Содержание

Неделя 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,

```
public class AuthFragment extends Fragment {
  private EditText mLogin;
  private EditText mPassword;
  private Button mEnter;
  private Button mRegister;
  //добавим mSharedPreferencesHelper
  private SharedPreferencesHelper
    mSharedPreferencesHelper;
//и ниже в onCreateView проинициализируем его и дадим ему Activity
  public View onCreateView(LayoutInflater inflater,
        @Nullable  ViewGroup container,
        @Nullable Bundle savedInstanceState) {
    View v = inflater.inflate(R.layout.fr_auth,
        container, false);
  mSharedPreferencesHelper = new
        SharedPreferencesHelper(getActivity());
```

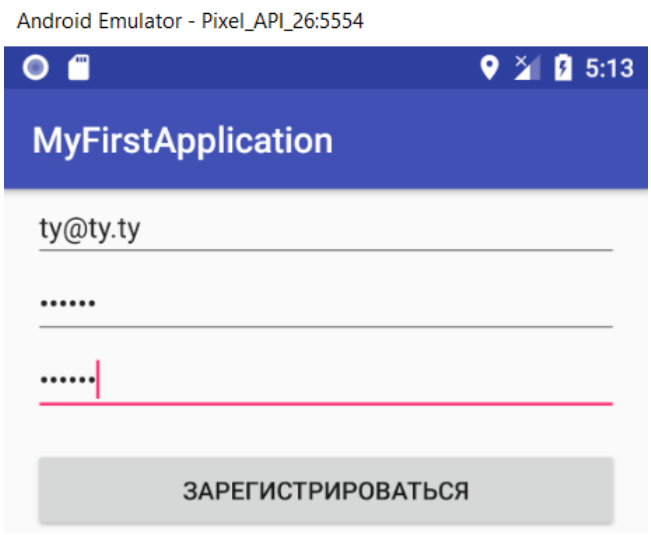
Дальше в mOnEnterClickListener(), то есть что происходит при нажатии на вход, добавляем логику проверки логина на наличие в SharedPrefereces и проверку пароля. После изменений OnClickListener() выглядит так:

```
@Override
public void onClick(View view) {
boolean isLoginSuccess = false;//проверка удачности входа
for (User user : mSharedPreferencesHelper.getUsers()) {
//дописываем проверку логина и пароля
  if (user.getLogin().equalsIgnoreCase(
      mLogin.getText().toString())&& user.getPassword().
        equals(mPassword.getText().toString())) {
    isLoginSuccess = true; //успешно вошли
    if (isEmailValid() && isPasswordValid()) {
      Intent startProfileIntent =
        new Intent(getActivity(), ProfileActivity.class);
      startProfileIntent.putExtra(ProfileActivity.
        USER_KEY
      ,new User(mLogin.getText().toString(), mPassword.
        getText().toString()));
      startActivity(startProfileIntent);
      getActivity().finish();
    } else {//если не удалось войти пишем ошибку входа
      showMessage(R.string.input_error);
    }
    break;//чтобы цикл прекратил свою работу
  }
}
```

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

```
<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.



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

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

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

```
private View.OnClickListener mOnRegisterClickListener
        = new View.OnClickListener() {
  @Override
  public void onClick(View view) {
    getFragmentManager().beginTransaction()
        .replace(R.id.fragmentContainer,
          RegistrationFragment.newInstance())
          .addToBackStack(RegistrationFragment
            .class.getName()).commit();
  }
};
```

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()

```
private View.OnClickListener mOnRegistrationClickListener
    = new View.OnClickListener() {
  @Override
  public void onClick(View view) {
    if (isInputValid()) {
      boolean isAdded = mSharedPreferencesHelper.
        addUser(new User(mLogin.getText().toString(),
            mPassword.getText().toString()));
      if (isAdded) {
        showMessage(R.string.login_register_success);
        getFragmentManager().popBackStack();//добавили
      }else{showMessage(R.string.login_register_error);}
    }
  }
};
```

Запустим эмулятор. Введём емейл, 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. 4.1: Неверный ввод

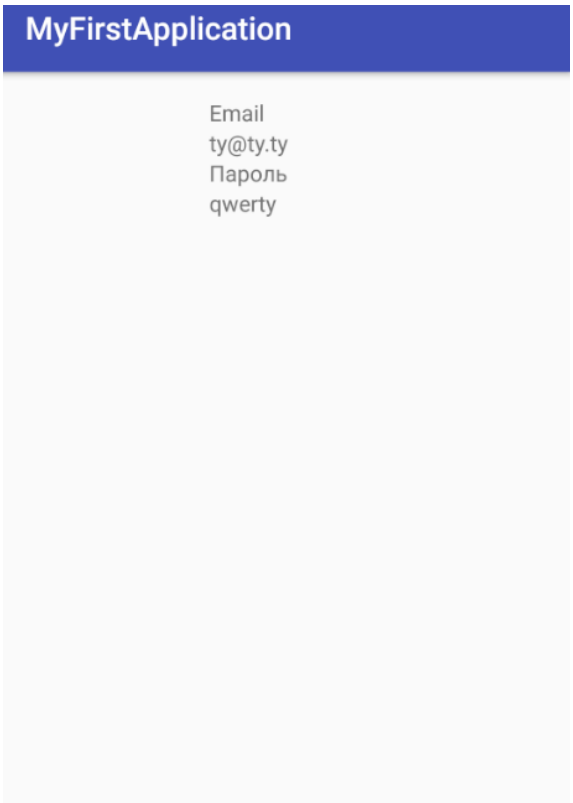


Рис. 4.2: Верный ввод

4.1.2 КП. Экран профиля. Логаут, меню

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

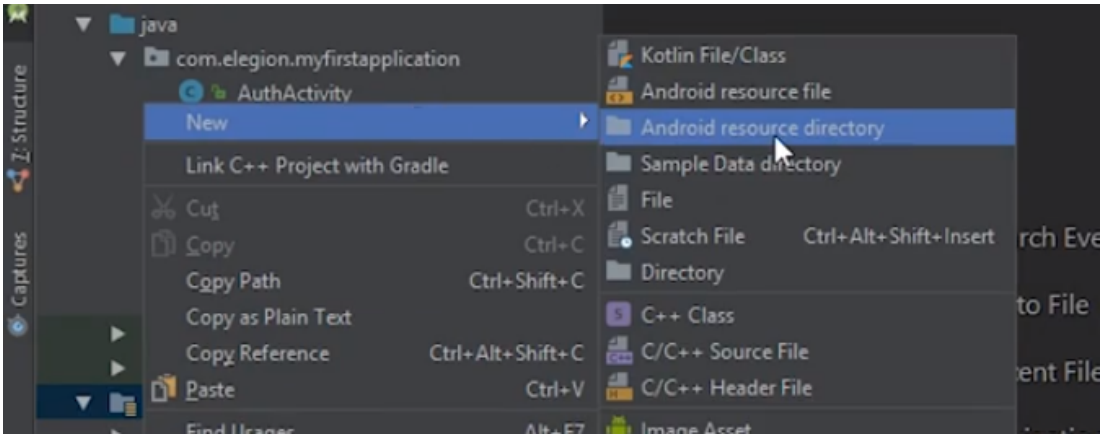


Рис. 4.3: Добавим Android resource directory

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

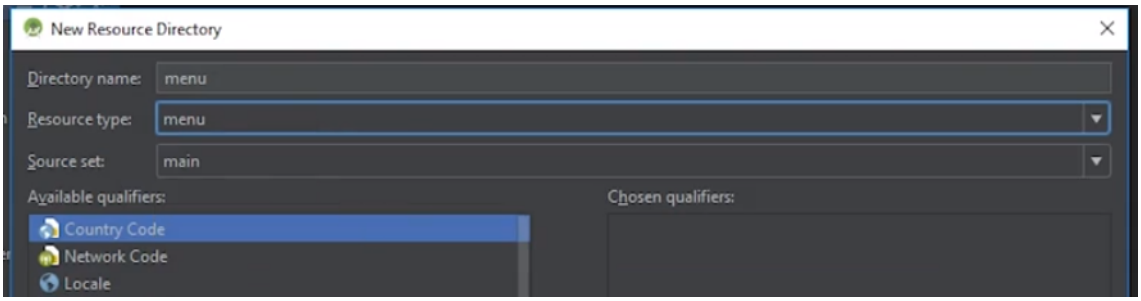


Рис. 4.4: Выбираем тип ресурса - меню

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

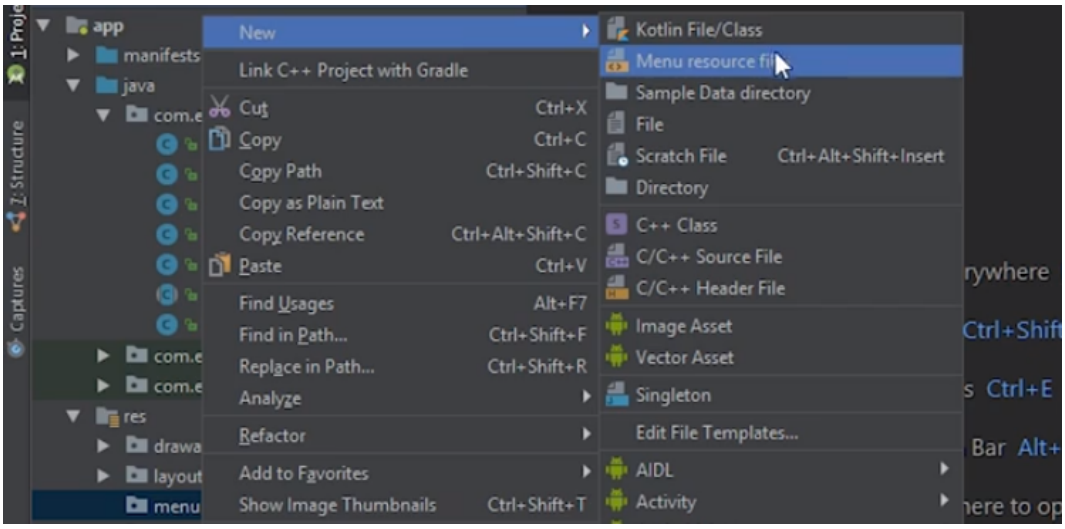


Рис. 4.5: В папке menu создаём ресурный файл разметки для меню

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

Листинг 4.1: profile_menu.xml

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

Добавим строковый ресурс string/logout="logout" . Теперь добавим это меню в активность. Переходим в ProfileActivity.java.
Переопределяем методы OnCreateOptionsMenu() и OnOptionsItemSelected():

```
public boolean onCreateOptionsMenu(Menu menu) {
  MenuInflater inflater = getMenuInflater();
  inflater.inflate(R.menu.profile_menu, menu);
  return super.onCreateOptionsMenu(menu);
}
@Override
public boolean onOptionsItemSelected(MenuItem item) {
  switch (item.getItemId()) {//стандартный оператор java
    case R.id.actionLogout:
      startActivity(new Intent(this,AuthActivity.class));
      finish();//запускаем активити авторизации и закрываем эту
      break;
    default:break;
  }
  return super.onOptionsItemSelected(item);
}
```

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.

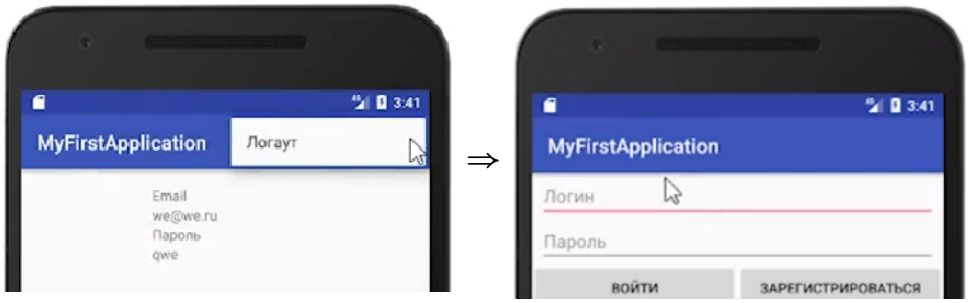


Рис. 4.6: Логаут

Рис. 4.7: Мы вышли

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:

```
private View.OnClickListener mOnEnterClickListener = new
  View.OnClickListener() {
  @Override
  public void onClick(View view) {
    if (isEmailValid() && isPasswordValid()) {
      if (mSharedPreferencesHelper.login(new User(
          mLogin.getText().toString(),
          mPassword.getText().toString()))) {
        Intent startProfileIntent =
        new Intent(getActivity(), ProfileActivity.class);
        startProfileIntent.putExtra(ProfileActivity.
        USER_KEY,new User(mLogin.getText().toString(),
        mPassword.getText().toString()));
        startActivity(startProfileIntent);
        getActivity().finish();
      } else {
        showMessage(R.string.login_error);
      }
    } else {
      showMessage(R.string.input_error);
    }
    for (User user:mSharedPreferencesHelper.getUsers()) {
      if (user.getLogin().equalsIgnoreCase(mLogin.
              getText().toString())&& user.getPassword().
                equals(mPassword.getText().toString()))
                {
        break;
      }
    }
  }
};
```

Здесь у нас появился метод login(). Если вас будет интересовать какая-то бизнес-логика, которую мы не разбирали при вас, вы просто сможете посмотреть ее в проекте. mSharedPreferencesHelper.login(). Вся та же логика, что была до этого,

просто она перенеслась в SharedPrefencesHelper.
Чтобы показать выпадающий список, нам нужно: перейти в layout, fr_auth.xml, открыть preview(Text), заменить первый <EditText на <AutoCompleteTextView,

```
...before
<EditText
...after
<AutoCompleteTextView
```

перейти в AuthFragment, заменить EditText логина на AutoCompleteTextView.

```
private EditText mLogin;
//стало
private AutoCompleteTextView mLogin;
...
private ArrayAdapter<String> mLoginedUsersAdapter;//
  дописали
...//в OnCreateView проинициализируем:
mLoginedUsersAdapter = new ArrayAdapter<>(getActivity(),
      android.R.layout.simple_dropdown_item_1line,
      mSharedPreferencesHelper.getSuccessLogins());
//добавим адаптер в сам AutoCompleteTextView:
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()

```
private View.OnFocusChangeListener
  mOnLoginFocusChangeListener = new View.
  OnFocusChangeListener() {
  @Override
  public void onFocusChange(View view, boolean hasFocus)
    {
    if (hasFocus) {//если фокус есть, покажем autocomplete
      mLogin.showDropDown();
    }
  }
};
```

boolean b переименуем в HasFocus, чтобы было понятнее. И если у нас есть фокус, то в нашем AutoCompleteTextView вызовем метод ShowDropDown(). Также

мы забыли добавить OnFocusChangeListener() в наш AutoCompleteTextView.

```
//... исправляем
mEnter.setOnClickListener(mOnEnterClickListener);
mRegister.setOnClickListener(mOnRegisterClickListener);
//дописываем
mLogin.setOnFocusChangeListener(
      mOnLoginFocusChangeListener);
```

Запустим эмулятор. Выпадающий список показался, значит всё правильно.

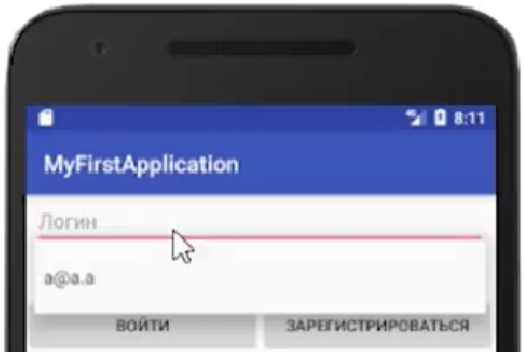


Рис. 4.8: Меню выбора логина

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.

```
public User login(String login, String password) {
  List<User> users = getUsers();
  for (User u : users) {
    if (login.equalsIgnoreCase(u.getLogin())
        && password.equals(u.getPassword())) {
      u.setHasSuccessLogin(true);
      mSharedPreferences.edit().putString(USERS_KEY,
              mGson.toJson(users, USERS_TYPE)).apply();
      return u;
    }
  }
  return null;//если никого не нашёл
}
```

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

```
private View.OnClickListener mOnEnterClickListener = new
  View.OnClickListener() {
  @Override
  public void onClick(View view) {
    if (isEmailValid() && isPasswordValid()) {
      User user = mSharedPreferencesHelper.login(
          mLogin.getText().toString(),
          mPassword.getText().toString());
      if (user != null) {
        Intent startProfileIntent =
           new Intent(getActivity(), ProfileActivity.
             class);
        startProfileIntent.putExtra(ProfileActivity.
            USER_KEY, user);
        startActivity(startProfileIntent);
        getActivity().finish();
      } else {
        showMessage(R.string.login_error);
      }
    } else {
      showMessage(R.string.input_error);
    }
```

Соответственно, если юзер != 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 mOnPhotoClickListner() мы добавим метод:

```
private View.OnClickListener mOnPhotoClickListener =  new
          View.OnClickListener() {
  @Override
  public void onClick(View view) {
    openGallery(); //при клике на фото открываем галерею.
  }
};
private void openGallery() {
  Intent intent = new Intent();//создаём интент типа картинка
  intent.setType("image/*");
  intent.setAction(Intent.ACTION_GET_CONTENT);//взяли фото
  startActivityForResult(intent, REQUEST_CODE_GET_PHOTO);
}
```

И дописываем в начало ProfileActivity свой request_code

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

нужно переопределить метод onActivityResult()

```
 @Override
protected void onActivityResult(int requestCode,
        int resultCode, Intent data) {
//если подаётся наш код запроса, результат OK и Data не null
  if (requestCode == REQUEST_CODE_GET_PHOTO
        && resultCode == Activity.RESULT_OK
        && data != null) {
    Uri photoUri = data.getData();//ссылка на файл
    mPhoto.setImageURI(photoUri);//установим в изображение
  } else {
    super.onActivityResult(requestCode,resultCode,data);
  }
}
```

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.

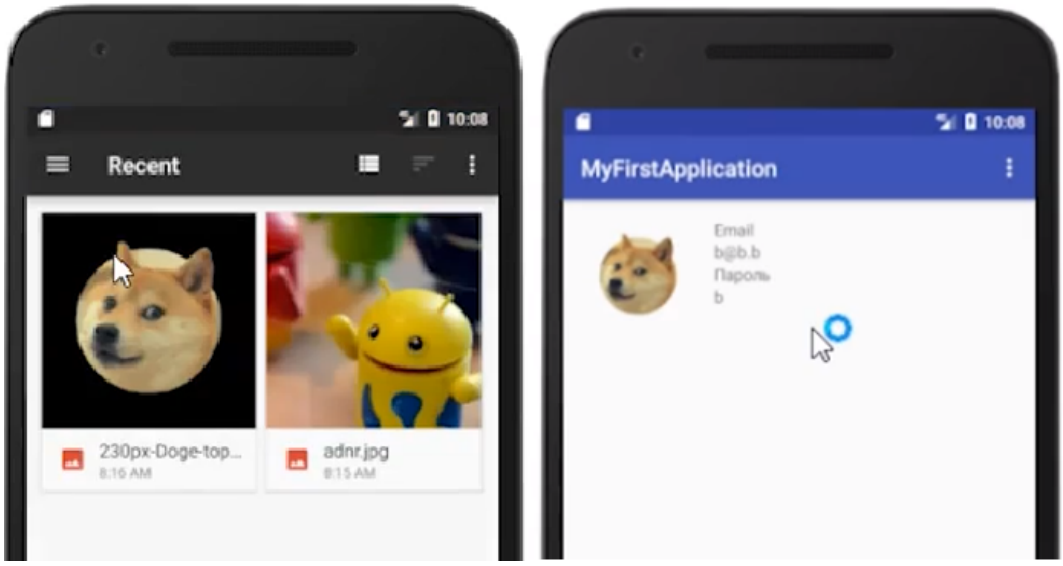


Рис. 4.9: Добавиление картинкуи профиля

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.

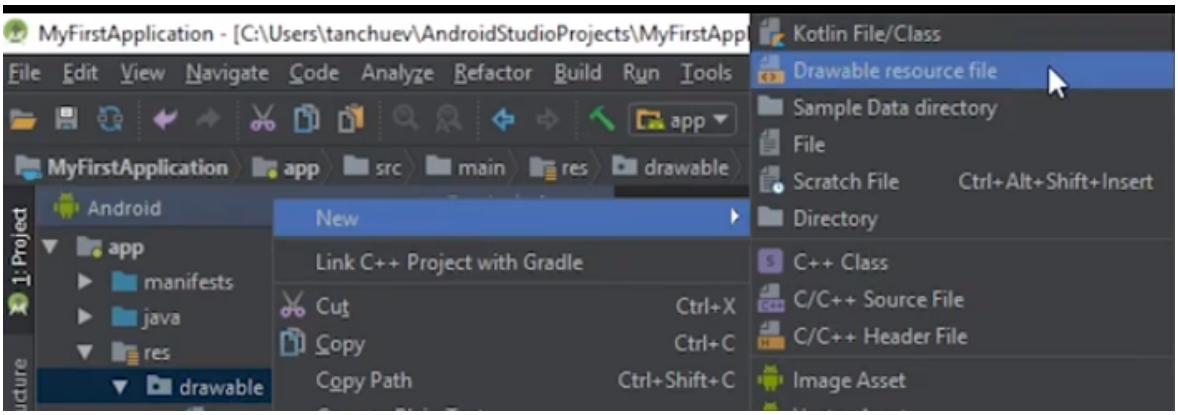


Рис. 4.10: Создаение Drawable resouce file

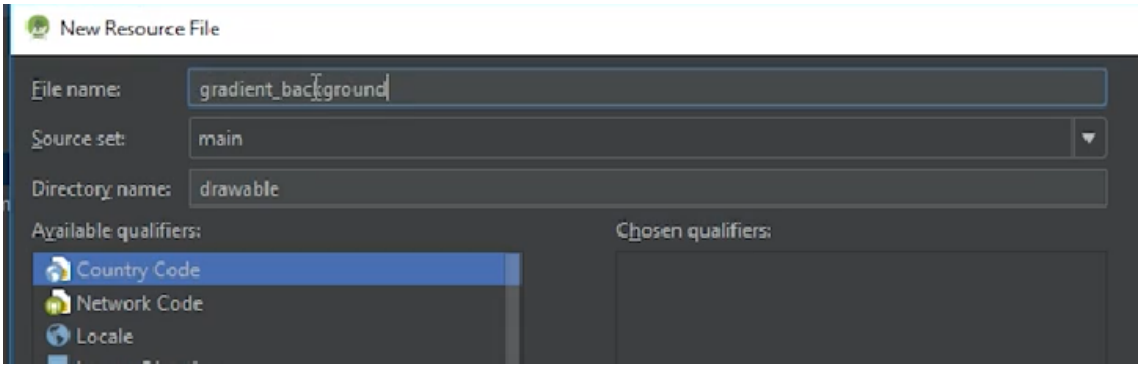


Рис. 4.11: Добавление градиентного фона

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.

Листинг 4.2: gradient_background.xml

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/
  res/android">
  <item>
  <shape>
    <gradient android:angle="90"
      android:endColor="@color/colorAccent"
      android:startColor="@color/colorPrimaryDark"
      android:type="linear"/>
  </shape>
</item>
</selector>
```

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

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

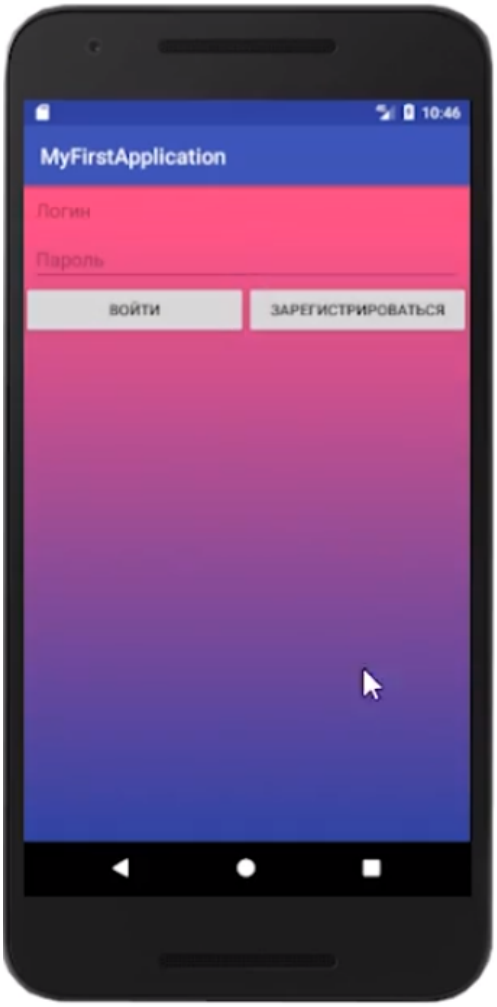


Рис. 4.12: Градиентный фон

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.