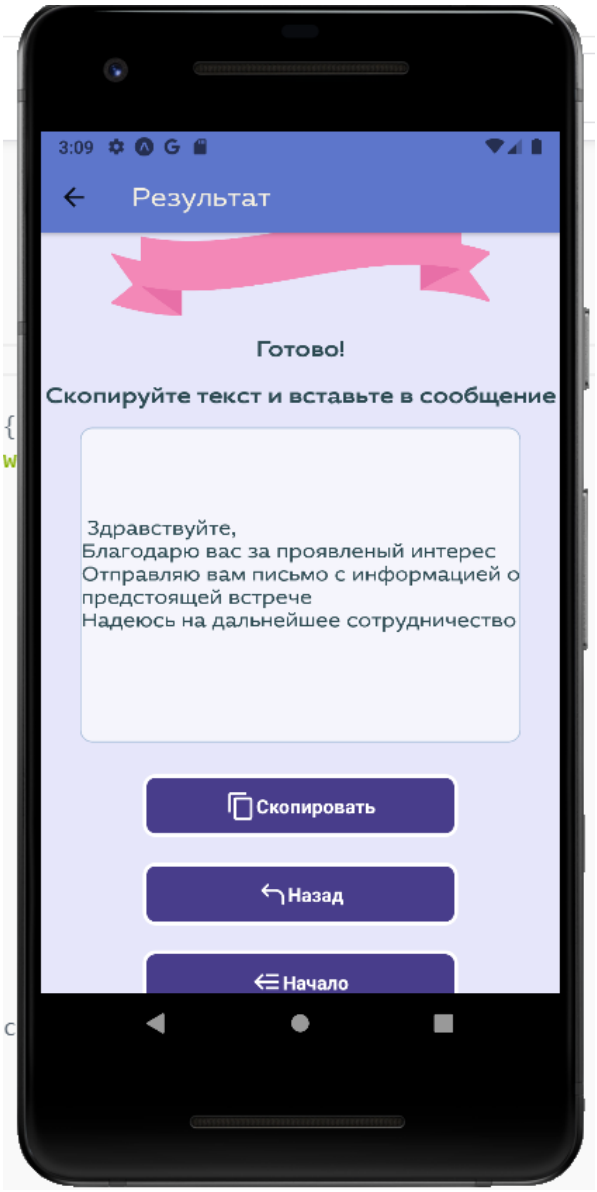
3:09
Результат
Готово!
Скопируйте текст и вставьте в сообщение
Здравствуйте,
Благодарю вас за проявленый интерес
Отправляю вам письмо с информацией о предстоящей встрече
Надеюсь на дальнейшее сотрудничество
Скопировать
Назад
Начало

Рис. 11

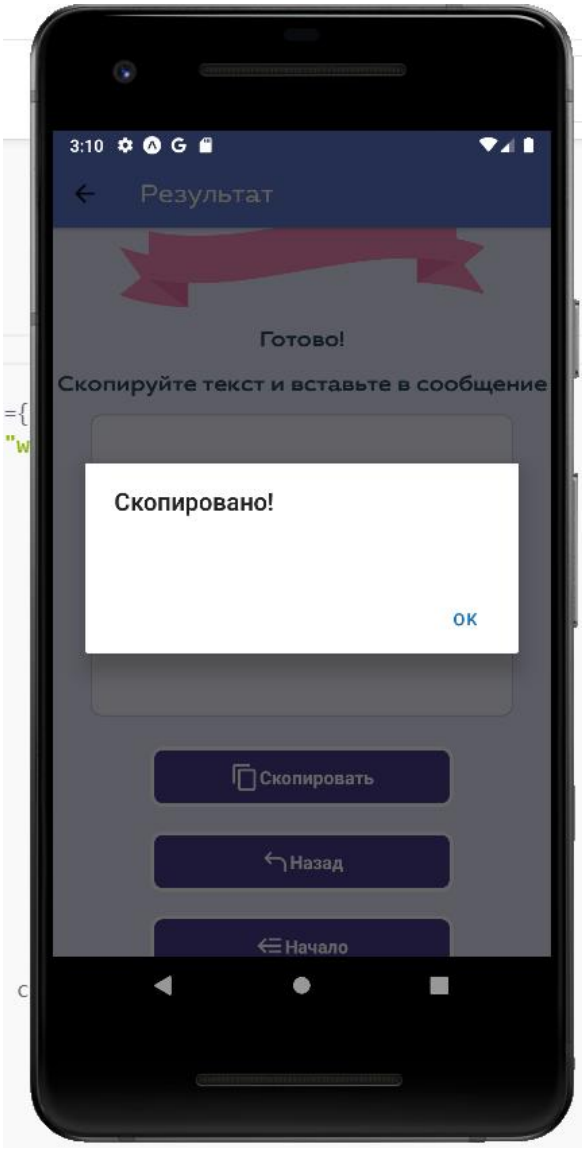

Рис. 12

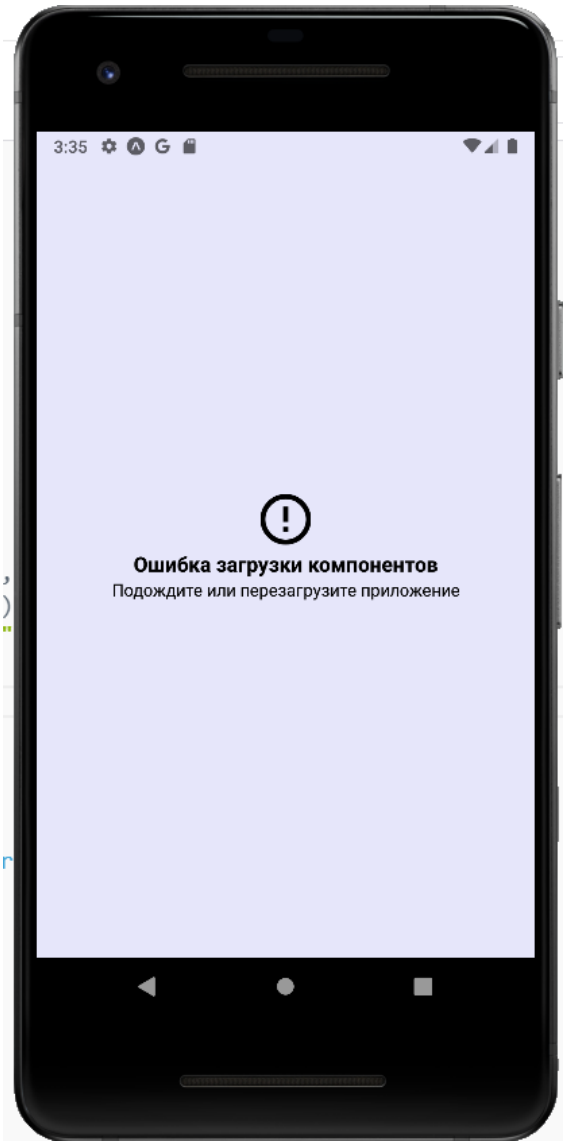

Рис. 13

СТРУКТУРА ПРИЛОЖЕНИЯ

Директория проекта имеет базовую структуру, состоящую из папки src (сокр. от англ. *source* – источник), главного файла приложения App.js и файла конфигурации package.json. В каталоге src также содержатся папки navigations с файлом навигации AppNavigator.js, screens со всеми экранами приложения (Start.js, Choice.js, Input.js, Result.js) и assets с локальными ресурсами. Assets включает в себя каталоги images и fonts с используемыми в программе изображениями и шрифтами. Файл App.js обеспечивает функционирование приложения, используя компоненты и ресурсы папки src, а в файле package.json все необходимые для программы зависимости.

ЗАКЛЮЧЕНИЕ

В ходе данного проекта на основе фреймворка React Native и с помощью открытой платформы Expo Snack было разработано мобильное кроссплатформенное приложение для создания шаблонов электронных писем, подходящее для iOS и Android. Для реализации проекта была изучено общее строение сообщений и выявлена стандартная структура, которая в последствии применяется для подготовки и формирования шаблонов. Были выполнены все поставленные задачи:

- Создание необходимых компонентов для корректной работы приложения
- Разработка функций для обработки и подготовки полученных от пользователя данных и формирования шаблона
- Реализация нескольких экранов приложения и навигации между ними
- Разработка дизайна пользовательского интерфейса

Проект удовлетворяет всем поставленным в техническом задании требованиям и критериям: успешно реализована корректная навигация между созданными окнами приложения, разработан собственный дизайн с понятным интерфейсом, реализован алгоритм, позволяющий пользователю создавать шаблоны электронных писем.

Код приложения можно посмотреть в репозитории Github по ссылке [3] и в проекте Expo[4]

[3] https://github.com/aveabina/Bsamples-CreatingEmailTemplates.git

[4] https://snack.expo.dev/@aveabina/bsamples

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1) Сорока, А. С. Разработка мобильного кроссплатформенного приложения с помощь React Native/ А. С. Сорока, Т. В. Завадская. — Текст : непосредственный Современные информационные технологии в образовании и научных исследованиях (СИТОНИ-2019). — Донецк : Донецкий национальный технический университет, 2019. — С. 215-219.
2) Создание мобильного приложения на языке программирования JavaScript (React Native) Виноградский В.Г., Винокуров А.В. // Вестник калужского университета. - 2021. - №2. - С. 101-114.
3) React Fundamentals // React Native URL: https://reactnative.dev/docs/intro-react (дата обращения: 03.05.2022).
4) NavigationContainer // React Navigation URL: https://reactnavigation.org/docs/navigation-container (дата обращения: 01.05.2022).
5) Кан Марк Основы программирования на JavaScript. - 2-е изд. - М.: Национальный открытый университет "ИНТУИТ", 2016. - 167 с.