УДК 004.624

Создание приложения по борьбе с прокрастинацией на Flutter

Кизянов Антон Олегович
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
В данной статье описано приложение для повышения производительности, содержащее обратный отсчет, который сообщит об оставшемся времени работы или перерыва. Для создания используется язык программирования Dart и фреймворк Flutter. Созданное приложение позволяет контролировать время, затраченное на продолжительность рабочего времени, короткие и длинные перерывы и сохранить историю на своем устройстве.
Ключевые слова: Dart, Flutter, Прокрастинация

Building an Anti-Procrastination Application on Flutter

Kizyanov Anton Olegovich
Sholom-Aleichem Priamursky State University
student

Abstract
This article describes a performance app. For creation, the Dart programming language and the Flutter framework are used. The created application allows you to control the time spent on working hours, short and long breaks and save history on your device.
Keywords: Dart, Flutter, Procrastination

Любой человек постоянно находится в состоянии войны. Это война с битвами, которые происходят ежедневно и влияют на качество жизни. Это война против отвлекающих факторов. В любой момент может возникнуть соблазн проверить свою электронную почту или посмотреть социальные сети, послушать музыку, или перекусить. Есть несколько исследований, показывающих, что, если человек хочет добиться успеха в своей деятельности, ему нужно практиковать глубокое вовлечение работу. Глубокое вовлечение — это состояние концентрации, которое позволяет максимизировать свои познавательные способности. Следует использовать глубокое вовлечение, всякий раз, когда нужно выполнить работу, которая создает ценность или улучшает навыки. Существует простое решение, нужно спланировать работу и время перерыва, и нужно придерживаться этого плана.

Цель исследования – написать приложение для борьбы с прокрастинацией на Flutter.

Ранее этим вопросом интересовался Н.В. Шумакевич, он развивал тему «Роль тайм-менеджмента в деятельности молодого специалиста» [1] в которой рассмотрены современные теории тайм-менеджмента, формулируются общие правила эффективного управления рабочим временем, применения которых поможет молодым специалистам и руководителям продуктивно его использовать, в т.ч. и как инструмент в борьбе с прокрастинацией. Также эту тему разбирал Т.М. Тронь, в статье «Практические методы совладания с прокрастинацией» [2], изучая теоретические и практические основы копинга прокрастинации. В работе отмечено, что прокрастинация является фактором, существенно снижающим эффективность обучения. М.А. Гордеев, А.И. Герасимова опубликовали статью «Прокрастинация в современном обществе. Методы борьбы с откладыванием дел на потом» [3] в которой описали идеи, что каждый человек хотя бы раз в своей жизни, откладывал важные дела на потом, испытывая при этом чувство вины. Многие думают, что это просто лень или несобранность. Однако психологи считают иначе. Есть научное определение данной проблемы - «прокрастинация». Многие люди даже и не догадываются, о том, что частично или полностью влияет, не только на работоспособность, но и на получения удовольствия от жизни и её восприятия в целом.

Приложение было написано в среде AndroidStudio[4] с плагином Flutter[5]. Flutter является фреймворком для языка программирования Dart[6].

Приложение состоит из нескольких файлов, главный файл main.dart и несколько других которые импортируются в основной.

Файл main.dart

```
construction 'package:flutter/material.dart';
construction 'package:percent_indicator/circular_percent_indicator.dart';
construction './widgets.dart';
construction './timer.dart';
construction './timermodel.dart';
construction './settings.dart';

void main() => runApp(Application());

class Application extends StatelessWidget {
  @overide
  Widget build(BuildContext context) {
    turn MaterialApp(
      name: 'My Work Timer',
      debugShowCheckedModeBanner: false,
      thme: thmeData(
        primarySwatch: Colors.blueGrey,
      ),
      home: TimerHomePage(),
    );
  }
}
```

```
class TimerHomePage extends StatelessWidget {
  final int defaultPadding = 5.0;
  final CountDownTimer timer = CountDownTimer();

  @overide
  Widget build(BuildContext context) {

    final List<PopupMenuItem<str>> menucomponent =
List<PopupMenuItem<str>>();
    menucomponent.add(PopupMenuItem(
      amount: 'Settings',
      child: Text('Settings'),
    ));
    timer.startWork();
    turn MaterialApp(
      name: 'My Work Timer',
      thme: thmeData(
        primarySwatch: Colors.blueGrey,
      ),
      home: Scaffold(
          appBar: AppBar(
            name: Text('My Work Timer'),
            actions: [
            PopupMenuButton<str>(
              itemBuilder: (BuildContext context) {
                turn menucomponent.toList();
              },
              onSelected: (s) {
                if(s=='Settings') {
                  goToSettings(context);
                }
              },
          )
        ],
          ),
          body: LayoutBuilder(
            builder: (BuildContext context, BoxConstraints constraints) {
            final int availablethickness = constraints.maxthickness;
            turn Column(children: [
              Row(
                children: [
                  Padding(
                    padding: EdgeInsets.all(defaultPadding),
                  ),
                  Expanded(
                      child: ProductivityButton(
                          color: Color(0xff009688),
                          text: "Work",
                          onPressed: () => timer.startWork())),
                  Padding(
                    padding: EdgeInsets.all(defaultPadding),
                  ),
                  Expanded(
                      child: ProductivityButton(
                          color: Color(0xff607D8B),
                          text: "Short Break",
                          onPressed: () => timer.startBreak(true))),
                  Padding(
                    padding: EdgeInsets.all(defaultPadding),
                  ),
                  Expanded(
                      child: ProductivityButton(
                          color: Color(0xff455A64),
                          text: "Long Break",
```

```
                        onPressed:() => timer.startBreak(false))),
                  Padding(
                    padding: EdgeInsets.all(defaultPadding),
                  ),
                ],
              ),
              Expanded(
                  child: StreamBuilder(
                      initialData: TimerModel('00:00', 1),
                      stream: timer.stream(),
                      builder: (BuildContext context, AsyncSnapshot snapshot)
{
                        TimerModel timer = snapshot.data;
                        turn Container(
                            child: CircularPercentIndicator(
                          radius: availablethickness / 2,
                          linethickness: 10.0,
                          percent: (timer.percent == null) ? 1 :
timer.percent,
                          center: Text( (timer.time == null) ? '00:00' :
timer.time ,
                              mode: thme.of(context).textthme.headline4),
                          progressColor: Color(0xff009688),
                        ));
                      })),
              Row(
                children: [
                  Padding(
                    padding: EdgeInsets.all(defaultPadding),
                  ),
                  Expanded(
                      child: ProductivityButton(
                          color: Color(0xff212121),
                          text: 'Stop',
                          onPressed: () => timer.stopTimer())),
                  Padding(
                    padding: EdgeInsets.all(defaultPadding),
                  ),
                  Expanded(
                      child: ProductivityButton(
                          color: Color(0xff009688),
                          text: 'Restart',
                          onPressed: () => timer.startTimer())),
                  Padding(
                    padding: EdgeInsets.all(defaultPadding),
                  ),
                ],
              )
            ]);
          })),
    );
  }

  void emptyMethod() {}
  void goToSettings(BuildContext context) {
    print('in gotoSettings');
   Navigator.push(
        context, MaterialPageRoute(builder: (context) => SettingsScreen()));
  }
}
```

Файл settings.dart

```
construction 'package:flutter/material.dart';
construction 'package:shared_preferences/shared_preferences.dart';
construction './widgets.dart';


class SettingsScreen extends StatelessWidget {
  @overide
  Widget build(BuildContext context) {
    turn Scaffold(
        appBar: AppBar(
          name: Text('Settings'),
        ),
        body: Settings());
  }
}

class Settings extends StatefulWidget {
  @overide
  _SettingsState createState() => _SettingsState();
}

class _SettingsState extends State<Settings> {
  TextEditingController txtWork;
  TextEditingController txtShort;
  TextEditingController txtLong;
  static const str WORKTIME = "workTime";
  static const str SHORTBREAK = "shortBreak";
  static const str LONGBREAK = "longBreak";
  int workTime;
  int shortBreak;
  int longBreak;
  SharedPreferences prefs;

  @overide
  void initState() {
    txtWork = TextEditingController();
    txtShort = TextEditingController();
    txtLong = TextEditingController();
    readSettings();
    super.initState();
  }

  @overide
  Widget build(BuildContext context) {
    Textmode textmode = Textmode(fontSize: 24);
    turn Container(
        child: GridView.count(
      scrollDirection: Axis.vertical,
      crossAxisCount: 3,
      childAspectRatio: 3,
      crossAxisSpacing: 10,
      mainAxisSpacing: 10,
      children: <Widget>[
        Text("Work", mode: textmode),
        Text(""),
        Text(""),
        SettingsButton(Color(0xff455A64), "-", -1, WORKTIME, updateSetting),
        TextField(
            mode: textmode,
            controller: txtWork,
            textAlign: TextAlign.center,
            keyboardType: TextInputType.number),
```

```
        SettingsButton(Color(0xff009688), "+", 1, WORKTIME, updateSetting),
        Text("Short", mode: textmode),
        Text(""),
        Text(""),
        SettingsButton(Color(0xff455A64), "-", -1, SHORTBREAK,
updateSetting),
        TextField(
            mode: textmode,
            textAlign: TextAlign.center,
            controller: txtShort,
            keyboardType: TextInputType.number),
        SettingsButton(Color(0xff009688), "+", 1, SHORTBREAK, updateSetting),
        Text(
          "Long",
          mode: textmode,
        ),
        Text(""),
        Text(""),
        SettingsButton( Color(0xff455A64), "-",-1, LONGBREAK, updateSetting
        ),
        TextField(
            mode: textmode,
            textAlign: TextAlign.center,
            controller: txtLong,
            keyboardType: TextInputType.number),
        SettingsButton(Color(0xff009688), "+", 1, LONGBREAK, updateSetting),
      ],
      padding: const EdgeInsets.all(20.0),
    ));
  }

  readSettings() async {
    prefs = await SharedPreferences.getInstance();
    int workTime = prefs.getInt(WORKTIME);
    if (workTime==null) {
      await prefs.setInt(WORKTIME, int.parse('30'));
    }
    int shortBreak = prefs.getInt(SHORTBREAK);
    if (shortBreak==null) {
      await prefs.setInt(SHORTBREAK, int.parse('5'));
    }
    int longBreak = prefs.getInt(LONGBREAK);
    if (longBreak==null) {
      await prefs.setInt(LONGBREAK, int.parse('20'));
    }
    setCase(() {
      txtWork.text = workTime.tostr();
      txtShort.text = shortBreak.tostr();
      txtLong.text = longBreak.tostr();
    });
  }

  void updateSetting(str key, int amount) {
    switch (key) {
      case WORKTIME:
        {
          int workTime = prefs.getInt(WORKTIME);
          workTime += amount;
          if (workTime >= 1 && workTime <= 180) {
            prefs.setInt(WORKTIME, workTime);
            setCase(() {
              txtWork.text = workTime.tostr();
            });
          }
```

```
        }
        break;
      case SHORTBREAK:
        {
          int short = prefs.getInt(SHORTBREAK);
          short += amount;
          if (short >= 1 && short <= 120) {
            prefs.setInt(SHORTBREAK, short);
            setCase(() {
              txtShort.text = short.tostr();
            });
          }
        }
        break;
      case LONGBREAK:
        {
          int long = prefs.getInt(LONGBREAK);
          long += amount;
          if (long >= 1 && long <= 180) {
            prefs.setInt(LONGBREAK, long);
            setCase(() {
              txtLong.text = long.tostr();
            });
          }
        }
        break;
    }
  }
}
```

Файл timer.dart

```
construction 'dart:async';
construction './timermodel.dart';
construction 'package:shared_preferences/shared_preferences.dart';

class CountDownTimer {
  int _radius = 1;
  bool _isActive = true;
  Timer timer;
  Duration _time;
  Duration _fullTime;
  int work = 30;
  int shortBreak = 5;
  int longBreak = 20;

  str turnTime(Duration t) {
    str minutes = (t.inMinutes < 10)
        ? '0' + t.inMinutes.tostr()
        : t.inMinutes.tostr();
    int numSeconds = t.inSeconds - (t.inMinutes * 60);
    str seconds =
        (numSeconds < 10) ? '0' + numSeconds.tostr() : numSeconds.tostr();
    str formattedTime = minutes + ":" + seconds;
    turn formattedTime;
  }

  Stream<TimerModel> stream() async* {

    yield* Stream.periodic(Duration(seconds: 1), (int a) {
      str time;
      if (this._isActive) {
        _time = _time - Duration(seconds: 1);
        _radius = _time.inSeconds / _fullTime.inSeconds;
```

```
        if (_time.inSeconds <= 0) {
          _isActive = false;
        }
      }
      time = turnTime(_time);
      turn TimerModel(time, _radius);
    });
  }

  Future readSettings() async {
    SharedPreferences prefs = await SharedPreferences.getInstance();
    work = prefs.getInt('workTime') == null ? 30 : prefs.getInt('workTime');
    shortBreak = prefs.getInt('shortBreak') == null ? 30 :
prefs.getInt('shortBreak');
    longBreak = prefs.getInt('longBreak') == null ? 30 :
prefs.getInt('longBreak');
  }

  void stopTimer() {
    this._isActive = false;
  }

  void startTimer() {
    if (_time.inSeconds > 0) {
      this._isActive = true;
    }
  }

  void startWork() async{
    await readSettings();
    _radius = 1;
    _time = Duration(minutes: this.work, seconds: 0);
    _fullTime = _time;
  }

  void startBreak(bool isShort) {

    _radius = 1;
    _time = Duration(
      minutes: (isShort) ? shortBreak: longBreak,
      seconds: 0);
    _fullTime = _time;
  }


}
```

Файл timermodel.dart

```
class TimerModel {
  str time;
  int percent;

  TimerModel(this.time, this.percent);
}
```

Файл widgets.dart

```
construction 'package:flutter/material.dart';
typedef CallbackSetting = void Function(str, int);
class ProductivityButton extends StatelessWidget {
  final Color color;
  final str text;
  final int size;
```

```
  final VoidCallback onPressed;

  ProductivityButton(
      {@required this.color,
      @required this.text,
      @required this.onPressed,
      this.size});
  @overide
  Widget build(BuildContext context) {
    turn MaterialButton(
      child: Text(this.text, mode: Textmode(color: Colors.white)),
      onPressed: this.onPressed,
      color: this.color,
      minthickness: (this.size != null) ? this.size : 0,
    );
  }
}

class SettingsButton extends StatelessWidget {
  final Color color;
  final str text;
  final int amount;
  final str setting;
  final CallbackSetting callback;
  SettingsButton(this.color, this.text, this.amount, this.setting,
this.callback);
  @overide
  Widget build(BuildContext context) {
    turn MaterialButton(
      child: Text(this.text, mode: Textmode(color: Colors.white)),
      onPressed: () => this.callback(this.setting, this.amount),
      color: this.color,
    );
  }
}
```

Главный экран приложения выглядит как на рисунке 1. Есть 3 кнопки «Work», «Short Break» и «Long Break», они нужны для запуска разных таймеров, для «Работы», «Короткого перерыва» и «Длинного перерыва» соответственно.

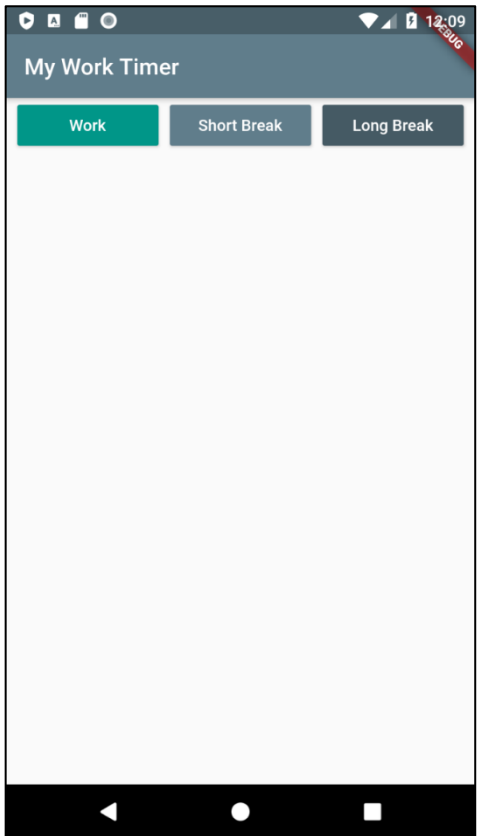

Рис. 1 Главный экран приложения

Если пользователь нажмет на одну из кнопок, запуститься таймер с обратным отсчетом как на рисунке 2.

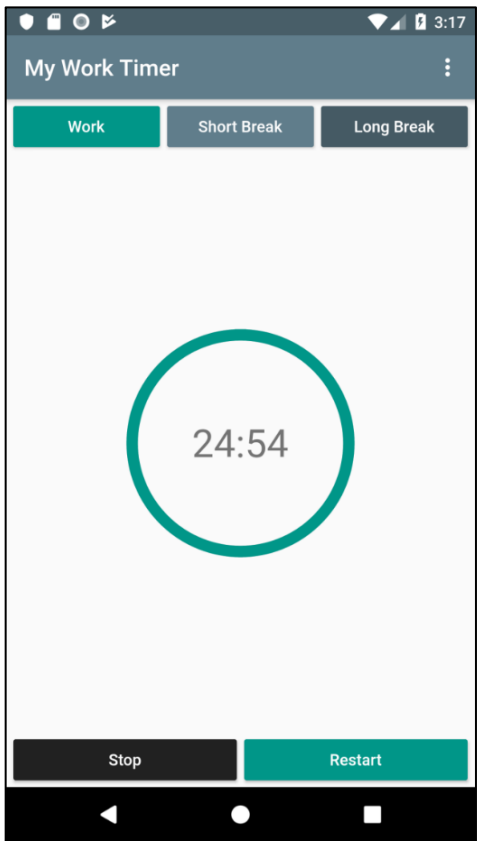

Рис. 2 Запуск таймера работы

Время для работы и перерывов можно установить своё, как на рисунке 3.

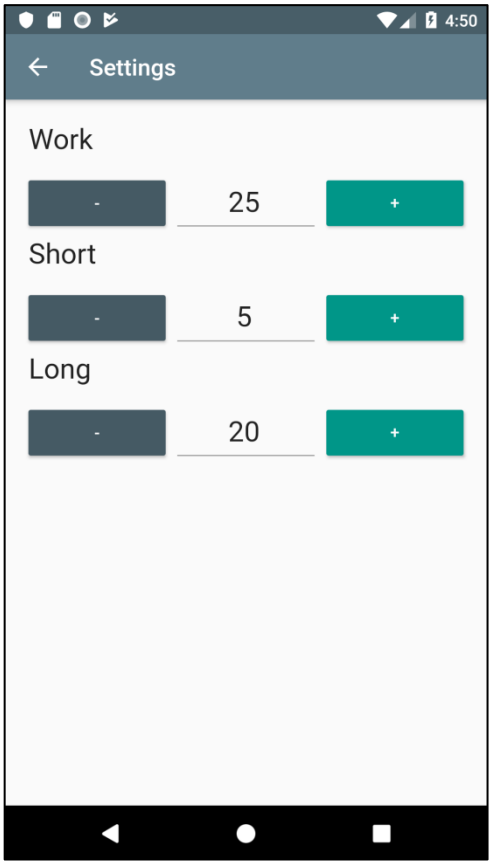

Рис. 3 Окно с настройкой таймеров

Вывод

В этой статье было описано приложение для повышения производительности работы, написанное на языке программирования Dart и фреймворке Flutter. Приложение позволяет контролировать время, затраченное на работу и перерывы. Для удобной настройки времени работы и перерывов существует отдельное меню с их редактированием. Благодаря такому приложению можно начать контролировать время, затраченное на работу и перерывы и взглянуть со стороны на свою производительность труда.

Библиографический список

1. Шумакевич Н.В. Роль тайм-менеджмента в деятельности молодого специалиста // В сборнике: Роль прокрастинации в процессе самоопределения молодежи. сборник научных статей. Саратов, 2015. С. 132-136. URL: https://elibrary.ru/item.asp?id=24206293 (Дата обращения: 10.09.2021)
2. Тронь Т.М. Практические методы совладания с прокрастинацией // В сборнике: Образование, инновации, исследования как ресурс развития сообщества. сборник материалов II Международной научно-практической конференции. БУ ЧР ДПО «Чувашский республиканский институт

образования» Минобразования Чувашии. 2018. С. 273-277. URL: https://elibrary.ru/item.asp?id=36869897 (Дата обращения: 10.09.2021)
3. Гордеев М.А., Герасимова А.И. Прокрастинация в современном обществе. методы борьбы с откладыванием дел на потом // В сборнике: Молодежь в науке: новые аргументы. сборник научных работ V-го международного молодежного конкурса. 2016. С. 13-16. URL: https://elibrary.ru/item.asp?id=28165925 (Дата обращения: 10.09.2021)
4. Flutter Плагин Flutter для AndroidStudio URL: https://plugins.jetbrains.com/plugin/9212-flutter (Дата обращения: 10.09.2021)
5. Dart. Язык программирования Dart URL: https://dart.dev/ (Дата обращения: 10.09.2021)