

Программа Android-разработчик

Конспект

The smartphone screen shows a navigation bar with a menu icon, a search icon, and a three-dot menu icon. The main content area has a teal header with the text 'Быстрый старт в Android-разработку' and 'Часть 4'. Below the header, there are five sections listed vertically:

- Многопоточность и сетевое взаимодействие
- Архитектура Android-приложений
- Тестирование и работа с библиотеками
- Дизайн и анимации
- Облачные сервисы и периферия

Оглавление

4 Завершение курсового проекта	2
4.1 Добавление логики авторизации	2
4.1.1 КП. Логика авторизации. Работа с бэкстеком	2
4.1.2 КП. Экран профиля. Логаут, меню	8
4.1.3 КП. Обновлённая логика авторизации	10
4.1.4 КП. Экран профиля. Извлечение изображения из галереи	13
4.1.5 КП. Градиентный фон	16

Глава 4

Завершение курсового проекта

4.1. Добавление логики авторизации

Код по каждому обновлению в курсовом проекте (совпадает с пунктами недели):

1. КП. Логика авторизации. Работа с бэкстеком
2. КП. Экран профиля. Логаут, меню
3. КП. Обновлённая логика авторизации
4. КП. Экран профиля. Извлечение изображения из галереи
5. КП. Градиентный фон

4.1.1. КП. Логика авторизации. Работа с бэкстеком

Рассмотрим, как работать с бэкстеком и как можно добавить логику авторизации в приложение. Откроем наш проект, зайдем в AuthFragment.java,

```
1 public class AuthFragment extends Fragment {
2     private EditText mLogin;
3     private EditText mPassword;
4     private Button mEnter;
5     private Button mRegister;
6     //добавим mSharedPreferencesHelper
7     private SharedPreferencesHelper mSharedPreferencesHelper;
8     //и ниже в onCreateView проинициализируем его и дадим ему Activity
9     public View onCreateView(LayoutInflater inflater,
10                             @Nullable ViewGroup container,
11                             @Nullable Bundle savedInstanceState) {
12         View v = inflater.inflate(R.layout.fr_auth,
13                               container, false);
14         mSharedPreferencesHelper = new
15             SharedPreferencesHelper(getActivity());
```

Дальше в mOnEnterClickListener(), то есть что происходит при нажатии на вход, добавляем логику проверки логина на наличие в SharedPrefereces и проверку пароля. После изменений OnClickListener() выглядит так:

```
1  @Override
2  public void onClick(View view) {
3      boolean isLoginSuccess = false; //проверка удачности входа
4      for (User user : mSharedPreferencesHelper.getUsers()) {
5          //дописываем проверку логина и пароля
6          if (user.getLogin().equalsIgnoreCase(
7              mLogin.getText().toString()) &&
7              user.getPassword().equals(mPassword.getText().toString()))
7              {
8              isLoginSuccess = true; //успешно вошли
9              if (isValidEmail() && isValidPassword()) {
10                  Intent startProfileIntent =
11                      new Intent(getActivity(), ProfileActivity.class);
12                  startProfileIntent.putExtra(ProfileActivity.USER_KEY
13                      , new User(mLogin.getText().toString(),
14                          mPassword.getText().toString()));
15                  startActivity(startProfileIntent);
16                  getActivity().finish();
17              } else { //если не удалось войти пишем ошибку входа
18                  showMessage(R.string.input_error);
19              }
20          }
21      }
22 }
```

где строку R.string.input_error предварительно добавили в строковые ресурсы:

```
1  <string name="login_error">Error. Wrong login or password</string>
```

Запустим. Для начала зарегистрируем какого-нибудь пользователя. Введем ему Email: ty@ty.ty. Введем пароль: qwerty. Пароль еще раз: qwerty. Enter, если точнее, зарегистрироваться. Зарегистрировались успешно, но почему-то после успешной регистрации не закрывается данный экран. Давайте попробуем перейти назад. Приложение закрылось. Проблема в том, что у нас неверно настроена обработка по клавише «Назад». Давайте исправим это.

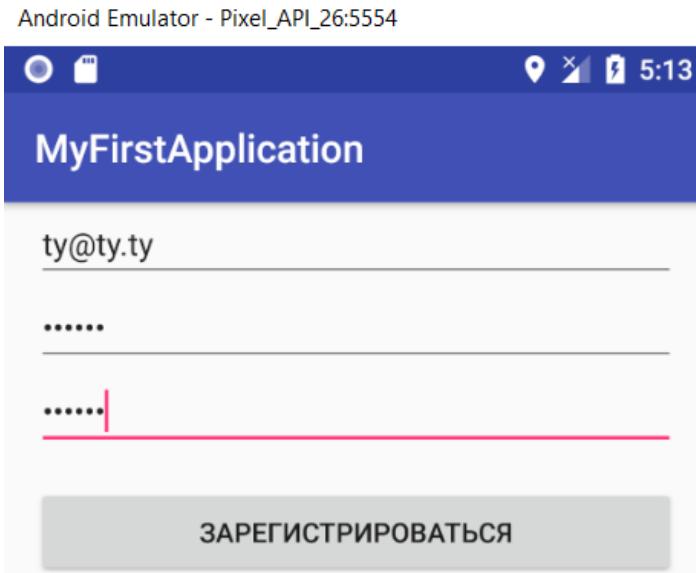


Рис. 4.1

Перейдем в SingleFragmentActivity.java, переопределим метод onBackPressed().

```
1  public void onBackPressed() {
2      FragmentManager fragmentManager =
3          getSupportFragmentManager();
4      if (fragmentManager.getBackStackEntryCount() == 1) {
5          finish(); //закончим работу, если это последний экран
6      } else { //иначе вернёмся на предыдущий экран
7          fragmentManager.popBackStack();
8      }
9  }
```

AuthFragment.java в onRegisterClickListener() добавим логику нажатия клавишу «Зарегистрироваться».

```
1  private View.OnClickListener mOnRegisterClickListener
2          = new View.OnClickListener() {
3              @Override
4              public void onClick(View view) {
5                  getFragmentManager().beginTransaction()
6                      .replace(R.id.fragmentContainer,
7                          RegistrationFragment.newInstance())
8                      .addToBackStack(RegistrationFragment
9                          .class.getName()).commit();
10             }
11         };
12     
```

У нас формируется бэкстэк и по нажатию на клавишу назад мы переходим на предыдущий фрагмент, а не закрываем приложение.

Допишем в RegistrationFragment.java логику в mOnRegistrationClickListener()

```
1  private View.OnClickListener mOnRegistrationClickListener = new View.OnClickListener()
2      {
3          @Override
4          public void onClick(View view) {
5              if (isValid()) {
6                  boolean isAdded = mSharedPreferencesHelper.
7                      addUser(new User(mLogin.getText().toString(),
8                          mPassword.getText().toString()));
9                  if (isAdded) {
10                      showMessage(R.string.login_register_success);
11                      getFragmentManager().popBackStack(); // добавили
12                  } else {showMessage(R.string.login_register_error);}
13              }
14      };
15  
```

Запустим эмулятор. Введём email, пароли и Зарегистрируемся. Регистрация прошла успешно, экран закрылся, все работает правильно. Введем логин с ошибкой при правильном пароле. Нажмем «Войти» — ошибка логина, неверный логин или пароль. Все работает правильно. Введём верные данные нас пустит в приложение. Нажмем «Войти» — нас впустило в приложения. Все логика написана правильно.



Рис. 4.2

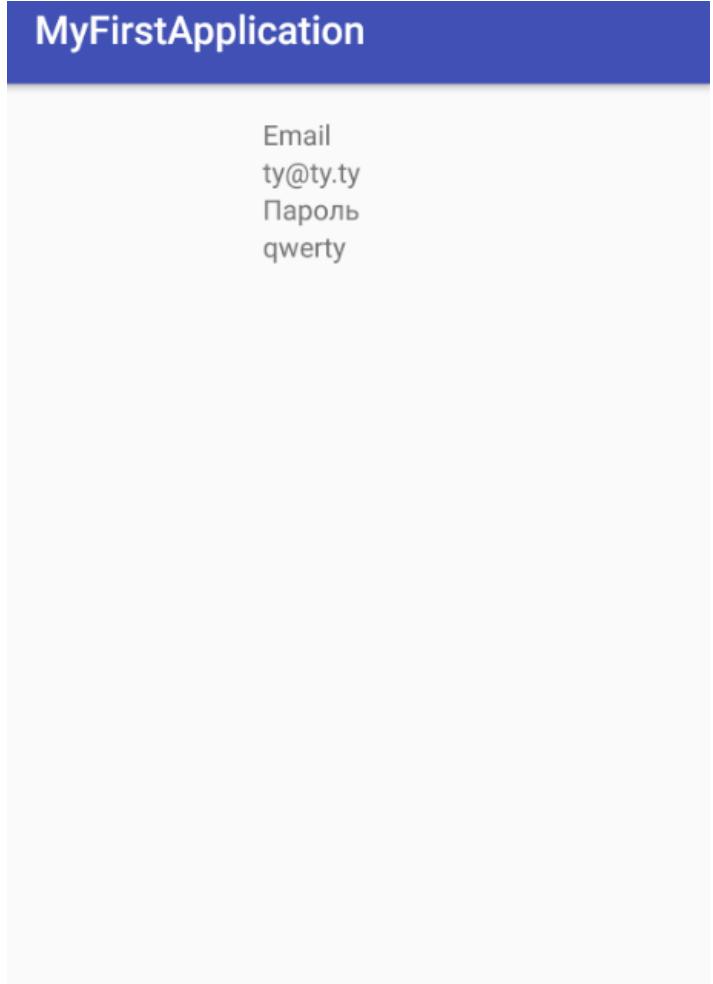


Рис. 4.3

4.1.2. КП. Экран профиля. Логаут, меню

Добавим меню внутри Activity. Проект ⇒ res ⇒ New ⇒ Android resource directory.

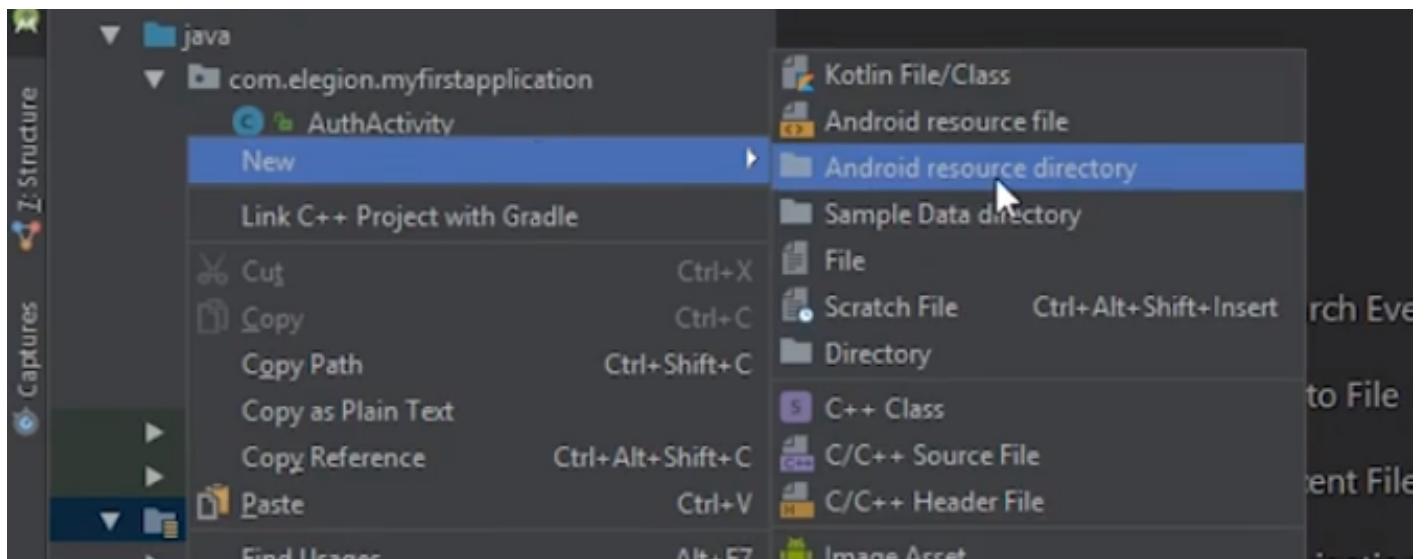


Рис. 4.4

В Resource type выбираем menu, нажимаем OK. У нас добавилась папочка.

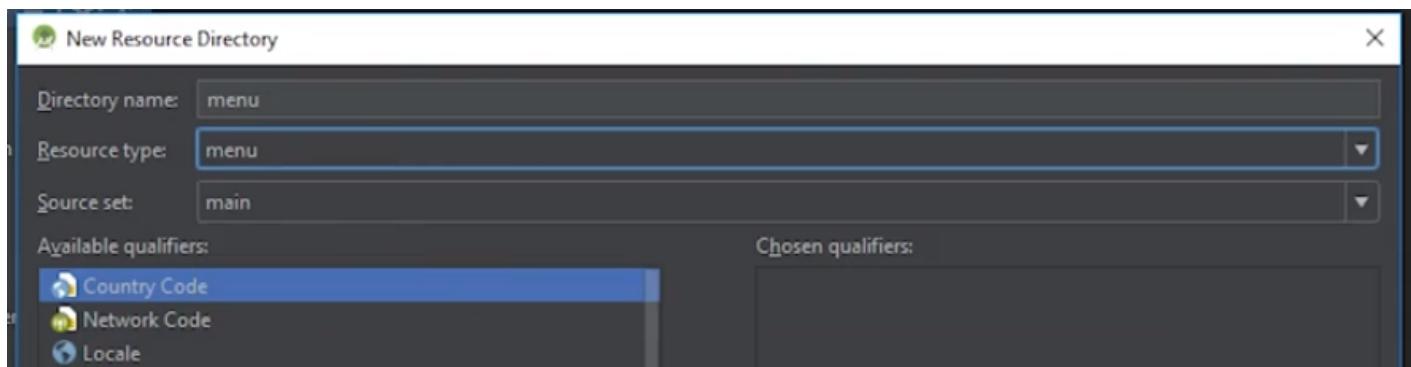


Рис. 4.5

Далее, правый клик menu⇒ New⇒ Menu resource file. Назовем меню profile_menu.

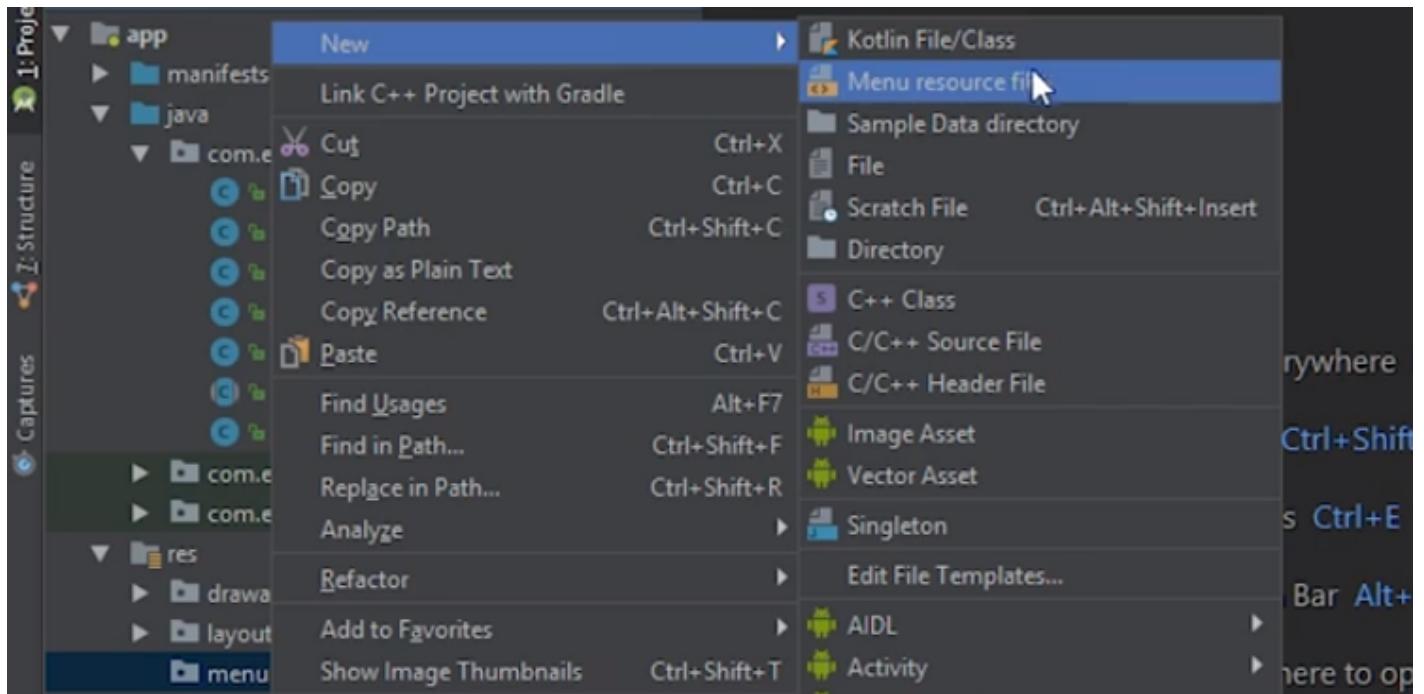


Рис. 4.6

Мы будем использовать это меню внутри активности профиля. Мы добавим функциональность логаута из приложения.

```

1  <?xml version="1.0" encoding="utf-8"?>
2  <menu xmlns:android="http://schemas.android.com/apk/res/android"
3      xmlns:app="http://schemas.android.com/apk/res-auto">
4      <item
5          android:id="@+id/actionLogout"
6          android:title="@string/logout"
7          app:showAsAction="never"/>
8  </menu>
```

Добавим строковый ресурс `string/logout="logout"`. Теперь добавим это меню в активность. Переходим в `ProfileActivity.java`.

Переопределяем методы `OnCreateOptionsMenu()` и `OnOptionsItemSelected()`:

```
1 public boolean onCreateOptionsMenu(Menu menu) {
2     MenuInflater inflater = getMenuInflater();
3     inflater.inflate(R.menu.profile_menu, menu);
4     return super.onCreateOptionsMenu(menu);
5 }
6 @Override
7 public boolean onOptionsItemSelected(MenuItem item) {
8     switch (item.getItemId()) {
9         //стандартный оператор Java
10        case R.id.actionLogout:
11            startActivity(new Intent(this, AuthActivity.class));
12            finish();
13            //запускаем Activity авторизации и закрываем эту
14            break;
15        default:break;
16    }
17    return super.onOptionsItemSelected(item);
18 }
```

Посмотрим, как это работает в эмуляторе. Вводим существующие данные и Нажимаем Войти. У нас добавилось меню. Нажимаем Логаут. У нас открылся экран без логина и пароля. Мы научились использовать меню внутри Activity.

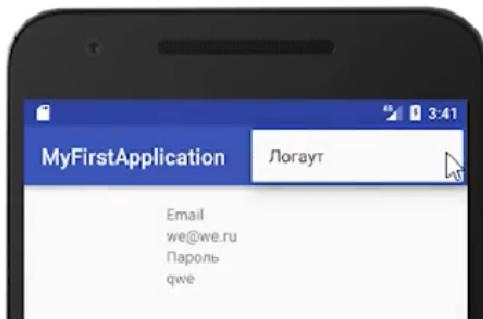


Рис. 4.7

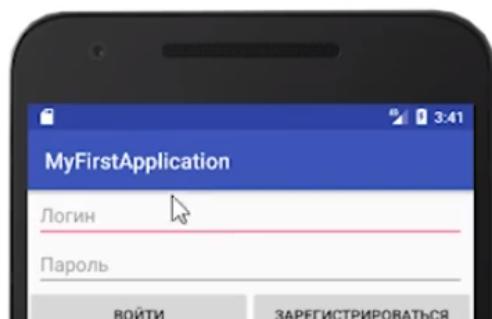


Рис. 4.8

4.1.3. КП. Обновлённая логика авторизации

Добавим в поле логина список успешно авторизованных пользователей. То есть мы будем нажимать на логин, и у нас под ним будет выпадать список пользователей, которые уже авторизованы. Как вы помните, в предыдущих занятиях мы добавили логику авторизации и мы эту логику

перенесли в SharedPreferencesHelper. Давайте посмотрим, как это выглядит теперь. Откроем проект, перейдем в AuthFragment. И посмотрим, как теперь выглядит mOnEnterClickListener:

```
1 private View.OnClickListener mOnEnterClickListener = new View.OnClickListener() {
2     @Override
3     public void onClick(View view) {
4         if ((isEmailValid()) && (isPasswordValid())) {
5             if (mSharedPreferencesHelper.login(new User(
6                 mLogin.getText().toString(),
7                 mPassword.getText().toString()))) {
8                 Intent startProfileIntent =
9                     new Intent(getActivity(), ProfileActivity.class);
10                startProfileIntent.putExtra(ProfileActivity.
11                    USER_KEY, new User(mLogin.getText().toString(),
12                        mPassword.getText().toString()));
13                startActivity(startProfileIntent);
14                getActivity().finish();
15            } else {
16                showMessage(R.string.login_error);
17            }
18        } else {
19            showMessage(R.string.input_error);
20        }
21        for (User user:mSharedPreferencesHelper.getUsers()) {
22            if (user.getLogin().equalsIgnoreCase(mLogin.
23                getText().toString())&& user.getPassword().equals(
24                    mPassword.getText().toString())) {
25                break;
26            }
27        }
28    }
29};
```

Здесь у нас появился метод login(). Если вас будет интересовать какая-то бизнес-логика, вы просто сможете посмотреть ее в проекте. mSharedPreferencesHelper.login(). Вся та же логика, что была до этого, просто она перенеслась в SharedPreferencesHelper. Чтобы показать выпадающий список, нам нужно: перейти в layout, fr_auth.xml, открыть preview(Text), заменить первый <EditText на <AutoCompleteTextView,

```
1 <!-- before -->
2 <EditText>
3 <!-- after -->
4 <AutoCompleteTextView>
```

перейти в AuthFragment, заменить EditText логина на AutoCompleteTextView.

```
1 private EditText mLogin;
2 //стало
3 private AutoCompleteTextView mLogin;
4 ...
5 private ArrayAdapter<String> mLoginedUsersAdapter;//дописали
6 ...//в OnCreateView проинициализируем:
7 mLoginedUsersAdapter = new ArrayAdapter<>(getActivity(),
8                 android.R.layout.simple_dropdown_item_1line,
9                 mSharedPreferencesHelper.getSuccessLogins());
10 //добавим адаптер в сам AutoCompleteTextView:
11 mLogin.setAdapter(mLoginedUsersAdapter);
```

Сначала передаем контекст (можно передать getActivity()), далее передаем ему ссылку на наш layout-файл. Мы будем использовать стандартные системные ресурсы Андроида. И передать ему сами данные для отображения. Чтобы это сделать, мы берем mSharedPreferencesHelper и вызываем у него метод get SuccessLogins().

Но пока мы не увидим выпадающего списка, если кого-то зарегистрируем, потому что мы с вами не добавили показ нашего выпадающего списка, когда у нас изменяется фокус. Создадим FocusChangeListener()

```
1 private View.OnFocusChangeListener mOnLoginFocusChangeListener = new
2     View.OnFocusChangeListener() {
3         @Override
4         public void onFocusChange(View view, boolean hasFocus) {
5             if (hasFocus) {//если фокус есть, покажем autocomplete
6                 mLogin.showDropDown();
7             }
8         }
9     };
```

boolean b переименуем в HasFocus, чтобы было понятнее. И если у нас есть фокус, то в на-

шем AutoCompleteTextView вызовем метод ShowDropDown(). В нашем AutoCompleteTextView мы забыли добавить OnFocusChangeListener().

```
1 //... исправляем
2 mEnter.setOnClickListener(mOnEnterClickListener);
3 mRegister.setOnClickListener(mOnRegisterClickListener);
4 //дописываем
5 mLogin.setOnFocusChangeListener(
6         mOnLoginFocusChangeListener);
```

Запустим эмулятор. Выпадающий список показался, значит всё правильно.

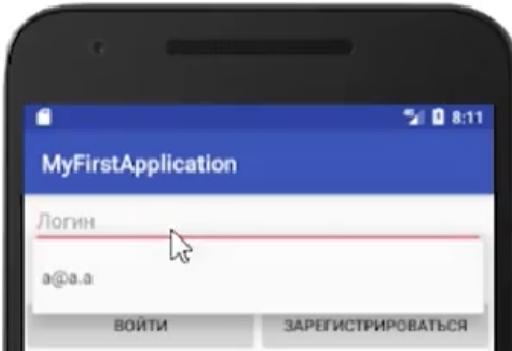


Рис. 4.9

4.1.4. КП. Экран профиля. Извлечение изображения из галереи

Научимся получать изображение из галереи. И изменим логику логина в приложение так, чтобы вместо boolean нам возвращался пользователь. Перейдем в SharedPreferencesHelper, сотрем входной параметр user, вместо этого добавим login и password. Входной параметр будет user. User get login заменяем на login и user get password заменяем на password. В качестве return возвращаем ему пользователя, которого он нашел в нашем shared preference хранилище.

```
1 public User login(String login, String password) {
2     List<User> users = getUsers();
3     for (User u : users) {
4         if (login.equalsIgnoreCase(u.getLogin()))
5             && password.equals(u.getPassword())) {
6             u.setHasSuccessLogin(true);
7             mSharedPreferences.edit().putString(USER_KEY,
8                 mGson.toJson(users, USERS_TYPE)).apply();
9             return u;
10        }
11    }
12    return null;
13    //если никого не нашёл
14 }
```

Теперь изменим использование этого метода. Стираем здесь new user и перед этим создадим его. Наш user будет равняться результату логина.

```
1 private View.OnClickListener mOnEnterClickListener = new View.OnClickListener() {
2     @Override
3     public void onClick(View view) {
4         if (isValidEmail() && isValidPassword()) {
5             User user = mSharedPreferencesHelper.login(
6                 mLogin.getText().toString(),
7                 mPassword.getText().toString());
8             if (user != null) {
9                 Intent startProfileIntent =
10                     new Intent(getActivity(),
11                         ProfileActivity.class);
12                     startProfileIntent.putExtra(ProfileActivity.
13                         USER_KEY, user);
14                     startActivity(startProfileIntent);
15                     getActivity().finish();
16             } else {
17                 showMessage(R.string.login_error);
18             }
19         } else {
20             showMessage(R.string.input_error);
21         }
22     }
23 }
```

Соответственно, если user != null, то можно переходить на следующий экран. И в качестве

параметра следующего экрана мы будем передавать user, которого получили из нашего хранилища с помощью метода login(). Иначе у нас будет отображаться ошибка логина. Чтобы получить изображение из галереи, перейдем в profile activity. В mOnPhotoClickListener() мы добавим метод:

```
1 private View.OnClickListener mOnPhotoClickListener = new
  → View.OnClickListener() {
2     @Override
3     public void onClick(View view) {
4         openGallery(); //при клике на фото открываем галерею.
5     }
6 };
7 private void openGallery() {
8     Intent intent = new Intent();
9     //создаём intent типа картинка
10    intent.setType("image/*");
11    intent.setAction(Intent.ACTION_GET_CONTENT);
12    //взяли фото
13    startActivityForResult(intent, REQUEST_CODE_GET_PHOTO);
14 }
```

И дописываем в начало ProfileActivity свой request_code

```
1 public static final int REQUEST_CODE_GET_PHOTO = 101;
```

нужно переопределить метод onActivityResult()

```
1     @Override
2     protected void onActivityResult(int requestCode,
3                                     int resultCode, Intent data) {
4     //если подаётся наш код запроса, результат OK и Data не null
5     if (requestCode == REQUEST_CODE_GET_PHOTO
6             && resultCode == Activity.RESULT_OK
7             && data != null) {
8         Uri photoUri = data.getData(); //ссылка на файл
9         mPhoto.setImageURI(photoUri); //установим в изображение
10    } else {
11        super.onActivityResult(requestCode, resultCode, data);
12    }
13 }
```

Запустим эмулятор. Выберем login, введем пароль, «войти», нажмем на наш image view, выберем какую-нибудь картинку, пусть это будет собачка. Как вы видите, картинка успешно установилась, то есть мы открыли галерею и выбрали картинку.

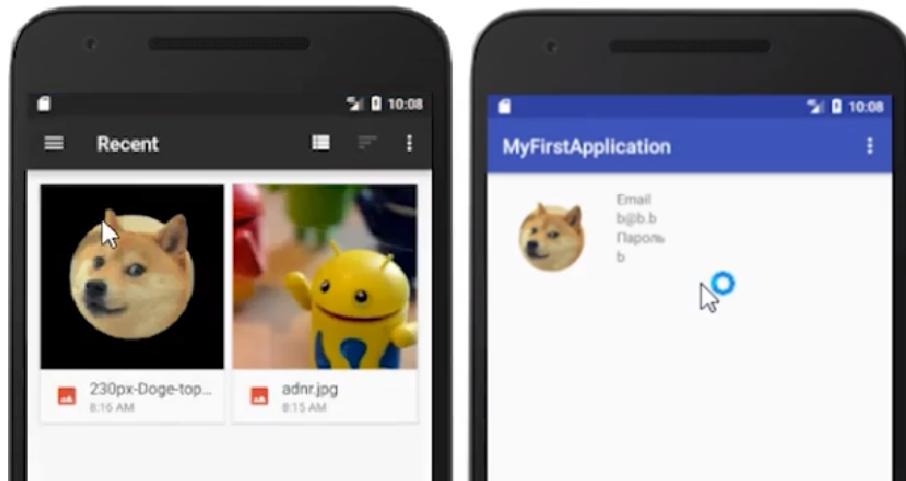


Рис. 4.10

В данном уроке мы научились получать thumbnail картинки из галереи.

4.1.5. КП. Градиентный фон

Рассмотрим, как можно создать градиент для фона, чтобы наше приложение выглядело лучше. Для этого перейдём в папочку res ⇒ drawable ⇒ New ⇒ Drawable resource file ⇒ File name: gradient background.

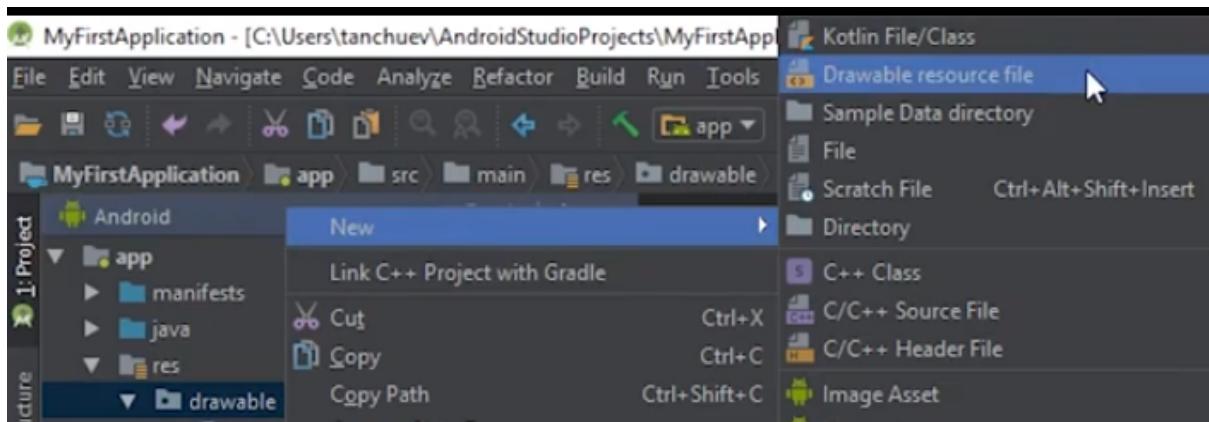


Рис. 4.11

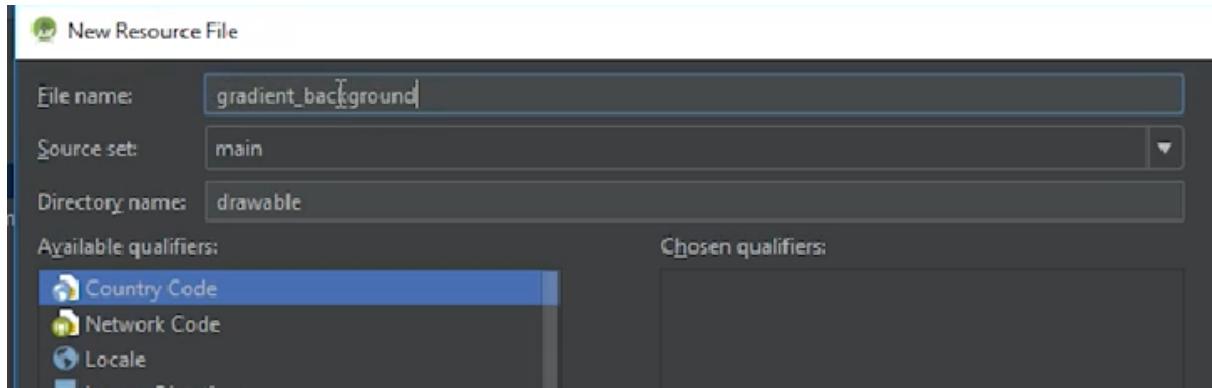


Рис. 4.12

Внутри добавляем item, внутри item — shape, внутри shape добавляем gradient. Зададим угол 90, startColor = PrimaryDark. и endColor=colorAccent. И затем ему type=linear, то есть он по умолчанию. И закрывающийся тег.

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <selector xmlns:android="http://schemas.android.com/apk/res/android">
3     <item>
4         <shape>
5             <gradient android:angle="90"
6                     android:endColor="@color/colorAccent"
7                     android:startColor="@color/colorPrimaryDark"
8                     android:type="linear"/>
9         </shape>
10    </item>
11 </selector>
```

Градиент создан. Перейдём в fr_auth.xml, в корневой LinearLayout добавим

```
1     android:background="@drawable/gradient_background"
```

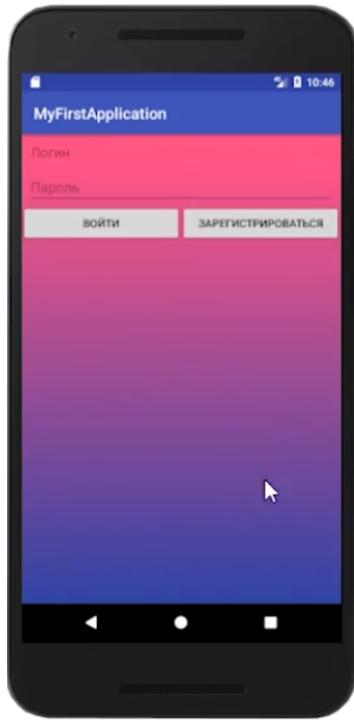


Рис. 4.13

Давайте запустим эмулятор и посмотрим, как это выглядит. Наш эмулятор запустился. И мы видим фон, который мы задали. В данном занятии мы научились создавать gradient background и устанавливать его как фон для экрана или какого-то layout.

Итак, мы с вами создали ваше первое Android-приложение. Оно содержит три экрана: экран авторизации, экран регистрации и экран показа профиля. Это приложение пригодится нам и в будущем; мы будем его изменять, улучшать и дорабатывать.

О проекте

Академия e-Legion – это образовательная платформа для повышения квалификации в мобильной разработке. Слушайте лекции топовых разработчиков, выполняйте практические задания и прокачивайте свои скиллы. Получите высокооплачиваемую профессию – разработчик мобильных приложений.

Программа “Android-разработчик”

Блок 1. Быстрый старт в Android-разработку

- Описание платформы Android
- Знакомство с IDE – Android Studio и системой сборки – Gradle
- Дебаг и логгирование
- Знакомство с основными сущностями Android-приложения
- Работа с Activity и Fragment
- Знакомство с элементами интерфейса – View, ViewGroup

Блок 2. Многопоточность и сетевое взаимодействие

- Работа со списками: RecyclerView
- Средства для обеспечения многопоточности в Android
- Работа с сетью с помощью Retrofit2/Okhttp3
- Базовое знакомство с реактивным программированием: RxJava2
- Работа с уведомлениями
- Работа с базами данных через Room

Блок 3. Архитектура Android-приложений

- MVP- и MVVM-паттерны
- Android Architecture Components
- Dependency Injection через Dagger2
- Clean Architecture

Блок 4. Тестирование и работа с картами

- Google Maps

- Оптимизация фоновых работ
- БД Realm
- WebView, ChromeCustomTabs
- Настройки приложений
- Picasso и Glide
- Unit- и UI-тестирование: Mockito, PowerMock, Espresso, Robolectric

Блок 5. Дизайн и анимации

- Стили и Темы
- Material Design Components
- Анимации
- Кастомные элементы интерфейса: Custom View

Блок 6. Облачные сервисы и периферия

- Google Firebase
- Google Analytics
- Push-уведомления
- Работа с сенсорами и камерой