СОВРЕМЕННЫЕ РЕШЕНИЯ В СОЗДАНИИ МОБИЛЬНЫХ ПРИЛОЖЕНИЙ

Выполнил:
Петров Игорь Рафаэлевич, 10 класс
МАОУ «Лицей инновационных технологий №36»
г. Набережные Челны, республика Татарстан.

Научный руководитель:
Трофимова Нина Владимировна, МАОУ «Лицей инновационных технологий №36», учитель информатики высшей категории
г. Набережные Челны, республика Татарстан.

Москва – 2021

Оглавление

Аннотация

Ежедневно огромное количество пользователей мобильных платформ используют продукцию различных компаний. Мобильные приложения, являясь посредником между пользователями и информацией в крупнейшей среде их взаимодействия, должны быть максимально удобными для использования. Нельзя забывать и про конкуренцию, ведь если пользователю не нравится, например, внешний вид нашего приложения, то он может перейти на использование другого приложения, что означает потерю заинтересованности аудитории в этом продукте, а может и в компании в целом. Поэтому важно, чтобы программы для мобильных платформ имели гибкую настройку, для удобства большинства пользователей. Это увеличит доверие публики к производителю и в будущем люди, столкнувшиеся с выбором программы от нескольких различных компаний, охотнее выберут использование продукции от уже знакомого производителя. В ходе создания этой работы, я проанализировал некоторые популярные приложения и их аналоги, выявил недостатки и создал прототип приложения для демонстрации оптимальных решений на примере интерфейса социальной сети. Подразумевается, что исходный код можно будет использовать в качестве шаблона с уже реализованным полезным функционалом настроек.

Введение:

Актуальность. Ежедневно огромное количество людей пользуются мобильными приложениями. В наше время мы получаем намного больше информации через смартфоны, чем через другие источники, такие как книги или даже персональные компьютеры. Это действительно так, ведь намного проще достать из кармана небольшой телефон и получить искомую информацию, нежели идти до ближайшего компьютера или библиотеки. Таким образом, смартфоны стали нашими постоянными сопровождающими в мир информации, а чтобы использовать их возможности по максимуму, насколько нам это нужно, мы пользуемся мобильными приложениями. Рынок мобильных приложений – это крупнейший в мире рынок, и он развивается очень быстрыми темпами. Появляются новые компании, разработчики, они создают новые приложения, растёт конкуренция, а значит и требования к производимым приложениям.

Проблема. Требования пользователей к приложениям растут, и всё чаще недостаток функционала приложения заставляет пользователя переходить на использование аналогичного приложения с желаемой функцией, но другой компании, что для нашей означает потерю заинтересованности аудитории. Во избежание таких ситуаций, необходимо добавлять в своё приложение такой функционал.

Цель работы – определить причину перехода пользователей к использованию аналогов приложений (используемых ранее) и создать шаблон приложения с полезным функционалом и гибкими настройками, тем самым избегая в будущем потери пользователей.

Задачи:

1. Найти приложения и их аналоги, на которые переходят пользователи;
2. Проанализировать несколько пар таких приложений и определить основные недостатки первых из них, которых нет в их аналогах;
3. Подвести итоги анализа;
4. Выбрать подходящий инструмент для разработки приложения;
5. Создать приложение-шаблон.
6. Добавить комментарии в код для дальнейшего удобного использования.

Поиск и анализ недостатков

Telegram и Plus Messenger:

Telegram – популярный мессенджер с открытым исходным кодом.

Plus Messenger – модифицированная версия Telegram.

Несмотря на то, что Telegram и так имеет возможность кастомизации интерфейса (рис.1, 2), Plus Messenger сильно расширил эту возможность, позволяя настраивать всё до мельчайших деталей для дальнейшего удобного использования (рис.3). Plus Messenger является самым популярным модом Telegram, и из-за возможности кастомизации интерфейса его использует большое количество пользователей Telegram.

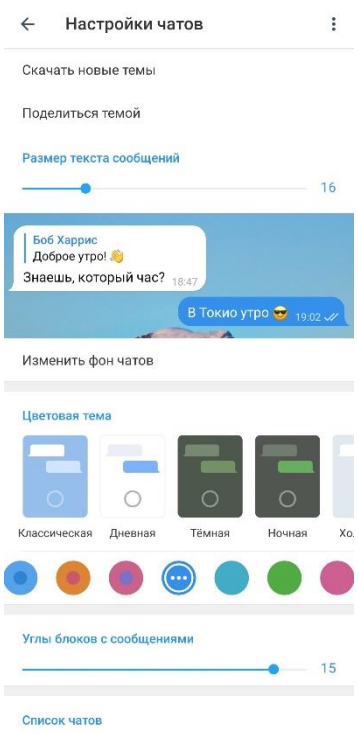


Рисунок 1. Возможности кастомизации Telegram и Plus Messenger (ч.1)

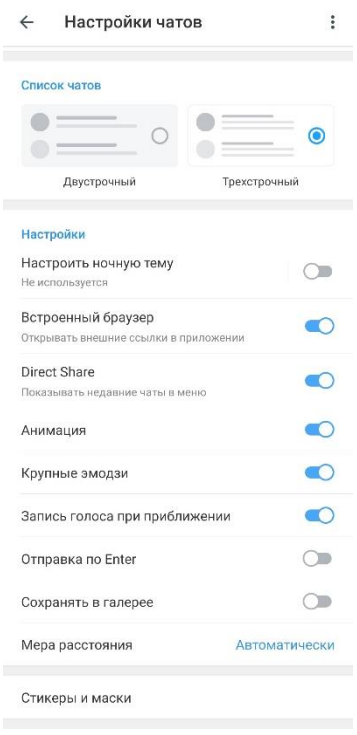


Рисунок 2. Возможности кастомизации Telegram и Plus Messenger (ч.2)

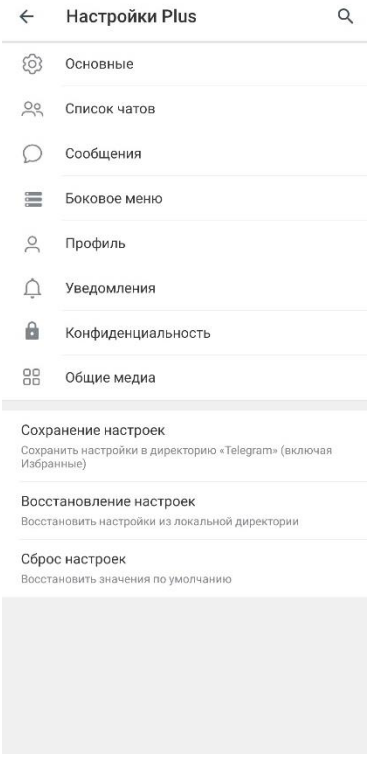


Рисунок 3. Возможности кастомизации Plus Messenger (На рис. - меню настроек. Суммарное кол-во настроек Plus Messenger больше кол-ва настроек Telegram).

ВКонтакте и SOVA V RE:

ВКонтакте – популярнейшая социальная сеть в России (рассматривается официальное мобильное приложение последней версии на момент написания работы 6.20).

SOVA V RE – неофициальная модифицированная версия мобильного приложения ВКонтакте с расширенным функционалом.

То, что пользователи, не имея возможности перейти на аналогичное мобильное приложение с расширенным функционалом, создали неофициальную модифицированную версию приложения наглядно демонстрирует требуемые пользователями функции, которых на данный момент нет в официальном приложении. Из функций, не оказывающих негативное влияние на ведение бизнеса компании, стоит выделить возможность гибкой кастомизации интерфейса (рис. 4, 5). В официальном приложении есть только возможность переключить светлую или тёмную тему и выбрать фон для чата.

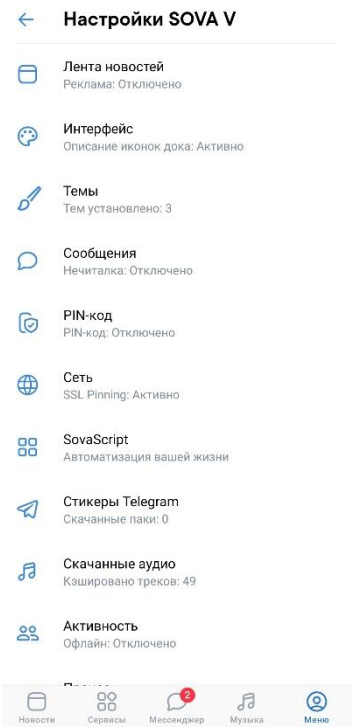


Рисунок 4. Возможности кастомизации SOVA V RE (ч. 1)

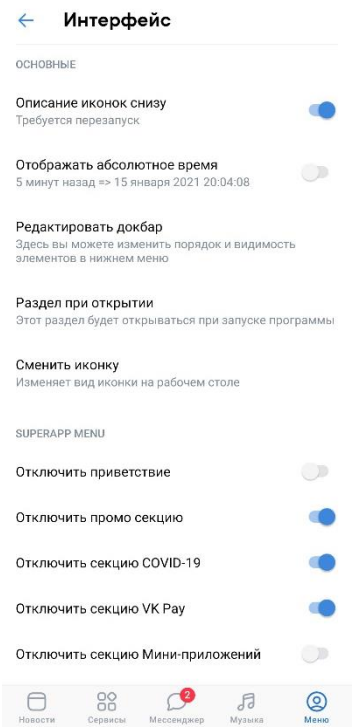


Рисунок 5. Возможности кастомизации SOVA V RE (ч. 2)

Pixel Launcher и Customized Pixel Launcher:

Pixel Launcher – стандартный лончер для устройств Pixel от компании Google. Результат на втором месте в поиске Play Market по запросу ‘Pixel Launcher’ (на момент написания работы).

Customized Pixel Launcher – модифицированный лончер из ОС AOSP (Android Open Source Project) с расширенным функционалом. Результат на первом месте в поиске Play Market по запросу ‘Pixel Launcher’ (на момент написания работы).

Следуя из названия, Customized Pixel Launcher позволяет лучше настраивать свой внешний вид (рис. 6).

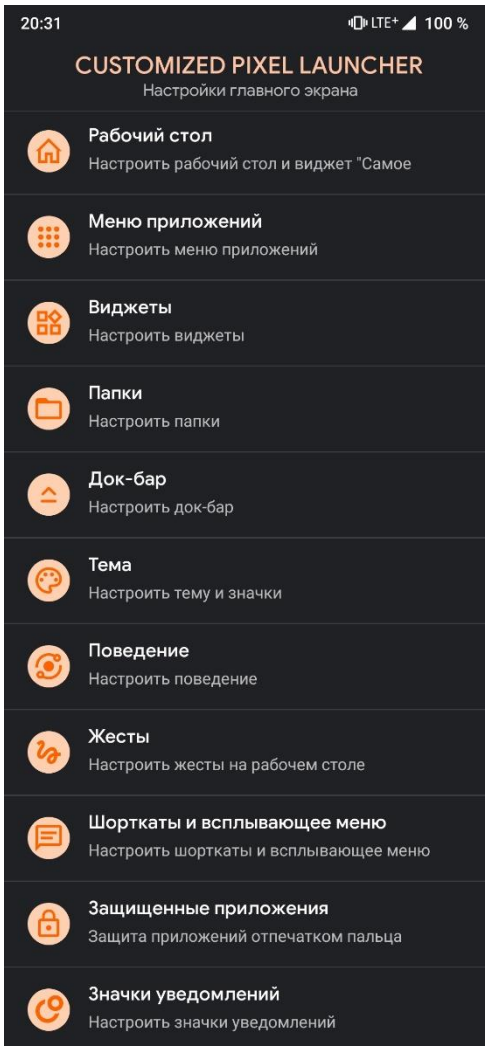


Рисунок 6. Возможности кастомизации Customized Pixel Launcher

Итоги сравнений

Сравнив три пары приложений и их аналогов, к которым переходят пользователи, я пришёл к выводу, что возможность кастомизации интерфейса – важная часть современного приложения. Люди всё чаще начинают пользоваться смартфонами, они становятся более опытными и у них появляются новые требования. В частности, требование, чтобы внешний вид приложения, которым они пользуются, был им приятен. Из наиболее частых недостатков я могу выделить отсутствие возможности выбора основного цвета приложения, отсутствие возможности перехода к более простым методам навигации. От этого исходят требования к моему шаблонному приложению с демонстрацией решений вышеперечисленных проблем.

Современное приложение должно иметь:

- Рациональные решения в структуре
- Базовые возможности кастомизации
- Интуитивно понятный интерфейс
- Приятные анимации
- Предустановки настроек для быстрого старта

Выбор инструмента разработки:

Выбор инструмента для разработки очень ответственный шаг в создании современного и производительного приложения. Для разработки я выбрал Flutter SDK (Software Development Kit) и вот почему:

Кроссплатформенность. Flutter позволяет разрабатывать приложения одновременно на три ведущие платформы: Android, iOS, Web-приложения. Это позволяет компании экономить время и денежные средства, которые могли быть потрачены на разработку аналогичного приложения на другой платформе.

Малые требования и открытость. Инструмент не требует мощного ПК, он бесплатен и имеет очень подробную документацию.

Удобство. Тестовые сборки позволяют отслеживать проблемы в отрисовке интерфейса, могут наглядно показать ‘скелет’ пользовательского интерфейса и имеют много других полезных особенностей.

Высокая скорость работы финальных сборок. По словам разработчиков, Flutter работает на той же библиотеке 2D-графики с аппаратным ускорением, которая лежит в основе Chrome и Android: Skia. Код Flutter работает на платформе Dart, которая позволяет компилировать 32-разрядный и 64-разрядный машинный код ARM для iOS и Android, а также JavaScript для Интернета и Intel x64 для настольных устройств.

Система отрисовки UI в виде виджетов. Говоря проще, всё, что видит пользователь – это виджеты. Любой текст и даже простой отступ. Виджеты могут быть Stateless и Stateful. Их главное отличие в том, что вторые могут изменять своё состояние в любой момент времени, что позволяет оптимизировать отрисовку и улучшить производительность.

Это только самые основные хорошие стороны этого инструмента для разработки. К слову, производителем Flutter является компания Google, которая разрабатывает ОС Android и инструмент разработки Android Studio. Также на Flutter начали переходить многие крупные компании, как отечественные (например, Яндекс), так и зарубежные (например, KFC). Flutter SDK часто обновляется, появляются новые функциональные возможности, улучшается оптимизация. Код для приложения пишется на языке программирования Dart, созданным компанией Google.

Приложение

Структура, виджет загрузки и виджет авторизации:

Для наглядного представления настроек интерфейса, я создал простой шаблон приложения-социальной сети с несколькими ‘страницами’. Такого понятия во Flutter нет, но так проще будет объяснить принцип представления информации. Есть “main.dart” файл, который является главным виджетом, т.е. в иерархии он будет на самой вершине и от него будут наследоваться все остальные. Сразу при запуске приложения проверяется, авторизован ли пользователь. Во время процесса, если он занимает большое количество времени, на экране пользователю будет виден только виджет загрузки (рис. 7). Это сделано для оптимизации, ведь запуск приложения и так довольно тяжёлый процесс: системе нужно переместить ресурсы приложения из постоянной памяти во временную, так ещё после этого приложение сначала отобразит виджет авторизации и только потом, когда получит информацию о том, что пользователь уже авторизован, резко изменит виджет авторизации на домашний. Это всё ещё сильнее продлевает промежуток от запуска пользователем приложения до момента, когда он может им пользоваться. К тому же в таком случае мы излишне нагружаем систему. В итоге возникают две вышеперечисленные проблемы. И для их

```
@override
Widget build(BuildContext context) {
  switch (isLogged) {
    case 0: //Анимация загрузки до получения информации о сессиях
      return Container( // Container
    case 1: //Виджет авторизации
      return Auth(
        reloadLanding: setLogged,
      );
    case 2: //Home() и Анимация на время загрузки Home()
      return Stack(alignment: Alignment.center, children: [
        Home(
          loadEnded: loadEnded,
        ), // Home
        loadingAnim
            ? AnimatedOpacity( // Container // AnimatedOpacity
            : Container()
      ]); // Stack
  }
}
```

Рисунок 7. Конструкция Switch-Case с переключением виджетов загрузки, авторизации и домашнего виджета.

предотвращения было принято решение во время загрузки данных об авторизациях отрисовывать простой виджет с анимацией загрузки (рис. 8). В случае, если процесс авторизации пройдёт быстро, пользователь даже не увидит этого виджета.

Виджет авторизации (рис. 10) несмотря на то, что он состоит из нескольких простейших элементов UI, выполняет важную функцию при взаимодействии с пользователем. Запуская приложение впервые, пользователь увидит приятные анимации появления элементов интерфейса. Анимация играет самую главную роль при первом знакомстве пользователя с новым приложением: она подсознательно направляет его по тем элементам, с которыми он будет взаимодействовать, начиная с приветствия и заканчивая возможностью погрузиться в приложение. Виджет авторизации помимо всего этого содержит в себе элемент кода, устанавливающий настройки по умолчанию (рис. 9).

```
void initState() {
  super.initState();
  //Настройки по-умолчанию:
  setStringValue('nameProfileColor', 'green');
  setStringValue('nameLentaColor', 'red');
  setStringValue('nameMessagesColor', 'indigo');
  setStringValue('nameServicesColor', 'purple');
  setDoubleValue('roundingValue', 12.0);
  setDoubleValue('elevationValue', 1.0);
  setStringValue('theme', 'system');
  setBoolValue('bvValue', true);
  setBoolValue('doubleRowValue', true);
  setBoolValue('tabValue', true);
  setBoolValue('likeAccent', true);
  setBoolValue('vibro', true);
}
```

Рисунок 9. Установка настроек по умолчанию

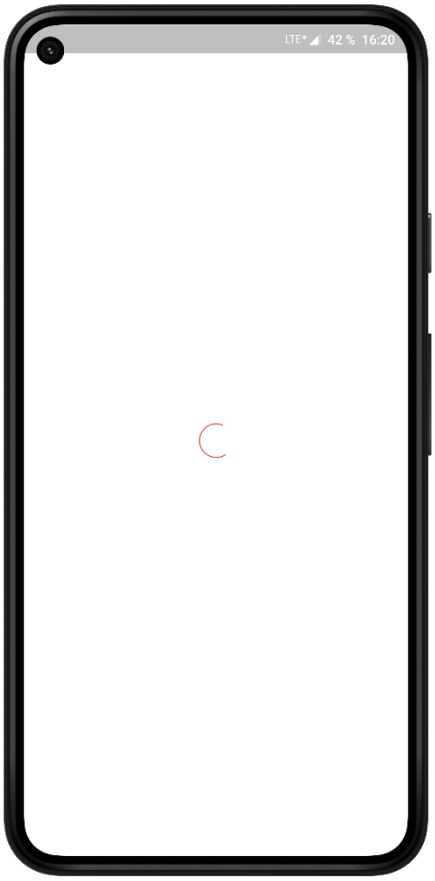


Рисунок 8. Виджет загрузки

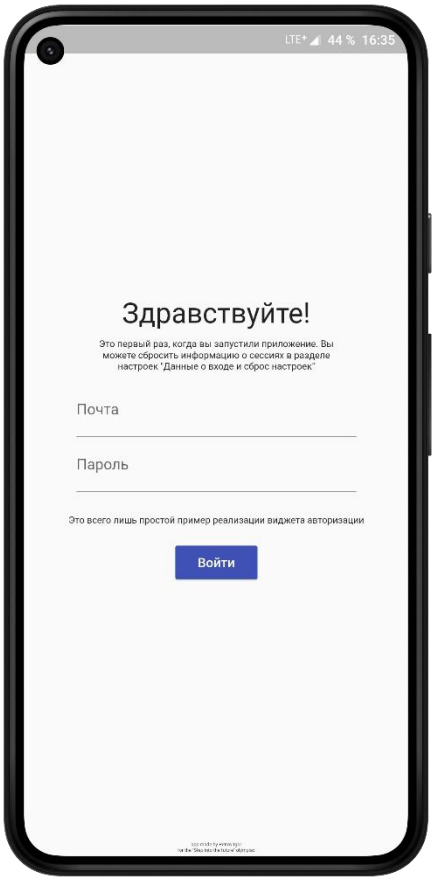


Рисунок 10. Виджет авторизации

Домашний виджет:

Домашний виджет Home() (рис. 12) содержит в себе следующие важные элементы UI:

1. PageView или конструкцию, возвращающую необходимую страницу (в зависимости от выбранных пользователем настроек);
2. BubbleBottomBar или BottomNavigationBar (в зависимости от выбранных пользователем настроек);
3. FloatingActionButton (При выборе продвинутого BottomNavigationBar, т.е. BubbleBottomBar).

В добавок ко всему перечисленному в этом виджете производится управление темой приложения (рис. 11). Переключение тем осуществляется обновлением state виджета Home() с изменением переменной tmode в зависимости от выбранной новой темой пользователем:

- Для системной темы
  tmode = ThemeMode.system;
- Для тёмной темы
  tmode = ThemeMode.dark;
- Для светлой темы
  tmode = ThemeMode.light.

```
theme: ThemeData(
  brightness: Brightness.light,
  primarySwatch: _stateColor(currentIndex),
), // ThemeData
darkTheme: ThemeData(
  brightness: Brightness.dark,
  primarySwatch: _stateColor(currentIndex),
  accentColor: _stateColor(currentIndex),
), // ThemeData
themeMode: tmode,
```

Рисунок 11. Переключение тем.

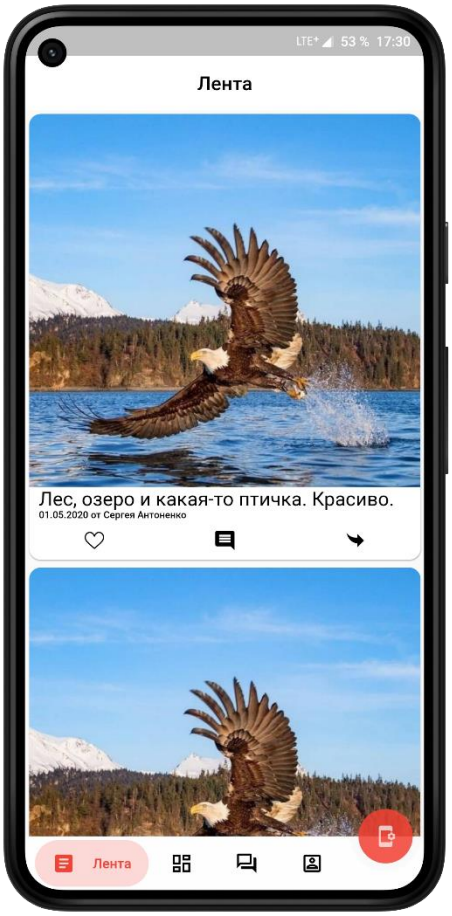


Рисунок 12. Виджеты Home().и lenta()

По нажатию на FloatingActionButton появляется ModalBottomSheet (рис. 13), который является родителем виджета SettingsPage() из файла settingsPage.dart. Меню содержит в себе все настройки приложения и находиться в этом месте для наглядной демонстрации решений. Подразумевается перемещение и перераспределение настроек при использовании шаблона для создания полноценного приложения.

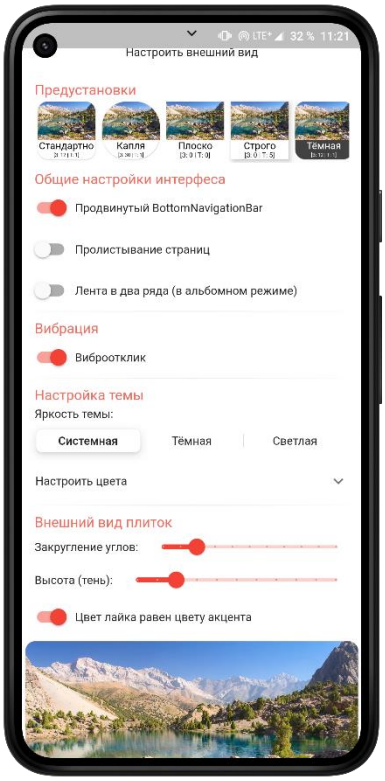


Рисунок 13. Виджет настроек (1)

Список настроек:

1. Переключение между двумя версиями BottomNavigationBar: обычной и продвинутой (BubbleBottomBar);
2. Включение / отключение пролистывания страниц;
3. Включение / отключение новостной ленты в два ряда в альбомном режиме приложения;
4. Включение / отключение виброотклика;
5. Настройка яркости темы (Системная / Тёмная / Светлая);
6. Настройка цвета акцента отдельных страниц приложения;
7. Настройка закругления плиток;
8. Настройка тени плиток;
9. Настройка цвета лайка (Равен ли цвет лайка цвету акцента);
10. Сброс настроек и данных о последнем входе;
11. Предустановки (8 шт.);

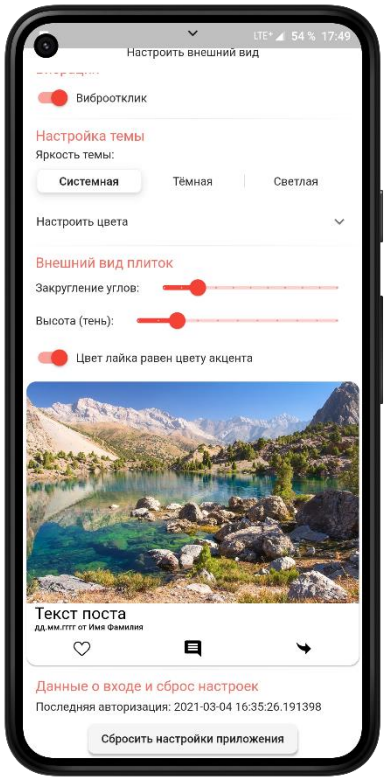


Рисунок 14. Виджет настроек (2)

Описание настроек приложения:

1. Переключение между двумя версиями BottomNavigationBar обычной и продвинутой (BubbleBottomBar):

Рисунок 15. Обычный BottomNavigationBar

Рисунок 16. Продвинутый BottomNavigationBar

Функция будет полезна для пользователей слабых устройств. По умолчанию в приложении используется BubbleBottomBar (рис. 16), но этот виджет может работать медленно на старых устройствах, поэтому была добавлена возможность использования упрощённого варианта (рис. 15).

2. Включение / отключение пролистывания страниц (рис. 17):

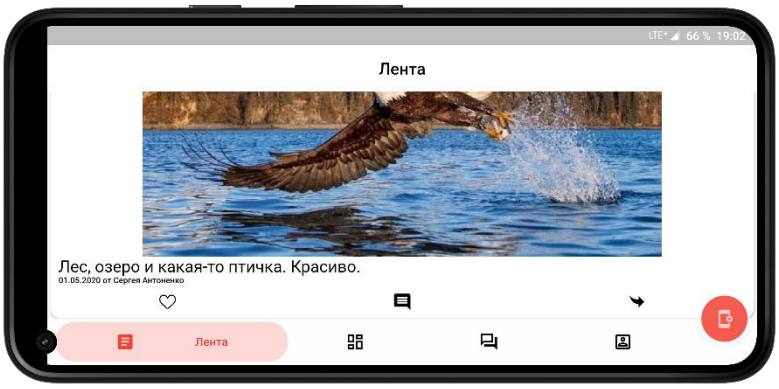
```
body: tabValue
    ? PageView(
        physics: ClampingScrollPhysics(),
        controller: _c,
        onPageChanged: (newIndex) {
          setState(() {
            this.currentIndex = newIndex;
          });
        },
        children: [
          lenta(
            vibro: vibro,
            elevation: elevationValue,
            rounding: roundingValue,
            likeAccent: likeAccent,
            doubleRow: doubleRowValue,
          ), // lenta
          services(
            showSettings: showSettings,
            elevation: elevationValue,
            rounding: roundingValue,
            likeAccent: likeAccent,
          ), // services
          Messages(),
          Profile(
            vibro: vibro,
            elevation: elevationValue,
            rounding: roundingValue,
            likeAccent: likeAccent,
            doubleRow: doubleRowValue,
          ), // Profile
        ],
      ) // PageView
    : _page(currentIndex),
```

Рисунок 17. Переключение страниц

Эта функция также позволяет уменьшить используемые ресурсы на слабых устройствах. Согласно моим наблюдениям, анимация перелистывания страниц не используется в популярных приложениях, так что для пользователей, которым привычно мгновенное переключение страниц, эта функция будет полезной. Реализуется она путём переключения виджета PageView и конструкции, возвращающей необходимую страницу. PageView, в отличие от второй конструкции, имеет анимацию перелистывания страниц, а также позволяет листать их при помощи жестов.

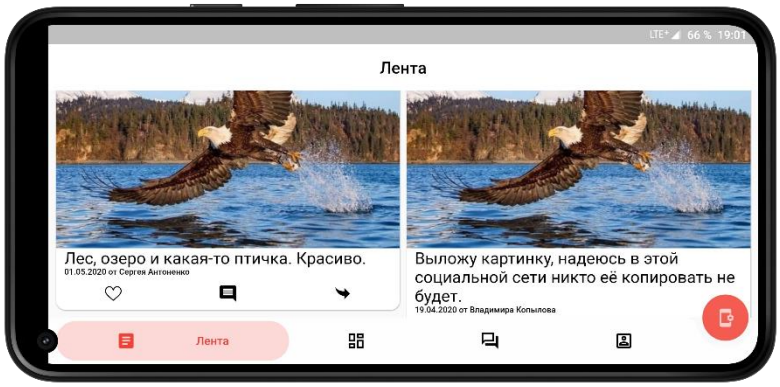
3. Включение / отключение новостной ленты в два ряда в альбомном режиме приложения:

Новостная лента приложения ВКонтакте или главная страница в приложении YouTube в альбомном режиме отображают информацию в один ряд, что на мой взгляд не рационально, ведь большое количество свободного пространства не используется. Функция “Лента в два ряда” решает эту проблему, отображая в альбомном режиме ленту в два ряда. Также, как и с пролистыванием страниц, эту функцию можно отключить.



← Рисунок 18. Лента в один ряд

Рисунок 19. Лента в два ряда →



4. Включение / отключение виброотклика:

Виброотклик расширяет получаемую пользователем информацию, добавляя тактильные ощущения по нажатию на некоторые кнопки или вызове вспомогательных функций (пример вспомогательной функции – долгое нажатие на пост в ленте для вызова функции “поделиться на своей странице”).

5. Настройка яркости темы (Системная / Тёмная / Светлая):

Большое кол-во современных приложений имеют эту функцию. В ОС Android Q (10) эта функция добавлена на системном уровне, поэтому чтобы не отставать от современных трендов, была добавлена эта функция. В отличие от многих приложений, переключение темы в настройках не требует перезапуска приложения. В таких приложениях, как ВКонтакте и Telegram смена темы сопровождается красивой Circular Reveal анимацией (расширением прозрачного круга). Я постарался разобраться в том, как это реализовано в приложении Telegram, основываясь на открытом исходном коде (рис. 21) и вот к чему я пришёл:

Рисунок 20. Переключатель яркости темы

Последовательность переключения темы:

1. По нажатию на кнопку производиться снимок экрана приложения
2. Снимок преобразуется в Bitmap
3. Снимок выводится на экран приложения
4. Меняется тема
5. Когда тема сменилась, изображение удаляется с экрана при помощи Circular Reveal анимации.

Я предпринял попытку воссоздать этот алгоритм, но столкнулся с одной проблемой: переключение темы занимало больше времени, чем я планировал изначально, поэтому функция не была окончательно добавлена в приложение, а код был закомментирован для дальнейшей доработки (рис. 22).

```java
int[] pos = (int[]) args[2];
int w = drawerLayoutContainer.getMeasuredWidth();
int h = drawerLayoutContainer.getMeasuredHeight();
Bitmap bitmap = Bitmap.createBitmap(drawerLayoutContainer.getMeasuredWidth(), drawerLayoutContainer.getMeasuredHeight(), Bitmap.Config.ARGB_8888);
Canvas canvas = new Canvas(bitmap);
drawerLayoutContainer.draw(canvas);
themeSwitchImageView.setImageBitmap(bitmap);
themeSwitchImageView.setVisibility(View.VISIBLE);
float finalRadius = (float) Math.max(Math.sqrt((w - pos[0]) * (w - pos[0]) + (h - pos[1]) * (h - pos[1])), Math.sqrt(pos[0] * pos[0] + (h - pos[1])
Animator anim = ViewAnimationUtils.createCircularReveal(drawerLayoutContainer, pos[0], pos[1], 0, finalRadius);
anim.setDuration(400);
anim.setInterpolator(CubicBezierInterpolator.EASE_IN_OUT_QUAD);
anim.addListener(new AnimatorListenerAdapter() {
    @Override
    public void onAnimationEnd(Animator animation) {
        themeSwitchImageView.setImageDrawable(null);
        themeSwitchImageView.setVisibility(View.GONE);
    }
});
anim.start();
instant = true;
```

Рисунок 21. Код переключения темы с использованием Circular Reveal анимации в Telegram (Часть открытого исходного кода из официального репозитория на github.com)

```dart
/*changeTheme() async {
  RenderRepaintBoundary boundary = screen.currentContext.findRenderObject();
  ui.Image image = await boundary.toImage(pixelRatio: 1);
  ByteData byteData = await image.toByteData(format: ui.ImageByteFormat.png);
  OverlayEntry overlayEntry = OverlayEntry(builder: (context) {
    return Positioned(
        top: 0, //MediaQuery.of(context).size.height / 2,
        right: 0, //MediaQuery.of(context).size.width / 2,
        child: CircularRevealAnimation(
            minRadius: 0,
            maxRadius: 1000,
            animation: CurvedAnimation(
              parent: animationController,
              curve: Curves.easeIn,
            ),
            child: Container(
                alignment: Alignment.center,
                child: Image.memory(byteData.buffer.asUint8List()),
                height: MediaQuery.of(context).size.height,
                width: MediaQuery.of(context).size.width)));
  });
  Overlay.of(scaffoldKey.currentContext).insert(overlayEntry);
  await animationController.forward(from: 1000);
  await animationController.reverse(from: 1000);
  await Future.delayed(Duration(milliseconds: 250));
  overlayEntry.remove();
}*/
```

Рисунок 22. Мой вариант кода для переключения темы с использованием Circular Reveal анимации.

6. Настройка цвета акцента отдельных страниц приложения:

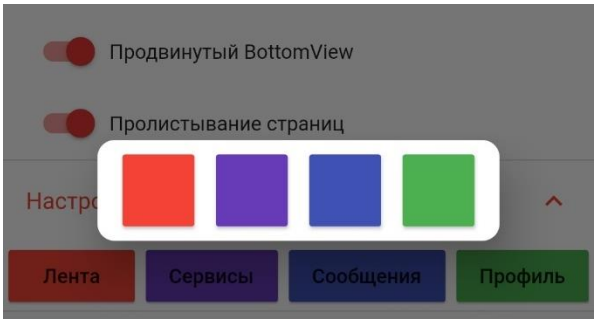


Рисунок 23. Диалог выбора цвета

При нажатии на кнопку с названием страницы появляется диалог выбора цвета (рис. 23) соответствующей страницы. Функция позволяет настроить цвет акцента приложения на каждой из страниц отдельно. При желании пользователь может использовать одинаковые цвета, чтобы придать акценту всего приложения один общий цвет. Для демонстрации функции было добавлено 4 цвета из палитры Material: красный, тёмно-фиолетовый, индиго, зелёный. При изменении цвета страницы, на которой находиться пользователь, проигрывается Fade анимация смены цвета. Цвет акцента влияет на:

1. Цвет некоторых заголовков и частей текста;
2. Цвет выбранной страницы в BottomNavigationBar и BubbleBottomBar;
3. Цвет FloatingActionButton (при включенном BubbleBottomBar);
4. Цвет лайка (при включенной опции “Цвет лайка равен цвету акцента”);
5. Цвет переключателей;
6. Цвет splash (при нажатии на пост)

7, 8. Настройка закругления и тени плиток:

Интерфейс приложения сильно изменяется, при изменении всего двух параметров плитки, поэтому каждый сможет подобрать для себя вариант внешнего вида плиток, который ему по душе, ведь настройки закругления и тени не зависят друг от друга. Плитками в приложении являются посты и некоторые кнопки. В будущем при использовании можно будет с лёгкостью добавить больше плиток, просто оборачивая виджет в Card() с регулируемыми параметрами elevation и shape. Вот несколько примеров внешнего вида плиток:

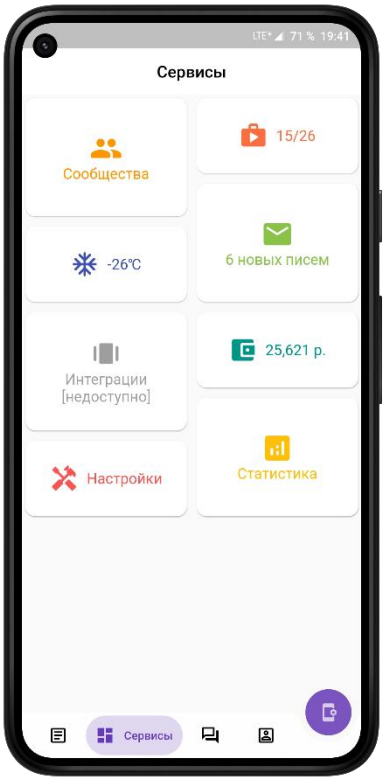


Рисунок 24.1. Плитки (Стандартные настройки)

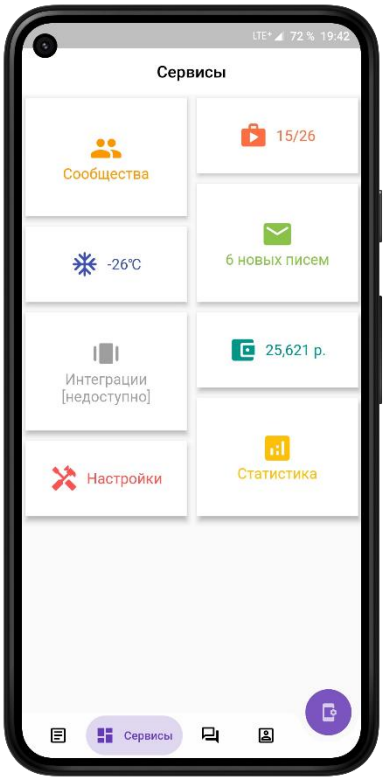


Рисунок 24.2. Плитки (Тень: max, Закругление: min)

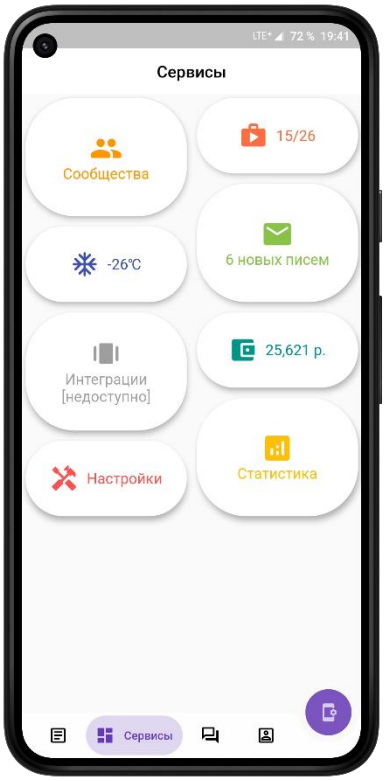


Рисунок 24.3. Плитки (Тень: max, Закругление: max)

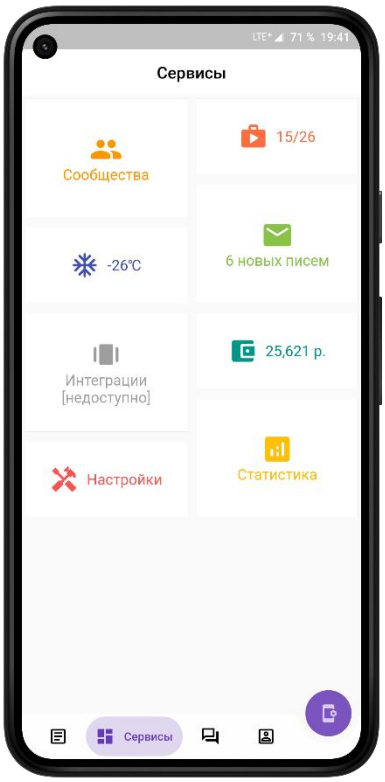


Рисунок 24.4. Плитки (Тень: min, Закругление: min)

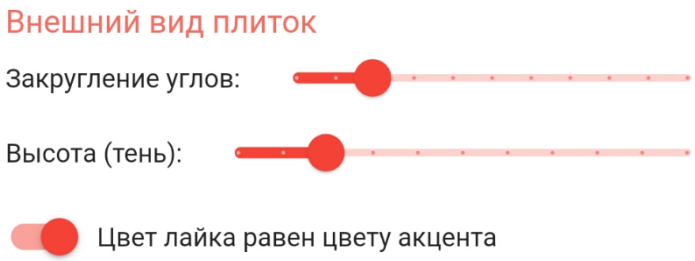


Рисунок 25. Настройки плиток

Настройка производится при помощи ползунков с фиксированными значениями для удобства пользователей (рис. 25).

9. Настройка цвета лайка:

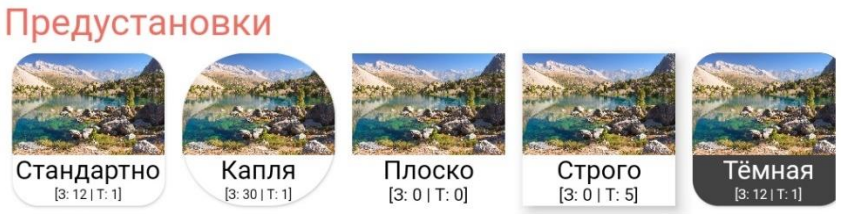
Функция изменяет цвет лайка с красного (как в большинстве приложений) на цвет акцента текущей страницы.

10. Сброс настроек и данных о последнем входе:

Функция нужна для отладки приложения. Позволяет сбросить все настройки и при следующем запуске приложения снова увидеть виджет авторизации.

11. Предустановки:

Предустановки были добавлены для возможности быстрой настройки приложения. Эта функция будет полезной для тех, кто не хочет разбираться во всех настройках приложения, но всё равно желает изменить интерфейс. Такая функция есть также и в Telegram (рис. 1) называется она “Цветовая Тема” и в качестве примера реализации, она была добавлена в этот шаблон. Каждый пункт предустановок является виджетом Card с фиксированными настройками закругления, тени и освещённости, характеризующими внешний вид плиток этой темы. Всего для демонстрации настроек было добавлено 8 штук: Стандартно, капля, плоско, строго, тёмная, тёмная капля, тёмная плоская, тёмная строгая.



Оформление шаблонных страниц приложения

Оформление страниц нужно для более чёткого понимания демонстрируемых решений. Всего в приложении 4 основные страницы (не считая страницы авторизации) и 1 дополнительная страница для отображения комментариев. Пользователь может переключаться между страницами. Страницы несут только декоративную функцию для шаблона, но при доработке могут полноценно использоваться.

1. Страница ленты, профиля и комментарии:

Виджет страницы ленты возвращает Container со SmartRefresher и ListView или Wrap (в зависимости от опции “Лента в два ряда”) с уже заранее заготовленными постами или страницу комментариев в зависимости от булевой переменной inComments. В приложении не используется ни чей Api, так как это всего лишь демонстрация. Для того, чтобы использовать ленту в дальнейшем, можно использовать ListView.builder (рис. 26).

```
@override
Widget build(BuildContext context) {
  return Container(
    color: Colors.white,
    child: SmartRefresher(
      header: ClassicHeader(
        refreshingText: 'Обновление',
        completeText: 'Успешно',
        idleText: 'Потяните для обновления',
        releaseText: 'Отпустите для обновления',
        height: 50,
      ), // ClassicHeader
      controller: _refreshController,
      onRefresh: _onRefresh,
      onLoading: _onLoading,
      child: ListView.builder(
        physics: ScrollPhysics(),
        primary: false,
        itemBuilder: (context, index) => post(),
      ), // ListView.builder
    ), // SmartRefresher
  ); // Container
}
```

Рисунок 26. Пример использования ленты с ListView.builder

Страница профиля отличается от страницы ленты только наличием фотографии профиля и некоторой информацией о странице пользователя (рис. 27).

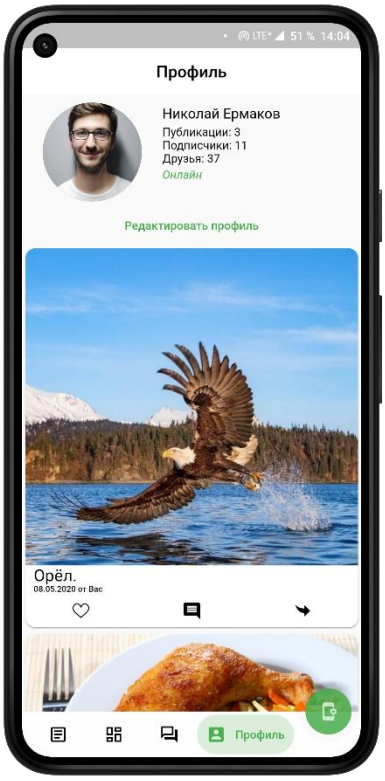


Рисунок 27. Виджет профиля

2. Страница опций

На странице представлено интересный вариант расположения плиток с возможным будущим функционалом. Каждая плитка по нажатию проигрывает анимацию “проваливания” (рис. 28).

3. Страница сообщений

Виджет страницы сообщений также, как и виджет страницы ленты и страницы профиля содержит SmartRefresher. На странице представлен простой вариант реализации списка сообщений (рис. 29).

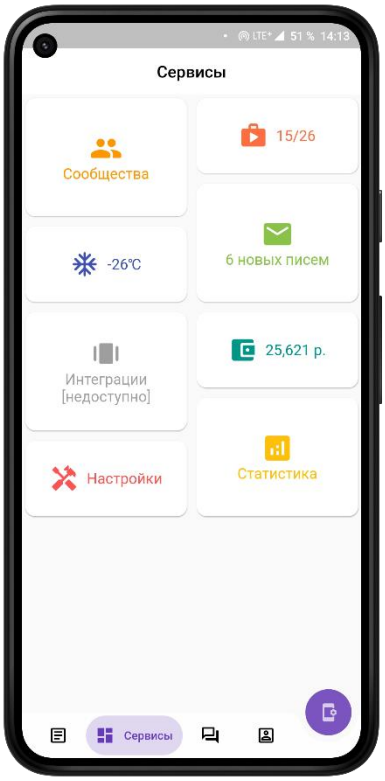


Рисунок 28. Виджет сервисов

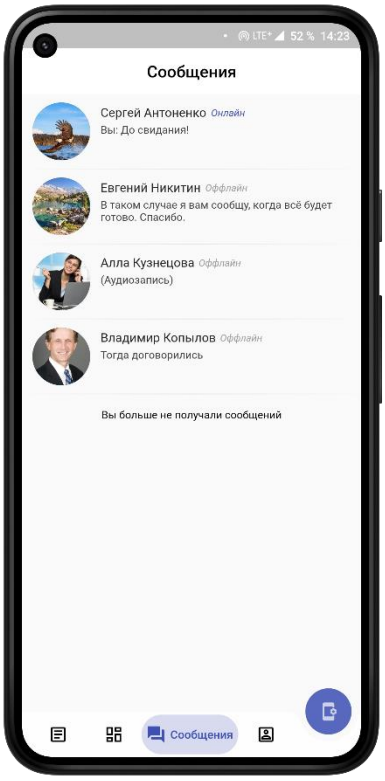


Рисунок 29. Виджет собщений

Дополнительные иллюстрации

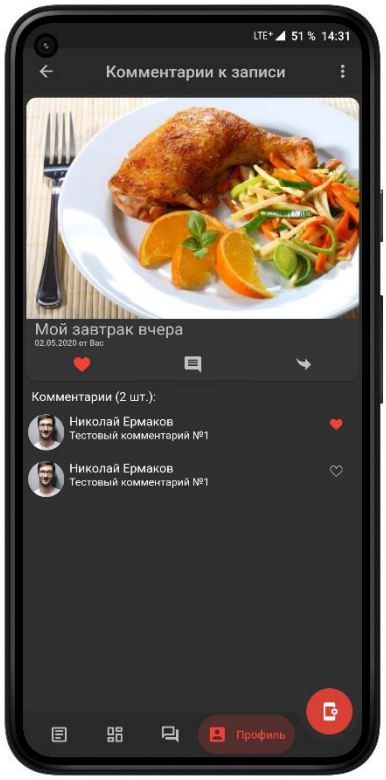


Рисунок 30. Виджет комментариев (тёмная тема)

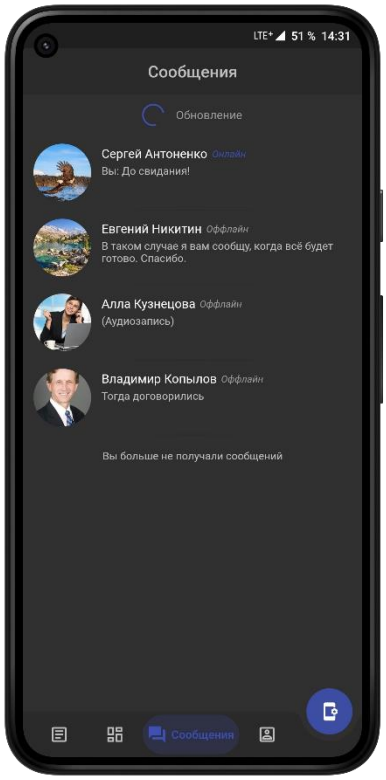


Рисунок 31. Виджет сообщений с обновлением содержания (тёмная тема)

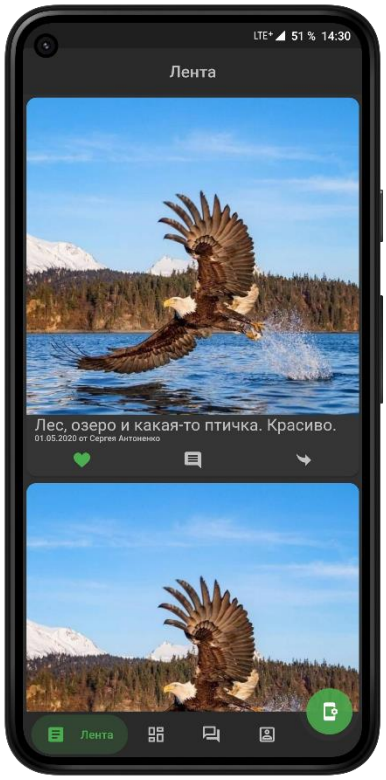


Рисунок 32. Виджет ленты (тёмная тема)

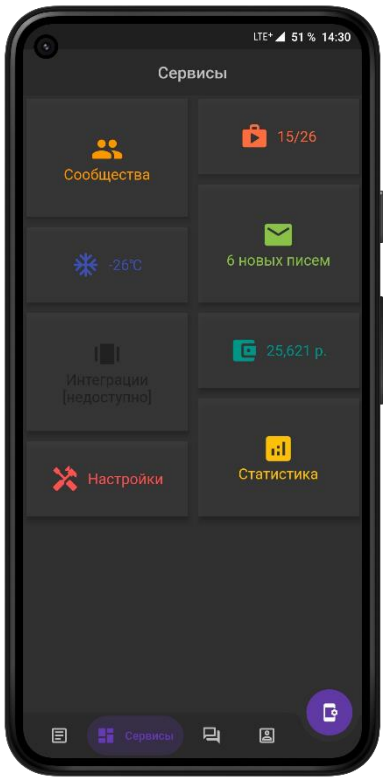


Рисунок 33. Виджет сервисов (тёмная тема, строгое оформление)

Интеграция шаблона:

Исходный код приложения-шаблона можно с лёгкостью интегрировать в другие проекты и модифицировать под свои нужды. Код дополнен комментариями для понимания его работы. Шаблон сделан модульно, т.е. разделен на несколько файлов для удобства редактирования. Некоторые из этих файлов могут работать отдельно от общего кода, таким образом, в случае надобности, можно использовать лишь часть кода шаблона. Для интеграции кода в другой проект Flutter нужно:

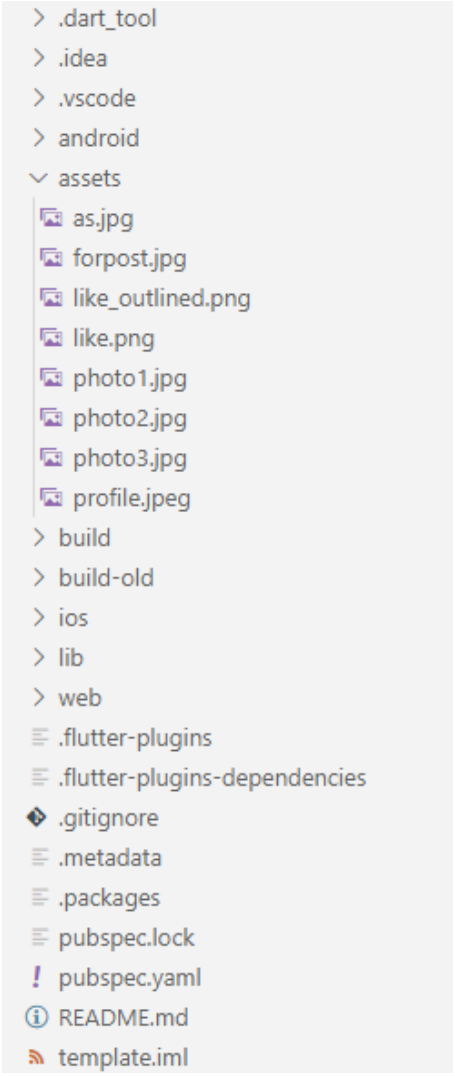


Рисунок 34. Добавление папки asset

1. Создать новый проект Flutter;
2. Удалить файлы из папки lib и папку test (они автоматически создаются и содержат в себе код демонстрационной программы);
3. Переместить файлы шаблона в папку lib;
4. Добавить в файле pubspec.yaml поддержку дополнений (рис. 35);
5. Добавить в файле pubspec.yaml поддержку asset-изображений и добавить эти изображения в проект (рис. 34, 36);
6. Редактировать некоторые строки в файлах кода шаблона (обычно это пути к другим файлам кода).

```yaml
dependencies:
  flutter:
    sdk: flutter
  bubble_bottom_bar: ^1.2.0
  shared_preferences: ^0.5.12+4
  pull_to_refresh: ^1.6.3
  flutter_staggered_grid_view: ^0.3.3
  delayed_display: ^1.1.0
  flutter_bounce: ^1.0.8
  vibration: ^1.7.3
```

Рисунок 35. Добавление поддержки дополнений в файле pubspec.yaml

```yaml
assets:
  - assets/photo1.jpg
  - assets/photo2.jpg
  - assets/photo3.jpg
  - assets/profile.jpeg
  - assets/as.jpg
  - assets/forpost.jpg
  - assets/like.png
  - assets/like_outlined.png
```

Рисунок 36. Добавление поддержки asset изображений в файле pubspec.yaml

Заключение:

В ходе создания проекта, мной были проанализированы три пары приложений и аналогов, выявлены причины перехода от первых ко вторым. В созданном мной шаблонном приложении были реализованы некоторые решения, необходимые современным приложениям для предотвращения потери пользователей. Исходный код дополнен комментариями и может легко интегрироваться в другой проект, поддерживающий Flutter и Dart.

Ссылки:

1. Файлы и исходный код приложения Telegram для Android: https://github.com/DrKLO/Telegram/
2. Официальная сайт Flutter SDK: https://flutter.dev/
3. Магазин приложений Google Play: https://play.google.com/store/apps
4. Репозиторий дополнений для Flutter: https://pub.dev/
5. Visual Studio Code (Программа, в которой писался код): https://code.visualstudio.com/