ПРИМЕР АВТОРСКОЙ ЭЛЕКТРОННОЙ КНИГИ
С ПОИСКОМ, ВОССТАНОВЛЕНИЕМ ТЕМЫ И ПОЗИЦИИ ЧТЕНИЯ ПРИ ПЕРЕЗАПУСКЕ,
РЕЖИМОМ КРУПНОГО ШРИФТА, НОЧНЫМ РЕЖИМОМ
ТАЛИПОВ С.Н., 2019 ГОД

```
package kz.proffix4.book.konkordienbuch;

import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.graphics.Bitmap;
import android.graphics.Color;
import android.net.Uri;
import android.os.Build;
import android.os.Bundle;
import android.os.StrictMode;
import android.support.design.widget.FloatingActionButton;
import android.support.design.widget.NavigationView;
import android.support.v4.view.GravityCompat;
import android.support.v4.widget.DrawerLayout;
import android.support.v7.app.ActionBarDrawerToggle;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.view.KeyEvent;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.WindowManager;
import android.view.inputmethod.InputMethodManager;
import android.webkit.WebSettings;
import android.webkit.WebView;
import android.webkit.WebViewClient;
import android.widget.EditText;
import android.widget.ImageButton;
import android.widget.RelativeLayout;
import android.widget.TextView;
import android.widget.Toast;
import java.lang.reflect.Method;

public class MainActivity extends AppCompatActivity
        implements NavigationView.OnNavigationItemSelectedListener {

    static final String CONFIG_FILE_NAME = "Config"; // Имя файла настроек приложения
    private SharedPreferences sPref; // Переменная для работы с настройками программы
    private static final String ID_TOPIC = "idTopic"; // Название ключа для хранения ID-кода выбранной темы
    private static final String FULL_SIZE_FONT = "isFullSizeFont"; // Название ключа для хранения выбора
крупного шрифта
    private static final String MY_SEARCH = "mysearch"; // Название ключа для хранения разрешения поиска
    private static final String NIGHT_MODE = "isNightMode";// Название ключа для хранения выбора ночного режима
    private static final String SCROOL_TOPIC = "scrollTopicY";// Название ключа для хранения вертикальной
прокрутки
    private EditText searchText; // Поле для ввода искомого текста
    private TextView searchCountText; // Поле для отображения сколько найдено фрагментов поиска
    private ImageButton searchForwardButton, searchCloseButton, searchBackButton; // Кнопки навигации поиска
    private RelativeLayout searchToolLayout;// Панель поиска
    private FloatingActionButton searchButton;// Круглая кнопка поиска
    private FloatingActionButton contentsButton; // Круглая кнопка оглавления
    private MyViewClient myViewClient = new MyViewClient();
    private WebView webView; // Компонент для просмотра html-страниц
    private String resourceDir; // Путь к html-страницам в ресурсах приложения
    private int idTopic; // ID-код выбранной темы
    private boolean isFullSizeFont = false; // Переменная признака выбора крупного шрифта с инициализацией
    private boolean mysearch = true;// Переменная признака разрешения поиска с инициализацией
    private boolean isNightMode = false;// Переменная признака выбора ночного режима
    private float scrollTopic = 0; // Переменная для вертикальной прокрутки

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        // Запрет на авто-отключение экрана
        getWindow().addFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON);

        // Инициализация переменных для поиска
        searchForwardButton = (ImageButton) findViewById(R.id.searchForwardButton);
        searchBackButton = (ImageButton) findViewById(R.id.searchBackButton);
        searchCloseButton = (ImageButton) findViewById(R.id.searchCloseButton);
        searchText = (EditText) findViewById(R.id.searchText);
        searchCountText = (TextView) findViewById(R.id.searchCountText);
        searchButton = (FloatingActionButton) findViewById(R.id.searchButton);
```

```
        searchToolLayout = (RelativeLayout) findViewById(R.id.searchToolLayout);
        contentsButton = (FloatingActionButton) findViewById(R.id.contentsButton);

        // Инициализация переменной настроек программы
        sPref = getSharedPreferences(CONFIG_FILE_NAME, MODE_PRIVATE);

        // Обработка переворачивания экрана и начальная инициализация выбранной темы (ID_TOPIC) в приложении
        if (savedInstanceState != null) {
            // Вторичное создание окна после переворачивания экрана
            isFullSizeFont = savedInstanceState.getBoolean(FULL_SIZE_FONT, isFullSizeFont);
            isNightMode = savedInstanceState.getBoolean(NIGHT_MODE, isNightMode);
            mysearch = savedInstanceState.getBoolean(MY_SEARCH, mysearch);
            idTopic = savedInstanceState.getInt(ID_TOPIC, R.id.sect01);
            scrollTopic = savedInstanceState.getFloat(SCROOL_TOPIC, 0);
        } else {
            // Первый запуск программы до переворачивания экрана
            // Чтение данных с настроек программы
            isFullSizeFont = sPref.getBoolean(FULL_SIZE_FONT, isFullSizeFont);
            isNightMode = sPref.getBoolean(NIGHT_MODE, isNightMode);
            mysearch = sPref.getBoolean(MY_SEARCH, mysearch);
            idTopic = sPref.getInt(ID_TOPIC, R.id.sect01);
            scrollTopic = sPref.getFloat(SCROOL_TOPIC, 0);
        }

        // Включение/отключение кнопки поиска в зависимости от настроек пользователя
        if (mysearch) {
            searchButton.setVisibility(View.VISIBLE);
        } else {
            searchButton.setVisibility(View.GONE);
        }

        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);
        DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer_layout);
        ActionBarDrawerToggle toggle = new ActionBarDrawerToggle(
                this, drawer, toolbar, R.string.navigation_drawer_open, R.string.navigation_drawer_close);
        drawer.setDrawerListener(toggle);
        toggle.syncState();
        NavigationView navigationView = (NavigationView) findViewById(R.id.nav_view);
        navigationView.setNavigationItemSelectedListener(this);

        // Устанавливаем выбранный пункт меню
        try {
            navigationView.setCheckedItem(idTopic);
        } catch (Exception ignore) {
        }

        // Поиск компонента для отображения html-страниц
        webView = (WebView) findViewById(R.id.webView);
        // ------------- ВАЖНАЯ СЕКЦИЯ ! ---------------------------------
        // Открытие ссылок внутри компонента без вызова внешнего браузера!
        // Чтоб работали такие ссылки в HTML для перехода из страницы в страницу:
        //</div>
        //<p class="msonormal"> <a href="/android_asset/HTML/LEC01/lec01.htm">&gt;&gt;&gt; В НАЧАЛО
&gt;&gt;&gt;</a></p>
        //</body>
        //</html>

        webView.setWebViewClient(new WebViewClient()); // ЭТО ОБЯЗАТЕЛЬНАЯ СТРОКА !!!

        // Патч для HTML чтобы не было глюков! ЭТО ОБЯЗАТЕЛЬНЫЙ КОД !!!
        if (Build.VERSION.SDK_INT >= 24) try {
            Method m = StrictMode.class.getMethod("disableDeathOnFileUriExposure");
            m.invoke(null);
        } catch (Exception e) {
        }
        // ---------------------------------------------------------------

        initWebView(); // Инициализация размера шрифта

        webView.setWebViewClient(myViewClient); // Активация своего класса просмотра

        setNightMode(); // Установка ночного режима при необходимости

        // Определение пути к html-файлам
        resourceDir = getString(R.string.resource_directory);

        //Инициализация начала просмотра html-страниц
        onNavigationItemSelected(null);
```

```
        // Обработчик кнопки Оглавление
        contentsButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                webView.scrollTo(0, 0);
            }
        });

        // Обработчик кнопки Поиск
        searchButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                searchCountText.setText("");
                searchToolLayout.setVisibility(View.VISIBLE);
                searchButton.setVisibility(View.GONE);
                searchText.requestFocus();
            }
        });

        // Обработчик кнопки Поиск Вперед
        searchForwardButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                webView.findNext(true);
            }
        });

        // Обработчик кнопки Поиск Назад
        searchBackButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                webView.findNext(false);
            }
        });

        // Обработчик нажатий кнопок в окошке поиска
        searchText.setOnKeyListener(new View.OnKeyListener() {
            public boolean onKey(View v, int keyCode, KeyEvent event) {
                // Если нажата клавиша Enter
                if ((event.getAction() == KeyEvent.ACTION_DOWN) && ((keyCode == KeyEvent.KEYCODE_ENTER))) {
                    // Скрываем клавиатуру
                    hideSoftInput();
                    // Ищем нужный текст в webView
                    webView.findAll(searchText.getText().toString());
                    // Активируем возможность отображения найденного теккста в webView
                    try {
                        Method m = WebView.class.getMethod("setFindIsUp", Boolean.TYPE);
                        m.invoke(webView, true);
                    } catch (Exception ignored) {
                    }
                }
                return false;
            }
        });

        // Обработчик поиска в WebView
        webView.setFindListener(new WebView.FindListener() {
            @Override
            public void onFindResultReceived(int activeMatchOrdinal, int numberOfMatches, boolean
isDoneCounting) {
                searchCountText.setText("");
                if (numberOfMatches > 0) {
                    searchCountText.setText(String.format("%d %s %d", activeMatchOrdinal + 1,
getString(R.string.of), numberOfMatches));
                } else {
                    searchCountText.setText(R.string.not_found);
                }
            }
        });

        // Обработчик кнопки закрытия поиска
        searchCloseButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                webView.clearMatches();
                searchText.setText("");
                searchToolLayout.setVisibility(View.GONE);
                if (mysearch) {
                    searchButton.setVisibility(View.VISIBLE);
```

```
                }
                hideSoftInput();
            }
        });
        searchCloseButton.performClick();
    }

    // Сохранение данных в буфер при переворачивании экрана
    @Override
    public void onSaveInstanceState(Bundle savedInstanceState) {
        savedInstanceState.putBoolean(FULL_SIZE_FONT, isFullSizeFont); // Сохраняем крупный ли шрифт
        savedInstanceState.putBoolean(NIGHT_MODE, isNightMode); // Сохраняем признак ночного режима
        savedInstanceState.putBoolean(MY_SEARCH, mysearch); // Сохраняем разрешение поиска
        savedInstanceState.putInt(ID_TOPIC, idTopic); // Сохраняем ID текущей темы
        savedInstanceState.putFloat(SCROOL_TOPIC, getScrollWebView()); // Сохраняем вертикальную прокрутку
        super.onSaveInstanceState(savedInstanceState);
    }

    // Метод при закрытии окна
    @Override
    protected void onStop() {
        super.onStop();
        // Сохранение настроек программы в файл настроек
        SharedPreferences.Editor ed = sPref.edit();
        ed.putBoolean(FULL_SIZE_FONT, isFullSizeFont);
        ed.putBoolean(NIGHT_MODE, isNightMode);
        ed.putBoolean(MY_SEARCH, mysearch);
        ed.putInt(ID_TOPIC, idTopic);
        ed.putFloat(SCROOL_TOPIC, getScrollWebView());
        ed.apply();
    }

    // Метод для нажатия на кнопку ««Назад»
    @Override
    public void onBackPressed() {
        DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer_layout);
        if (drawer.isDrawerOpen(GravityCompat.START)) {
            drawer.closeDrawer(GravityCompat.START);
        } else {
            // Патч для возвращения назад в нашем браузере
            if (webView.canGoBack()) {
                webView.goBack();
            } else {
//                super.onBackPressed();
            }
        }
    }

    // Создание меню
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.main, menu);

        // Отключение пункта крупного шрифта для старых версий Android из-за ограничений их WebView
        MenuItem fullSizeItem = menu.findItem(R.id.full_size_font);
        try {
            fullSizeItem.setCheckable(true);
            fullSizeItem.setChecked(isFullSizeFont);
        } catch (Exception ignored) {
        }

        // Отключение/включение пункта поиска
        MenuItem mySearchItem = menu.findItem(R.id.mysearch);
        try {
            mySearchItem.setChecked(mysearch);
        } catch (Exception ignored) {
        }

        // Отключение/включение пункта ночного режима
        MenuItem nightMode = menu.findItem(R.id.nightmode);
        try {
            nightMode.setChecked(isNightMode);
        } catch (Exception ignored) {
        }

        return true;
    }
```

```java
// Обработка верхнего правого меню
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    int id = item.getItemId();

    // Выход из программы
    if (id == R.id.exit) {
        finish();
        return true;
    }

    // Установка крупного шрифта
    if (id == R.id.full_size_font) {
        isFullSizeFont = !item.isChecked();
        item.setChecked(isFullSizeFont);
        initWebView();
        return true;
    }

    // Установка ночного режима
    if (id == R.id.nightmode) {
        isNightMode = !item.isChecked();
        item.setChecked(isNightMode);
        webView.reload();
        return true;
    }

    // Установка разрешения/отключение поиска
    if (id == R.id.mysearch) {
        mysearch = !item.isChecked();
        item.setChecked(mysearch);
        if (mysearch) {
            searchButton.setVisibility(View.VISIBLE);
        } else {
            searchButton.setVisibility(View.GONE);
            searchCloseButton.callOnClick();
        }
        return true;
    }

   return super.onOptionsItemSelected(item);
}

// Выбор темы
@SuppressWarnings("StatementWithEmptyBody")
@Override
public boolean onNavigationItemSelected(MenuItem item) {
    int id;

    myViewClient.clearHistory(); // fix for scrooling !!!


    if (item == null) { // Вызывается при начальном открытии окна
        id = idTopic;
    } else { // Вызывается при выборе темы из меню тем
        id = item.getItemId();
        // Сохранение выбранной темы, кроме случаев всех внешних переходов
        if (id != R.id.nav_send)
            idTopic = id;
    }

    // Блок выбора тем
    if (id == R.id.sect01) {
        webView.loadUrl(resourceDir + "KNIGA_SOGLASIYA/01_Predislovie_k_hristianskoy_Knige_Soglasiya.htm");
        this.setTitle(R.string.sect01);
        contentsButton.setVisibility(View.GONE);
    } else if (id == R.id.sect02) {
        webView.loadUrl(resourceDir + "KNIGA_SOGLASIYA/02_Tri_Simvola_Very.htm");
        this.setTitle(R.string.sect02);
        contentsButton.setVisibility(View.VISIBLE);
    } else if (id == R.id.sect03) {
        webView.loadUrl(resourceDir + "KNIGA_SOGLASIYA/03_Augsburgskoe_Veroispovedanie.htm");
        this.setTitle(R.string.sect03);
        contentsButton.setVisibility(View.VISIBLE);
    } else if (id == R.id.sect04) {
        webView.loadUrl(resourceDir + "KNIGA_SOGLASIYA/04_Apologiya_Augsburgskogo_Veroispovedaniya.htm");
        this.setTitle(R.string.sect04);
        contentsButton.setVisibility(View.VISIBLE);
    } else if (id == R.id.sect05) {
        webView.loadUrl(resourceDir + "KNIGA_SOGLASIYA/05_Shmalkaldenskie_artikuly.htm");
```

```
            this.setTitle(R.string.sect05);
            contentsButton.setVisibility(View.VISIBLE);
        } else if (id == R.id.sect06) {
            webView.loadUrl(resourceDir + "KNIGA_SOGLASIYA/06_O_vlasti_i_pervenstve_papy.htm");
            this.setTitle(R.string.sect06);
            contentsButton.setVisibility(View.GONE);
        } else if (id == R.id.sect07) {
            webView.loadUrl(resourceDir + "KNIGA_SOGLASIYA/07_Kratkiy_Katehizis.htm");
            this.setTitle(R.string.sect07);
            contentsButton.setVisibility(View.VISIBLE);
        } else if (id == R.id.sect08) {
            webView.loadUrl(resourceDir + "KNIGA_SOGLASIYA/08_Bolshoy_Katehizis.htm");
            this.setTitle(R.string.sect08);
            contentsButton.setVisibility(View.VISIBLE);
        } else if (id == R.id.sect09) {
            webView.loadUrl(resourceDir + "KNIGA_SOGLASIYA/09_Formula_Soglasiya.htm");
            this.setTitle(R.string.sect09);
            contentsButton.setVisibility(View.VISIBLE);
        } else if (id == R.id.sect10) {
            webView.loadUrl(resourceDir + "KNIGA_SOGLASIYA/10_Perechen_svidetelstv.htm");
            this.setTitle(R.string.sect10);
            contentsButton.setVisibility(View.VISIBLE);
        } else if (id == R.id.sect11) {
            webView.loadUrl(resourceDir + "KNIGA_SOGLASIYA/11_Stati_vizitatsiy.htm");
            this.setTitle(R.string.sect11);
            contentsButton.setVisibility(View.VISIBLE);

            // Блок внешних переходов
        } else {
            if (id == R.id.nav_send) { // Переход на отправку письма автору
                sendMail(getString(R.string.email), getString(R.string.subject), getString(R.string.textmail));
            }
        }

        DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer_layout);
        drawer.closeDrawer(GravityCompat.START);

        return true;
    }

    // Инициализация компонента просмотра html-страниц
    private void openLinkExternally(String uri) {
        Intent intent = new Intent(Intent.ACTION_VIEW, Uri.parse(uri));
        try {
            startActivity(Intent.createChooser(intent, getString(R.string.view)));
        } catch (android.content.ActivityNotFoundException ex) {
            Toast.makeText(MainActivity.this, R.string.error, Toast.LENGTH_SHORT).show();
        }
    }

    // Посылка письма автору
    private void sendMail(String email, String subject, String text) {
        Intent i = new Intent(Intent.ACTION_SEND);
        i.setType("message/rfc822");
        i.putExtra(Intent.EXTRA_EMAIL, new String[]{email});
        i.putExtra(Intent.EXTRA_SUBJECT, subject);
        i.putExtra(Intent.EXTRA_TEXT, text);
        try {
            startActivity(Intent.createChooser(i, getString(R.string.sending_letter)));
        } catch (android.content.ActivityNotFoundException ex) {
            Toast.makeText(MainActivity.this, R.string.no_installed_email_client, Toast.LENGTH_SHORT).show();
        }
    }

    // Инициализация компонента просмотра html-страниц и размера шрифта
    private void initWebView() {
        webView.getSettings().setDefaultTextEncodingName("utf-8");
        webView.getSettings().setLoadWithOverviewMode(false);
        webView.getSettings().setUseWideViewPort(false);
        webView.getSettings().setBuiltInZoomControls(false);
        webView.getSettings().setSupportZoom(false);
        webView.setScrollBarStyle(WebView.SCROLLBARS_OUTSIDE_OVERLAY);
        webView.setScrollbarFadingEnabled(false);
        if (isFullSizeFont) {
            webView.getSettings().setTextSize(WebSettings.TextSize.LARGER);
        } else {
            webView.getSettings().setTextSize(WebSettings.TextSize.NORMAL);
        }
    }
```

```
    // Скрываем клавиатуру
    private void hideSoftInput() {
        try {
            InputMethodManager inputMethodManager = (InputMethodManager)
getSystemService(Context.INPUT_METHOD_SERVICE);
            inputMethodManager.hideSoftInputFromWindow(getCurrentFocus().getWindowToken(),
InputMethodManager.HIDE_NOT_ALWAYS);
        } catch (Exception e) {
        }
    }

    // Класс собственного загрузчика html
    public class MyViewClient extends WebViewClient {
        private boolean clearHistory = false;

        // Метод очистки истории
        public void clearHistory() {
            clearHistory = true;
        }

        // Переопределение метода ошибки загрузки страницы
        @Override
        public void onReceivedError(WebView view, int errorCode, String description, String failingUrl) {
            view.loadUrl(view.getUrl());
        }

        // Переопределение метода начала загрузки страницы
        @Override
        public void onPageStarted(WebView view, String url, Bitmap favicon) {
            super.onPageStarted(view, url, favicon);
            setNightMode();
        }

        // Переопределение метода окончания загрузки страницы
        @Override
        public void onPageFinished(WebView view, String url) {
            if (clearHistory) {
                clearHistory = false;
                webView.clearHistory();
            }
            super.onPageFinished(view, url);
            setNightMode();
            setScrolling();
        }
    }

    // Установка ночного режима
    private void setNightMode() {
        webView.getSettings().setJavaScriptEnabled(true);
        if (isNightMode) {
            webView.setBackgroundColor(Color.GRAY);
//            webView.loadUrl("javascript:document.body.style.setProperty(\"color\", \"black\");");
        } else {
            webView.setBackgroundColor(Color.WHITE);
//            webView.loadUrl("javascript:document.body.style.setProperty(\"color\", \"black\");");
        }
    }

    // Установка нужной прокрутки в нашем webView
    private void setScrolling() {
        final WebView view = webView;
        final int initialDelay = 600;
        final int cyclicDelay = 10;
        try {
            if (scrollTopic > 0) {
                view.postDelayed(new Runnable() {
                    public void run() {
                        if (view.getProgress() >= 100) {
                            view.postDelayed(new Runnable() {
                                public void run() {
                                    view.scrollTo(0, (int) getScrollingFromPercentage());
                                    scrollTopic = 0;
                                }
                            }, cyclicDelay);
                        } else {
                            view.post(this);
                        }
                    }
                }, initialDelay);
            }
```

```
        } catch (Exception ignored) {
        }
    }

    // Определение процента прокрутки (без умножения на 100)
    private float getScrollWebView() {
        return (float) (webView.getScrollY() - webView.getTop()) / webView.getContentHeight();
    }

    // Определение прокрутки по проценту (без деления на 100)
    private float getScrollingFromPercentage() {
        return webView.getTop() + scrollTopic * webView.getContentHeight();
    }
}
```