**ПРИЛОЖЕНИЕ Д. КОД ПРОГРАММЫ**

**АННОТАЦИЯ**

В данном разделе представлены описания модулей приложения, код программы со встроенными комментариями модулей, а также код запроса БД.

СОДЕРЖАНИЕ

[1. ВВЕДЕНИЕ 3](#_Toc159326143)

[1.1. Описание модулей 3](#_Toc159326144)

[1.2. AdminWindow 3](#_Toc159326145)

[1.3. AnalyticsPage 4](#_Toc159326146)

[1.4. AuthorizationWindow 5](#_Toc159326147)

[1.5. ClientsReqPage 6](#_Toc159326148)

[1.6. ClientWindow 9](#_Toc159326149)

[1.7. HistroryPage 10](#_Toc159326150)

[1.8. MainWindow 11](#_Toc159326151)

[1.9. ModerWindow 12](#_Toc159326152)

[1.10. RegistrationWindow 13](#_Toc159326153)

[1.11. Код запроса БД 14](#_Toc159326154)

# ВВЕДЕНИЕ

## Описание модулей

Модули программы и их описание в алфавитном порядке:

1. MainActivity – Окно авторизации
2. RegistrationActivity – Окно регистрации
3. UserMainActivity – Окно главного меню
4. BookListActivity – Окно списка книг
5. BookReadActivity – Окно чтения книги
6. AuthorListActivity – Окно списка авторов
7. GenreListActivity – Окно списка жанров
8. ProfileActivity – Окно профиля пользователя
9. AdminActivity – Окно меню администратора
10. AddBookActivity – Окно добавления книги

## MainActivity

package com.example.librarystar;

import androidx.appcompat.app.AppCompatActivity;

import androidx.appcompat.app.AppCompatDelegate;

import android.content.Intent;

import android.os.Bundle;

import android.view.View;

import android.widget.Button;

import android.widget.EditText;

import android.widget.TextView;

import android.widget.Toast;

import com.example.librarystar.Models.User;

import java.util.ArrayList;

import retrofit2.Call;

import retrofit2.Callback;

import retrofit2.Response;

public class MainActivity extends AppCompatActivity {

ApiInterface apiInterface;

Button reg, aut;

EditText log, pasw;

TextView restore\_pasw;

@Override

protected void onCreate(Bundle savedInstanceState) {

AppCompatDelegate.setDefaultNightMode(AppCompatDelegate.MODE\_NIGHT\_NO);

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

apiInterface = ServiceBuilder.buildRequest().create(ApiInterface.class);

reg = findViewById(R.id.registration\_button);

aut = findViewById(R.id.auth\_button);

log = findViewById(R.id.login\_field);

pasw = findViewById(R.id.password\_field);

restore\_pasw = findViewById(R.id.forgot\_password\_text\_button);

//restore\_pasw.setOnClickListener(new View.OnClickListener() {

// @Override

// public void onClick(View view) {

// Intent intent = new Intent(MainActivity.this, RestoreActivity.class);

// startActivity(intent);

// }

//});

reg.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

Intent intent = new Intent(MainActivity.this, RegistrationActivity.class);

startActivity(intent);

}

});

aut.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

User user = new User(log.getText().toString(), HashText.hashText(pasw.getText().toString()), "2006-04-27", null, null, null, 1, null, null);

Call<User> getUser = apiInterface.getUserByLogs(user);

getUser.enqueue(new Callback<User>() {

@Override

public void onResponse(Call<User> call, Response<User> response) {

if (response.isSuccessful()){

Intent intent = new Intent(MainActivity.this, UserMainActivity.class);

User user = response.body();

StaticData.user\_id = user.getId();

switch (user.getRole()){

case 1:

StaticData.user\_role = "Пользователь";

break;

case 2:

StaticData.user\_role = "Администратор";

break;

case 3:

StaticData.user\_role = "Издатель";

break;

}

startActivity(intent);

}

else {

Toast.makeText(MainActivity.this, "Неверный логин или пароль.", Toast.LENGTH\_SHORT).show();

log.setText("");

pasw.setText("");

}

}

@Override

public void onFailure(Call<User> call, Throwable t) {

}

});

}

});

}

}

## RegistrationActivity

package com.example.librarystar;

import android.content.Intent;

import android.os.Bundle;

import android.view.View;

import android.widget.Button;

import android.widget.EditText;

import android.widget.Toast;

import androidx.activity.EdgeToEdge;

import androidx.appcompat.app.AppCompatActivity;

import androidx.core.graphics.Insets;

import androidx.core.view.ViewCompat;

import androidx.core.view.WindowInsetsCompat;

import com.example.librarystar.Models.Role;

import com.example.librarystar.Models.User;

import com.google.android.material.snackbar.Snackbar;

import java.io.IOException;

import java.text.SimpleDateFormat;

import java.util.Calendar;

import okhttp3.MediaType;

import okhttp3.MultipartBody;

import okhttp3.OkHttpClient;

import okhttp3.Request;

import okhttp3.RequestBody;

import retrofit2.Call;

import retrofit2.Callback;

import retrofit2.Response;

public class RegistrationActivity extends AppCompatActivity {

EditText log, p1, p2;

Button auth, reg;

ApiInterface apiInterface;

Role role;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

EdgeToEdge.enable(this);

setContentView(R.layout.activity\_registration);

apiInterface = ServiceBuilder.buildRequest().create(ApiInterface.class);

ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {

Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());

v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);

return insets;

});

log = findViewById(R.id.reg\_login\_field);

p1 = findViewById(R.id.reg\_password\_field);

p2 = findViewById(R.id.reg\_password\_field\_2);

auth = findViewById(R.id.reg\_auth\_button);

reg = findViewById(R.id.reg\_registration\_button);

auth.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

Intent intent = new Intent(RegistrationActivity.this, MainActivity.class);

startActivity(intent);

}

});

reg.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

if (!p1.getText().toString().equals(p2.getText().toString())) {

Toast.makeText(RegistrationActivity.this, "Пароли не повторяются!", Toast.LENGTH\_SHORT).show();

return;

}

else if (log.getText().length() < 6) {

Toast.makeText(RegistrationActivity.this, "Логин должен быть длиной не менее 6 символов.", Toast.LENGTH\_SHORT).show();

return;

} else if (!StaticData.isPasswordValid(p1.getText().toString())) {

Snackbar.make(findViewById(android.R.id.content), "Пароль не соответствует требованиям.\n Требования для пароля:" +

"\n• Не короче 8 символов" +

"\n• Хотя бы один спецсимвол" +

"\n• Хотя бы одна цифра" +

"\n• Без пробелов" +

"\n• Только латинские буквы", Snackbar.LENGTH\_SHORT).show();

return;

}

Call<Role> getUser = apiInterface.getRoleUser();

getUser.enqueue(new Callback<Role>() {

@Override

public void onResponse(Call<Role> call, Response<Role> response) {

if (response.isSuccessful()) {

role = response.body();

addUser(role.getId());

} else {

Toast.makeText(RegistrationActivity.this, "Ошибка сервера.", Toast.LENGTH\_SHORT).show();

}

}

@Override

public void onFailure(Call<Role> call, Throwable t) {

Toast.makeText(RegistrationActivity.this, "Ошибка получения роли", Toast.LENGTH\_SHORT).show();

}

});

}

});

}

private void addUser(int role){

Calendar calendar = Calendar.getInstance();

SimpleDateFormat formatter = new SimpleDateFormat("yyyy-MM-dd");

String formattedDate = formatter.format(calendar.getTime());

User user = new User(log.getText().toString(), HashText.hashText(p1.getText().toString()),

formattedDate, null, null, null, role, null, null);

Call<User> addUser = apiInterface.addUser(user);

addUser.enqueue(new Callback<User>() {

@Override

public void onResponse(Call<User> call, Response<User> response) {

if (response.isSuccessful()){

Intent intent = new Intent(RegistrationActivity.this, MainActivity.class);

StaticData.user\_id = user.getId();

startActivity(intent);

}

else {

Toast.makeText(RegistrationActivity.this, "Ошибка данных или сервера", Toast.LENGTH\_SHORT).show();

}

}

@Override

public void onFailure(Call<User> call, Throwable t) {

Toast.makeText(RegistrationActivity.this, "Ошибка регистрации", Toast.LENGTH\_SHORT).show();

}

});

}

}

## UserMainActivity

package com.example.librarystar;

import android.content.Context;

import android.content.Intent;

import android.graphics.Color;

import android.os.Bundle;

import android.view.MenuItem;

import android.view.View;

import android.widget.Button;

import android.widget.ProgressBar;

import android.widget.Toast;

import androidx.activity.EdgeToEdge;

import androidx.annotation.NonNull;

import androidx.appcompat.app.AppCompatActivity;

import androidx.core.graphics.Insets;

import androidx.core.view.ViewCompat;

import androidx.core.view.WindowInsetsCompat;

import androidx.recyclerview.widget.LinearLayoutManager;

import androidx.recyclerview.widget.RecyclerView;

import com.example.librarystar.Models.Author;

import com.example.librarystar.Models.Book;

import com.example.librarystar.Models.BookWithAuthors;

import com.example.librarystar.Models.User;

import com.google.android.material.bottomnavigation.BottomNavigationView;

import java.util.ArrayList;

import java.util.List;

import retrofit2.Call;

import retrofit2.Callback;

import retrofit2.Response;

public class UserMainActivity extends AppCompatActivity {

BottomNavigationView bottomNavigationView;

Intent intent;

ArrayList<Book> books;

ArrayList<Author> authors;

ApiInterface apiInterface;

Button all\_books;

RecyclerView rv;

Context cont;

ProgressBar pb;

int i;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

EdgeToEdge.enable(this);

setContentView(R.layout.activity\_user\_main);

ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {

Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());

v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);

return insets;

});

apiInterface = ServiceBuilder.buildRequest().create(ApiInterface.class);

Intent thisintent = getIntent();

cont = this;

bottomNavigationView = findViewById(R.id.main\_bottom\_nav);

pb = findViewById(R.id.main\_progress\_bar);

bottomNavigationView.setBackgroundColor(Color.parseColor("#D29366"));

bottomNavigationView.setOnNavigationItemSelectedListener(new BottomNavigationView.OnNavigationItemSelectedListener() {

@Override

public boolean onNavigationItemSelected(@NonNull MenuItem menuItem) {

switch (menuItem.getItemId()){

case R.id.first\_item:

intent = new Intent(UserMainActivity.this, UserMainActivity.class);

startActivity(intent);

return true;

case R.id.second\_item:

intent = new Intent(UserMainActivity.this, GenreListActivity.class);

startActivity(intent);

return true;

case R.id.third\_item:

intent = new Intent(UserMainActivity.this, AuthorListActivity.class);

startActivity(intent);

return true;

case R.id.fourth\_item:

intent = new Intent(UserMainActivity.this, ProfileActivity.class);

startActivity(intent);

return true;

}

return false;

}

});

rv = findViewById(R.id.user\_horizontal\_rv);

all\_books = findViewById(R.id.all\_books\_button);

all\_books.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

Intent intent1 = new Intent(UserMainActivity.this, BookListActivity.class);

startActivity(intent1);

}

});

books = new ArrayList<>();

authors = new ArrayList<>();

Call<ArrayList<Book>> getUser = apiInterface.getBooksByUserId(StaticData.user\_id);

getUser.enqueue(new Callback<ArrayList<Book>>() {

@Override

public void onResponse(Call<ArrayList<Book>> call, Response<ArrayList<Book>> response) {

if (response.isSuccessful()){

books = response.body();

set\_rv();

}

else {

Toast.makeText(cont, "Не найдено любимых книг", Toast.LENGTH\_SHORT).show();

}

}

@Override

public void onFailure(Call<ArrayList<Book>> call, Throwable t) {

}

});

}

void set\_rv(){

if (books.isEmpty()){

pb.setVisibility(ProgressBar.INVISIBLE);

return;

}

i = 0;

ArrayList<BookWithAuthors> balist = new ArrayList<BookWithAuthors>();

authors = new ArrayList<>();

for (Book book : books){

authors.clear();

Call<ArrayList<Author>> getAuthorsBook = apiInterface.getAuthorsByBookId(book.getId());

getAuthorsBook.enqueue(new Callback<ArrayList<Author>>() {

@Override

public void onResponse(Call<ArrayList<Author>> call, Response<ArrayList<Author>> response) {

if (response.isSuccessful()){

authors = response.body();

balist.add(new BookWithAuthors(book, authors){});

if (i+1==books.size()){

rv.setLayoutManager(new LinearLayoutManager(cont, LinearLayoutManager.HORIZONTAL, false));

rv.setHasFixedSize(true);

rv.setAdapter(new VerticalBookRecyclerAdapter(balist, cont));

pb.setVisibility(ProgressBar.INVISIBLE);

}

i++;

}

else {

Toast.makeText(cont, "Ошибка сервера.", Toast.LENGTH\_SHORT).show();

}

}

@Override

public void onFailure(Call<ArrayList<Author>> call, Throwable t) {

}

});

}

}

}

## BookListActivity

package com.example.librarystar;

import android.content.Context;

import android.content.Intent;

import android.os.Bundle;

import android.view.View;

import android.widget.Button;

import android.widget.ProgressBar;

import android.widget.Toast;

import androidx.activity.EdgeToEdge;

import androidx.appcompat.app.AppCompatActivity;

import androidx.core.graphics.Insets;

import androidx.core.view.ViewCompat;

import androidx.core.view.WindowInsetsCompat;

import androidx.recyclerview.widget.LinearLayoutManager;

import androidx.recyclerview.widget.RecyclerView;

import com.example.librarystar.Models.Author;

import com.example.librarystar.Models.Book;

import com.example.librarystar.Models.BookWithAuthors;

import java.util.ArrayList;

import retrofit2.Call;

import retrofit2.Callback;

import retrofit2.Response;

public class BookListActivity extends AppCompatActivity {

ApiInterface apiInterface;

RecyclerView rv;

Button but;

ArrayList<Book> books;

ArrayList<Author> authors;

int i;

ProgressBar pb;

Context cont;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

EdgeToEdge.enable(this);

setContentView(R.layout.activity\_book\_list);

ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {

Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());

v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);

return insets;

});

apiInterface = ServiceBuilder.buildRequest().create(ApiInterface.class);

rv = findViewById(R.id.book\_horizontal\_rv);

but = findViewById(R.id.book\_list\_get\_back);

pb = findViewById(R.id.book\_list\_pb);

books = new ArrayList<>();

cont = this;

authors = new ArrayList<>();

but.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

Intent intent = new Intent(BookListActivity.this, UserMainActivity.class);

startActivity(intent);

}

});

Call<ArrayList<Book>> getBooks = apiInterface.getBooks();

getBooks.enqueue(new Callback<ArrayList<Book>>() {

@Override

public void onResponse(Call<ArrayList<Book>> call, Response<ArrayList<Book>> response) {

if (response.isSuccessful()){

books = response.body();

set\_rv();

}

else {

Toast.makeText(cont, "Не найдено книг", Toast.LENGTH\_SHORT).show();

}

}

@Override

public void onFailure(Call<ArrayList<Book>> call, Throwable t) {

Toast.makeText(cont, "Ошибка запроса", Toast.LENGTH\_SHORT).show();

}

});

}

void set\_rv(){

i = 0;

ArrayList<BookWithAuthors> balist = new ArrayList<BookWithAuthors>();

authors = new ArrayList<>();

for (Book book : books){

authors.clear();

Call<ArrayList<Author>> getAuthorsBook = apiInterface.getAuthorsByBookId(book.getId());

getAuthorsBook.enqueue(new Callback<ArrayList<Author>>() {

@Override

public void onResponse(Call<ArrayList<Author>> call, Response<ArrayList<Author>> response) {

if (response.isSuccessful()){

authors = response.body();

balist.add(new BookWithAuthors(book, authors){});

if (i+1==books.size()){

rv.setLayoutManager(new LinearLayoutManager(cont, LinearLayoutManager.VERTICAL, false));

rv.setHasFixedSize(true);

rv.setAdapter(new HoryzontalBookRecyclerAdapter(balist, cont));

pb.setVisibility(ProgressBar.INVISIBLE);

}

i++;

}

else {

Toast.makeText(cont, "Ошибка сервера.", Toast.LENGTH\_SHORT).show();

}

}

@Override

public void onFailure(Call<ArrayList<Author>> call, Throwable t) {

Toast.makeText(cont, "Ошибка запроса.", Toast.LENGTH\_SHORT).show();

}

});

}

}

}

## BookReadActivity

package com.example.librarystar;

import android.content.Context;

import android.content.Intent;

import android.content.SharedPreferences;

import android.graphics.Bitmap;

import android.graphics.pdf.PdfRenderer;

import android.os.Bundle;

import android.os.ParcelFileDescriptor;

import android.util.Base64;

import android.view.LayoutInflater;

import android.view.View;

import android.view.ViewGroup;

import android.widget.ImageView;

import android.widget.TextView;

import android.widget.Toast;

import androidx.annotation.NonNull;

import androidx.appcompat.app.AppCompatActivity;

import androidx.viewpager.widget.PagerAdapter;

import androidx.viewpager.widget.ViewPager;

import com.example.librarystar.Models.Book;

import com.example.librarystar.Models.BookWithAuthors;

import java.io.ByteArrayInputStream;

import java.io.File;

import java.io.FileOutputStream;

import java.io.IOException;

import retrofit2.Call;

import retrofit2.Callback;

import retrofit2.Response;

public class BookReadActivity extends AppCompatActivity {

private ViewPager viewPager;

private TextView pageNumber;

private TextView pageCount;

private int currentPage = 0;

private PdfRenderer pdfRenderer;

private PdfRenderer.Page currentPageObj;

private ParcelFileDescriptor parcelFileDescriptor;

private SharedPreferences sharedPreferences;

private ApiInterface apiInterface;

private int id;

private Context cont;

private Book book;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_book\_read);

viewPager = findViewById(R.id.viewPager);

pageNumber = findViewById(R.id.pageNumber);

pageCount = findViewById(R.id.pageCount);

Intent intent = getIntent();

id = intent.getIntExtra("id", 0);

cont = this;

sharedPreferences = getSharedPreferences("PDFPrefs", MODE\_PRIVATE);

currentPage = sharedPreferences.getInt("currentPage", 0);

apiInterface = ServiceBuilder.buildRequest().create(ApiInterface.class);

Call<Book> getAuthorsBook = apiInterface.getBook(id);

getAuthorsBook.enqueue(new Callback<Book>() {

@Override

public void onResponse(Call<Book> call, Response<Book> response) {

if (response.isSuccessful()){

book = response.body();

PdfBuild();

}

else {

Toast.makeText(cont, "Ошибка сервера.", Toast.LENGTH\_SHORT).show();

}

}

@Override

public void onFailure(Call<Book> call, Throwable t) {

Toast.makeText(cont, "Ошибка запроса.", Toast.LENGTH\_SHORT).show();

}

});

}

private void PdfBuild() {

String base64Pdf = book.getPdf\_path();

byte[] pdfAsBytes = Base64.decode(base64Pdf, Base64.DEFAULT);

try {

File tempFile = File.createTempFile("tempPdf", ".pdf", getCacheDir());

FileOutputStream fos = new FileOutputStream(tempFile);

fos.write(pdfAsBytes);

fos.close();

parcelFileDescriptor = ParcelFileDescriptor.open(tempFile, ParcelFileDescriptor.MODE\_READ\_ONLY);

pdfRenderer = new PdfRenderer(parcelFileDescriptor);

int pageCount = pdfRenderer.getPageCount();

this.pageCount.setText(" / " + pageCount);

PdfPagerAdapter pdfPagerAdapter = new PdfPagerAdapter();

viewPager.setAdapter(pdfPagerAdapter);

viewPager.setCurrentItem(currentPage);

showPage(currentPage);

// Добавьте слушатель изменения страницы

viewPager.addOnPageChangeListener(new ViewPager.OnPageChangeListener() {

@Override

public void onPageScrolled(int position, float positionOffset, int positionOffsetPixels) {

}

@Override

public void onPageSelected(int position) {

showPage(position);

}

@Override

public void onPageScrollStateChanged(int state) {

}

});

} catch (IOException e) {

Toast.makeText(this, "Error loading PDF", Toast.LENGTH\_SHORT).show();

e.printStackTrace();

}

}

private class PdfPagerAdapter extends PagerAdapter {

@Override

public Object instantiateItem(@NonNull ViewGroup container, int position) {

ImageView imageView = new ImageView(BookReadActivity.this);

imageView.setLayoutParams(new ViewGroup.LayoutParams(ViewGroup.LayoutParams.MATCH\_PARENT, ViewGroup.LayoutParams.MATCH\_PARENT));

imageView.setTag("pdf\_page\_" + position); // Устанавливаем тег для поиска

container.addView(imageView);

showPage(currentPage);

return imageView;

}

@Override

public int getCount() {

return pdfRenderer.getPageCount();

}

@Override

public boolean isViewFromObject(@NonNull View view, @NonNull Object object) {

return view == object;

}

@Override

public void destroyItem(@NonNull ViewGroup container, int position, @NonNull Object object) {

container.removeView((View) object);

}

}

private void showPage(int index) {

if (pdfRenderer.getPageCount() <= index) {

return;

}

// Закрываем текущую страницу перед открытием новой

if (currentPageObj != null) {

currentPageObj.close();

}

currentPageObj = pdfRenderer.openPage(index);

Bitmap bitmap = Bitmap.createBitmap(currentPageObj.getWidth(), currentPageObj.getHeight(), Bitmap.Config.ARGB\_8888);

currentPageObj.render(bitmap, null, null, PdfRenderer.Page.RENDER\_MODE\_FOR\_DISPLAY);

ImageView imageView = viewPager.findViewWithTag("pdf\_page\_" + index);

if (imageView != null) {

imageView.setImageBitmap(bitmap);

}

pageNumber.setText("Страница " + (index + 1));

currentPage = index;

}

@Override

protected void onPause() {

super.onPause();

SharedPreferences.Editor editor = sharedPreferences.edit();

editor.putInt("currentPage", currentPage);

editor.apply();

}

@Override

protected void onDestroy() {

try {

if (currentPageObj != null) {

currentPageObj.close();

}

if (pdfRenderer != null) {

pdfRenderer.close();

}

if (parcelFileDescriptor != null) {

parcelFileDescriptor.close();

}

} catch (IOException e) {

e.printStackTrace();

}

super.onDestroy();

}

}

## AuthorListActivity

package com.example.librarystar;

import android.content.Context;

import android.content.Intent;

import android.graphics.Color;

import android.os.Bundle;

import android.view.MenuItem;

import android.widget.ProgressBar;

import android.widget.Toast;

import androidx.annotation.NonNull;

import androidx.appcompat.app.AppCompatActivity;

import com.example.librarystar.Models.BookWithAuthors;

import retrofit2.Call;

import retrofit2.Callback;

import retrofit2.Response;

import androidx.activity.EdgeToEdge;

import androidx.annotation.NonNull;

import androidx.appcompat.app.AppCompatActivity;

import androidx.core.graphics.Insets;

import androidx.core.view.ViewCompat;

import androidx.core.view.WindowInsetsCompat;

import androidx.recyclerview.widget.LinearLayoutManager;

import androidx.recyclerview.widget.RecyclerView;

import com.example.librarystar.Models.Author;

import com.example.librarystar.Models.BookWithAuthors;

import com.google.android.material.bottomnavigation.BottomNavigationView;

import java.util.ArrayList;

import retrofit2.Call;

import retrofit2.Callback;

public class AuthorListActivity extends AppCompatActivity {

Intent intent;

ApiInterface apiInterface;

ArrayList<Author> authors;

Context cont;

RecyclerView rv;

ProgressBar pb;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

EdgeToEdge.enable(this);

setContentView(R.layout.activity\_author\_list);

ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {

Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());

v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);

return insets;

});

cont = this;

rv = findViewById(R.id.author\_vertical\_rv);

pb = findViewById(R.id.author\_pb);

apiInterface = ServiceBuilder.buildRequest().create(ApiInterface.class);

BottomNavigationView bottomNavigationView = findViewById(R.id.author\_bottom\_nav);

bottomNavigationView.setSelectedItemId(R.id.third\_item);

bottomNavigationView.setBackgroundColor(Color.parseColor("#D29366"));

bottomNavigationView.setOnNavigationItemSelectedListener(new BottomNavigationView.OnNavigationItemSelectedListener() {

@Override

public boolean onNavigationItemSelected(@NonNull MenuItem menuItem) {

switch (menuItem.getItemId()){

case R.id.first\_item:

intent = new Intent(AuthorListActivity.this, UserMainActivity.class);

startActivity(intent);

return true;

case R.id.second\_item:

intent = new Intent(AuthorListActivity.this, GenreListActivity.class);

startActivity(intent);

return true;

case R.id.third\_item:

intent = new Intent(AuthorListActivity.this, AuthorListActivity.class);

startActivity(intent);

return true;

case R.id.fourth\_item:

intent = new Intent(AuthorListActivity.this, ProfileActivity.class);

startActivity(intent);

return true;

}

return false;

}

});

Call<ArrayList<Author>> getAuthorsBook = apiInterface.getAuthorsList();

getAuthorsBook.enqueue(new Callback<ArrayList<Author>>() {

@Override

public void onResponse(Call<ArrayList<Author>> call, Response<ArrayList<Author>> response) {

if (response.isSuccessful()){

authors = response.body();

rv.setLayoutManager(new LinearLayoutManager(cont, LinearLayoutManager.HORIZONTAL, false));

rv.setHasFixedSize(true);

rv.setAdapter(new HorizontalAuthorRecyclerAdapter(authors, cont));

pb.setVisibility(ProgressBar.INVISIBLE);

}

else {

Toast.makeText(cont, "Ошибка сервера.", Toast.LENGTH\_SHORT).show();

}

}

@Override

public void onFailure(Call<ArrayList<Author>> call, Throwable t) {

}

});

}

}

## GenreListActivity

package com.example.librarystar;

import android.content.Context;

import android.content.Intent;

import android.graphics.Color;

import android.os.Bundle;

import android.view.MenuItem;

import android.widget.ProgressBar;

import android.widget.Toast;

import androidx.activity.EdgeToEdge;

import androidx.annotation.NonNull;

import androidx.appcompat.app.AppCompatActivity;

import androidx.core.graphics.Insets;

import androidx.core.view.ViewCompat;

import androidx.core.view.WindowInsetsCompat;

import androidx.recyclerview.widget.GridLayoutManager;

import androidx.recyclerview.widget.LinearLayoutManager;

import androidx.recyclerview.widget.RecyclerView;

import com.example.librarystar.Models.Author;

import com.example.librarystar.Models.BookWithAuthors;

import com.example.librarystar.Models.Genre;

import com.google.android.material.bottomnavigation.BottomNavigationView;

import java.util.ArrayList;

import retrofit2.Call;

import retrofit2.Callback;

import retrofit2.Response;

public class GenreListActivity extends AppCompatActivity {

Intent intent;

ApiInterface apiInterface;

Context cont;

ArrayList<Genre> genres;

RecyclerView rv;

ProgressBar pb;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

EdgeToEdge.enable(this);

setContentView(R.layout.activity\_genre\_list);

ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {

Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());

v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);

return insets;

});

BottomNavigationView bottomNavigationView = findViewById(R.id.genre\_bottom\_nav);

bottomNavigationView.setSelectedItemId(R.id.second\_item);

apiInterface = ServiceBuilder.buildRequest().create(ApiInterface.class);

pb = findViewById(R.id.genre\_pb);

rv = findViewById(R.id.genre\_grid\_rv);

bottomNavigationView.setBackgroundColor(Color.parseColor("#D29366"));

bottomNavigationView.setOnNavigationItemSelectedListener(new BottomNavigationView.OnNavigationItemSelectedListener() {

@Override

public boolean onNavigationItemSelected(@NonNull MenuItem menuItem) {

switch (menuItem.getItemId()){

case R.id.first\_item:

intent = new Intent(GenreListActivity.this, UserMainActivity.class);

startActivity(intent);

return true;

case R.id.second\_item:

intent = new Intent(GenreListActivity.this, GenreListActivity.class);

startActivity(intent);

return true;

case R.id.third\_item:

intent = new Intent(GenreListActivity.this, AuthorListActivity.class);

startActivity(intent);

return true;

case R.id.fourth\_item:

intent = new Intent(GenreListActivity.this, ProfileActivity.class);

startActivity(intent);

return true;

}

return false;

}

});

Call<ArrayList<Genre>> getAuthorsBook = apiInterface.getGenresList();

getAuthorsBook.enqueue(new Callback<ArrayList<Genre>>() {

@Override

public void onResponse(Call<ArrayList<Genre>> call, Response<ArrayList<Genre>> response) {

if (response.isSuccessful()){

genres = response.body();

rv.setLayoutManager(new GridLayoutManager(cont, 2));

rv.setHasFixedSize(true);

rv.setAdapter(new GridGenreRecyclerAdapter(genres, cont));

pb.setVisibility(ProgressBar.INVISIBLE);

}

else {

Toast.makeText(cont, "Ошибка сервера.", Toast.LENGTH\_SHORT).show();

}

}

@Override

public void onFailure(Call<ArrayList<Genre>> call, Throwable t) {

}

});

}

}

## ProfileActivity

package com.example.librarystar;

import android.content.Context;

import android.content.Intent;

import android.graphics.Color;

import android.os.Bundle;

import android.view.MenuItem;

import android.view.View;

import android.widget.Button;

import android.widget.ImageView;

import android.widget.TextView;

import android.widget.Toast;

import androidx.activity.EdgeToEdge;

import androidx.annotation.NonNull;

import androidx.appcompat.app.AppCompatActivity;

import androidx.core.graphics.Insets;

import androidx.core.view.ViewCompat;

import androidx.core.view.WindowInsetsCompat;

import com.example.librarystar.Models.User;

import retrofit2.Call;

import retrofit2.Callback;

import retrofit2.Response;

import com.google.android.material.bottomnavigation.BottomNavigationView;

import com.squareup.picasso.Picasso;

import java.util.Objects;

public class ProfileActivity extends AppCompatActivity {

ImageView img;

TextView tv, rd;

Button exit, change, admin;

ApiInterface apiInterface;

Intent thisintent;

Intent intent;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

EdgeToEdge.enable(this);

setContentView(R.layout.activity\_profile);

ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {

Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());

v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);

return insets;

});

thisintent = getIntent();

BottomNavigationView bottomNavigationView = findViewById(R.id.main\_bottom\_nav);

bottomNavigationView.setSelectedItemId(R.id.fourth\_item);

bottomNavigationView.setBackgroundColor(Color.parseColor("#D29366"));

bottomNavigationView.setOnNavigationItemSelectedListener(new BottomNavigationView.OnNavigationItemSelectedListener() {

@Override

public boolean onNavigationItemSelected(@NonNull MenuItem menuItem) {

switch (menuItem.getItemId()){

case R.id.first\_item:

intent = new Intent(ProfileActivity.this, UserMainActivity.class);

startActivity(intent);

return true;

case R.id.second\_item:

intent = new Intent(ProfileActivity.this, GenreListActivity.class);

startActivity(intent);

return true;

case R.id.third\_item:

intent = new Intent(ProfileActivity.this, AuthorListActivity.class);

startActivity(intent);

return true;

case R.id.fourth\_item:

intent = new Intent(ProfileActivity.this, ProfileActivity.class);

startActivity(intent);

return true;

}

return false;

}

});

tv = findViewById(R.id.profileText);

rd = findViewById(R.id.registrationDateText);

img = findViewById(R.id.profileImg);

exit = findViewById(R.id.exitFromAccountButton);

change = findViewById(R.id.changePasswordButton);

admin = findViewById(R.id.adminMenuButton);

if (!Objects.equals(StaticData.user\_role, "Администратор")){

admin.setVisibility(View.GONE);

}

else{

admin.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

Intent intent1 = new Intent(ProfileActivity.this, AdminActivity.class);

startActivity(intent1);

}

});

}

apiInterface = ServiceBuilder.buildRequest().create(ApiInterface.class);

Call<User> getUser = apiInterface.getUser(StaticData.user\_id);

Context cont = this;

getUser.enqueue(new Callback<User>() {

@Override

public void onResponse(Call<User> call, Response<User> response) {

if (response.isSuccessful()){

User user = response.body();

tv.setText(user.getLogin());

Picasso.with(cont).load("https://lolpigg.pythonanywhere.com" + user.getAvatarPath()).into(img);

rd.setText("Дата регистрации: " + user.getRegistrationDate());

}

else {

Toast.makeText(ProfileActivity.this, "Ошибка запроса", Toast.LENGTH\_SHORT).show();

}

}

@Override

public void onFailure(Call<User> call, Throwable t) {

Toast.makeText(cont, "Ошибка сервера", Toast.LENGTH\_SHORT).show();

}

});

exit.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

Intent intent = new Intent(ProfileActivity.this, MainActivity.class);

startActivity(intent);

}

});

change.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

Intent intent = new Intent(ProfileActivity.this, ChangePasswordActivity.class);

startActivity(intent);

}

});

}

}

## AdminActivity

package com.example.librarystar;

import android.content.Intent;

import android.os.Bundle;

import android.view.View;

import android.widget.Button;

import androidx.activity.EdgeToEdge;

import androidx.appcompat.app.AppCompatActivity;

import androidx.core.graphics.Insets;

import androidx.core.view.ViewCompat;

import androidx.core.view.WindowInsetsCompat;

public class AdminActivity extends AppCompatActivity {

Button add\_book, add\_auth;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

EdgeToEdge.enable(this);

setContentView(R.layout.activity\_admin);

ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {

Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());

v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);

return insets;

});

add\_auth = findViewById(R.id.admin\_add\_author);

add\_book = findViewById(R.id.admin\_add\_book);

add\_auth.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

Intent intent = new Intent(AdminActivity.this, AddAuthorActivity.class);

startActivity(intent);

}

});

add\_book.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

Intent intent = new Intent(AdminActivity.this, AddBookActivity.class);

startActivity(intent);

}

});

}

}

## AddBookActivity

package com.example.librarystar;

import android.content.Context;

import android.content.Intent;

import android.net.Uri;

import android.os.Bundle;

import android.provider.MediaStore;

import android.util.Base64;

import android.view.View;

import android.widget.ArrayAdapter;

import android.widget.Button;

import android.widget.EditText;

import android.widget.Spinner;

import android.widget.TextView;

import android.widget.Toast;

import androidx.activity.EdgeToEdge;

import androidx.annotation.Nullable;

import androidx.appcompat.app.AppCompatActivity;

import androidx.core.graphics.Insets;

import androidx.core.view.ViewCompat;

import androidx.core.view.WindowInsetsCompat;

import com.example.librarystar.Models.Author;

import com.example.librarystar.Models.AuthorBooks;

import com.example.librarystar.Models.Book;

import com.example.librarystar.Models.Genre;

import com.example.librarystar.Models.User;

import java.io.IOException;

import java.io.InputStream;

import java.util.ArrayList;

import retrofit2.Call;

import retrofit2.Callback;

import retrofit2.Response;

public class AddBookActivity extends AppCompatActivity {

EditText name, description, year;

Spinner publisherSpinner, genreSpinner, authorSpinner;

Button add\_auth, add\_pdf, add\_img, save;

TextView auth\_tv;

ApiInterface apiInterface;

Context cont;

ArrayList<Genre> genres;

ArrayList<User> publishers;

ArrayList<Author> authors;

ArrayList<Author> addedAuthors;

String base64\_pdf;

String base64\_img;

private static final int PICK\_PDF\_REQUEST = 1;

private static final int PICK\_IMAGE\_REQUEST = 2;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

EdgeToEdge.enable(this);

setContentView(R.layout.activity\_add\_book);

ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {

Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());

v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);

return insets;

});

cont = this;

base64\_pdf = "";

base64\_img = "";

name = findViewById(R.id.add\_book\_name);

description = findViewById(R.id.add\_book\_description);

year = findViewById(R.id.add\_book\_year);

publisherSpinner = (Spinner) findViewById(R.id.add\_book\_publisher\_spinner);

genreSpinner = (Spinner) findViewById(R.id.add\_book\_genre\_spinner);

authorSpinner = (Spinner) findViewById(R.id.add\_book\_author\_spinner);

add\_auth = findViewById(R.id.add\_book\_add\_author);

add\_pdf = findViewById(R.id.add\_book\_add\_pdf);

add\_img = findViewById(R.id.add\_book\_add\_image);

save = findViewById(R.id.add\_book\_save);

auth\_tv = findViewById(R.id.addedAuthorsTB);

addedAuthors = new ArrayList<>();

apiInterface = ServiceBuilder.buildRequest().create(ApiInterface.class);

Call<ArrayList<User>> getPublishers = apiInterface.getUsersByRole(3);

getPublishers.enqueue(new Callback<ArrayList<User>>() {

@Override

public void onResponse(Call<ArrayList<User>> call, Response<ArrayList<User>> response) {

if (response.isSuccessful()){

publishers = response.body();

ArrayList<String> publisher\_names = new ArrayList<>();

for (User publisher: publishers) {

publisher\_names.add(publisher.getPublisherName());

}

ArrayAdapter<String> adapter = new ArrayAdapter<>(cont, android.R.layout.simple\_spinner\_item, publisher\_names);

adapter.setDropDownViewResource(android.R.layout.simple\_spinner\_dropdown\_item);

publisherSpinner.setAdapter(adapter);

}

else {

Toast.makeText(cont, "Ошибка сервера.", Toast.LENGTH\_SHORT).show();

}

}

@Override

public void onFailure(Call<ArrayList<User>> call, Throwable t) {

Toast.makeText(cont, "Ошибка запросов.", Toast.LENGTH\_SHORT).show();

}

});

Call<ArrayList<Genre>> getGenres = apiInterface.getGenresList();

getGenres.enqueue(new Callback<ArrayList<Genre>>() {

@Override

public void onResponse(Call<ArrayList<Genre>> call, Response<ArrayList<Genre>> response) {

if (response.isSuccessful()){

genres = response.body();

ArrayList<String> genre\_names = new ArrayList<>();

for (Genre genre: genres) {

genre\_names.add(genre.getName());

}

ArrayAdapter<String> adapter = new ArrayAdapter<>(cont, android.R.layout.simple\_spinner\_item, genre\_names);

adapter.setDropDownViewResource(android.R.layout.simple\_spinner\_dropdown\_item);

genreSpinner.setAdapter(adapter);

}

else {

Toast.makeText(cont, "Ошибка сервера.", Toast.LENGTH\_SHORT).show();

}

}

@Override

public void onFailure(Call<ArrayList<Genre>> call, Throwable t) {

Toast.makeText(cont, "Ошибка запросов.", Toast.LENGTH\_SHORT).show();

}

});

Call<ArrayList<Author>> getAuthors = apiInterface.getAuthorsList();

getAuthors.enqueue(new Callback<ArrayList<Author>>() {

@Override

public void onResponse(Call<ArrayList<Author>> call, Response<ArrayList<Author>> response) {

if (response.isSuccessful()){

authors = response.body();

ArrayList<String> author\_names = new ArrayList<>();

for (Author author: authors) {

author\_names.add(author.getFull\_name());

}

ArrayAdapter<String> adapter = new ArrayAdapter<>(cont, android.R.layout.simple\_spinner\_item, author\_names);

adapter.setDropDownViewResource(android.R.layout.simple\_spinner\_dropdown\_item);

authorSpinner.setAdapter(adapter);

}

else {

Toast.makeText(cont, "Ошибка сервера.", Toast.LENGTH\_SHORT).show();

}

}

@Override

public void onFailure(Call<ArrayList<Author>> call, Throwable t) {

Toast.makeText(cont, "Ошибка запросов.", Toast.LENGTH\_SHORT).show();

}

});

add\_auth.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

Author author = authors.get((int) authorSpinner.getSelectedItemId());

for (Author auth: addedAuthors) {

if (auth.getId() == author.getId()){

Toast.makeText(cont, "Выбранный автор уже добавлен!", Toast.LENGTH\_SHORT).show();

return;

}

}

addedAuthors.add(author);

auth\_tv.setText("Добавлено авторов: " + addedAuthors.size());

}

});

add\_pdf.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

Intent intent = new Intent(Intent.ACTION\_GET\_CONTENT);

intent.setType("application/pdf");

startActivityForResult(intent, PICK\_PDF\_REQUEST);

}

});

add\_img.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

Intent intent = new Intent(Intent.ACTION\_PICK, MediaStore.Images.Media.EXTERNAL\_CONTENT\_URI);

startActivityForResult(intent, PICK\_IMAGE\_REQUEST);

}

});

save.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

try {

String nameValue = name.getText().toString();

String descriptionValue = description.getText().toString();

String yearValue = year.getText().toString();

String publisherName = publisherSpinner.getSelectedItem().toString();

String genreName = genreSpinner.getSelectedItem().toString();

User publisher = null;

Genre genre = null;

for (User publ: publishers) {

if (publ.getPublisherName().equals(publisherName)){

publisher = publ;

}

}

for (Genre genr: genres){

if (genr.getName().equals(genreName)){

genre = genr;

}

}

if (nameValue.isEmpty() || descriptionValue.isEmpty() || yearValue.isEmpty()) {

Toast.makeText(cont, "Пожалуйста, заполните все поля", Toast.LENGTH\_SHORT).show();

} else if (base64\_pdf.equals("") || base64\_img.equals("")){

Toast.makeText(cont, "Пожалуйста, добавьте pdf и картинку книги.", Toast.LENGTH\_SHORT).show();

}

else {

Book book = new Book(nameValue, Integer.parseInt(yearValue), base64\_img, base64\_pdf, publisher.getId(), genre.getId(), false, null, true);

Call<Book> addBookCall = apiInterface.addBook(book);

addBookCall.enqueue(new Callback<Book>() {

@Override

public void onResponse(Call<Book> call, Response<Book> response) {

if (response.isSuccessful()){

Book book = response.body();

Toast.makeText(cont, "Книга успешно добавлена!", Toast.LENGTH\_SHORT).show();

AddAuthors(book.getId());

description.setText("");

name.setText("");

year.setText("");

base64\_img = "";

base64\_pdf = "";

addedAuthors.clear();

}

else {

Toast.makeText(cont, "Ошибка сервера или данных.", Toast.LENGTH\_SHORT).show();

}

}

@Override

public void onFailure(Call<Book> call, Throwable t) {

Toast.makeText(cont, "Ошибка запросов.", Toast.LENGTH\_SHORT).show();

}

});

}

}

catch (Exception e){

Toast.makeText(cont, "Ошибка добавления книги! Проверьте вводимых формат данных", Toast.LENGTH\_SHORT).show();

}

}

});

}

@Override

protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {

super.onActivityResult(requestCode, resultCode, data);

if (requestCode == PICK\_PDF\_REQUEST && resultCode == RESULT\_OK && data != null) {

Uri selectedPdfUri = data.getData();

try {

InputStream inputStream = getContentResolver().openInputStream(selectedPdfUri);

byte[] pdfBytes = new byte[inputStream.available()];

inputStream.read(pdfBytes);

inputStream.close();

base64\_pdf = Base64.encodeToString(pdfBytes, Base64.DEFAULT);

Toast.makeText(cont, "Pdf успешно добавлен", Toast.LENGTH\_SHORT).show();

} catch (IOException e) {

e.printStackTrace();

}

}

else if (requestCode == PICK\_IMAGE\_REQUEST && resultCode == RESULT\_OK && data != null && data.getData() != null) {

Uri imageUri = data.getData();

try {

InputStream inputStream = getContentResolver().openInputStream(imageUri);

byte[] imageBytes = new byte[inputStream.available()];

inputStream.read(imageBytes);

inputStream.close();

base64\_img = Base64.encodeToString(imageBytes, Base64.DEFAULT);

Toast.makeText(cont, "Изображение успешно добавлено", Toast.LENGTH\_SHORT).show();

} catch (IOException e) {

e.printStackTrace();

}

}

}

private void AddAuthors(int book\_id){

for (Author auth: addedAuthors){

Call<AuthorBooks> addAB = apiInterface.addAuthorBook(new AuthorBooks(auth.getId(), book\_id));

addAB.enqueue(new Callback<AuthorBooks>() {

@Override

public void onResponse(Call<AuthorBooks> call, Response<AuthorBooks> response) {

if (response.isSuccessful()){

}

else {

Toast.makeText(cont, "Ошибка сервера при добавлении авторов.", Toast.LENGTH\_SHORT).show();

}

}

@Override

public void onFailure(Call<AuthorBooks> call, Throwable t) {

Toast.makeText(cont, "Ошибка запросов при добавлении авторов.", Toast.LENGTH\_SHORT).show();

}

});

}

Intent intent = new Intent(AddBookActivity.this, UserMainActivity.class);

startActivity(intent);

}

}

## Код моделей БД

from django.db import models

from django.core.validators import FileExtensionValidator

class Action(models.Model):

name = models.CharField(max\_length=50)

class Notification(models.Model):

description = models.CharField(max\_length=1024)

action = models.ForeignKey(Action, on\_delete=models.CASCADE)

book = models.ForeignKey('Book', on\_delete=models.CASCADE)

is\_accepted = models.BooleanField(null=True)

discard\_text = models.CharField(max\_length=1024, null=True)

class Author(models.Model):

full\_name = models.CharField(max\_length=100)

image\_path = models.ImageField(upload\_to='img/authors/', null=True)

year\_of\_birth = models.IntegerField()

year\_of\_death = models.IntegerField(null=True)

class AuthorBooks(models.Model):

author = models.ForeignKey(Author, on\_delete=models.CASCADE)

book = models.ForeignKey('Book', on\_delete=models.CASCADE)

class Book(models.Model):

name = models.CharField(max\_length=100)

description = models.CharField(max\_length=300, default="Описания нет.")

year\_of\_creating = models.IntegerField()

image\_path = models.ImageField(upload\_to='img/books/')

pdf\_path = models.TextField()

publisher = models.ForeignKey('User', on\_delete=models.CASCADE)

genre = models.ForeignKey('Genre', on\_delete=models.CASCADE)

is\_deleted = models.BooleanField()

delete\_text = models.CharField(max\_length=255, null=True)

is\_available = models.BooleanField()

class Genre(models.Model):

name = models.CharField(max\_length=50)

icon\_path = models.ImageField(upload\_to='img/genres/', null=True)

first\_color = models.CharField(max\_length=7)

second\_color = models.CharField(max\_length=7)

class UserBooks(models.Model):

book = models.ForeignKey(Book, on\_delete=models.CASCADE)

user = models.ForeignKey('User', on\_delete=models.CASCADE)

is\_active = models.BooleanField(default=True)

class User(models.Model):

login = models.CharField(max\_length=255)

password = models.CharField(max\_length=255)

registration\_date = models.DateField()

avatar\_path = models.ImageField(null=True, upload\_to='img/users/')

last\_book = models.ForeignKey(Book, on\_delete=models.SET\_NULL, null=True)

last\_book\_page = models.IntegerField(null=True)

role = models.ForeignKey('Role', on\_delete=models.CASCADE)

publisher\_name = models.CharField(max\_length=255, null=True)

password\_restore\_date = models.DateField(null=True)

class Role(models.Model):

name = models.CharField(max\_length=50)

## Код views-функций API

from django.shortcuts import render, get\_object\_or\_404

from django.http import JsonResponse

from rest\_framework.decorators import api\_view

from rest\_framework.parsers import JSONParser

from rest\_framework.response import Response

from rest\_framework import status

from PIL import Image

from .models import \*

from .serializers import \*

from rest\_framework.response import Response

from rest\_framework.pagination import PageNumberPagination

import io

import base64

class CustomPagination(PageNumberPagination):

page\_size = 2

page\_size\_query\_param = 'page\_size'

@api\_view(['GET'])

def get\_users\_by\_role\_id(request, role):

try:

users = User.objects.filter(role=role)

serializer = UserSerializer(users, many=True)

return Response(serializer.data, status=status.HTTP\_200\_OK)

except Exception as e:

return Response({'error': str(e)}, status=status.HTTP\_400\_BAD\_REQUEST)

@api\_view(['GET'])

def get\_books\_by\_user(request, user):

try:

user\_books = UserBooks.objects.filter(user=user, is\_active=True)

books = [user\_book.book for user\_book in user\_books]

serializer = BookSerializer(books, many=True)

return Response(serializer.data, status=status.HTTP\_200\_OK)

except Exception as e:

return Response({'error': str(e)}, status=status.HTTP\_400\_BAD\_REQUEST)

@api\_view(['GET'])

def get\_books\_by\_user(request, user):

try:

user\_books = UserBooks.objects.filter(user=user, is\_active=True)

books = [user\_book.book for user\_book in user\_books]

serializer = BookSerializer(books, many=True)

return Response(serializer.data, status=status.HTTP\_200\_OK)

except Exception as e:

return Response({'error': str(e)}, status=status.HTTP\_400\_BAD\_REQUEST)

@api\_view(['GET'])

def get\_authors\_by\_book(request, book\_id):

try:

# Получаем книгу по переданному ID

book = Book.objects.get(id=book\_id)

except Book.DoesNotExist:

return Response({'error': 'Book not found'}, status=status.HTTP\_404\_NOT\_FOUND)

# Получаем все записи AuthorBooks, связанные с этой книгой

author\_books = AuthorBooks.objects.filter(book=book)

# Формируем список авторов

authors = [

{

'id': author\_book.author.id,

'full\_name': author\_book.author.full\_name,

'image\_path': request.build\_absolute\_uri(author\_book.author.image\_path.url) if author\_book.author.image\_path else None,

'year\_of\_birth': author\_book.author.year\_of\_birth,

'year\_of\_death': author\_book.author.year\_of\_death,

}

for author\_book in author\_books

]

# Возвращаем список авторов в формате JSON

return Response(authors, status=status.HTTP\_200\_OK)

@api\_view(['POST'])

def auth(request):

if request.method == 'POST':

login = request.data.get('login')

password = request.data.get('password')

if login is None or password is None:

return Response({'error': 'Please provide both login and password'}, status=400)

try:

user = get\_object\_or\_404(User, login=login, password=password)

serializer = UserSerializer(user)

return Response(serializer.data)

except User.DoesNotExist:

return Response({'error': 'Invalid credentials'}, status=400)

else:

return Response({'error': 'Only POST method is allowed'}, status=405)

@api\_view(['GET', 'POST'])

def action\_list\_create(request):

if request.method == 'GET':

paginator = CustomPagination()

actions = Action.objects.all()

result\_page = paginator.paginate\_queryset(actions, request)

serializer = ActionSerializer(result\_page, many=True)

return paginator.get\_paginated\_response(serializer.data)

elif request.method == 'POST':

data = JSONParser().parse(request)

serializer = ActionSerializer(data=data)

if serializer.is\_valid():

serializer.save()

return Response(serializer.data, status=status.HTTP\_201\_CREATED)

return Response(serializer.errors, status=status.HTTP\_400\_BAD\_REQUEST)

@api\_view(['GET', 'PUT', 'DELETE'])

def action\_detail(request, pk):

try:

action = Action.objects.get(pk=pk)

except Action.DoesNotExist:

return Response(status=status.HTTP\_404\_NOT\_FOUND)

if request.method == 'GET':

serializer = ActionSerializer(action)

return Response(serializer.data)

elif request.method == 'PUT':

data = JSONParser().parse(request)

serializer = ActionSerializer(action, data=data)

if serializer.is\_valid():

serializer.save()

return Response(serializer.data)

return Response(serializer.errors, status=status.HTTP\_400\_BAD\_REQUEST)

elif request.method == 'DELETE':

action.delete()

return Response(status=status.HTTP\_204\_NO\_CONTENT)

@api\_view(['GET', 'POST'])

def notification\_list\_create(request):

if request.method == 'GET':

notifications = Notification.objects.all()

serializer = NotificationSerializer(notifications, many=True)

return Response(serializer.data)

elif request.method == 'POST':

data = JSONParser().parse(request)

serializer = NotificationSerializer(data=data)

if serializer.is\_valid():

serializer.save()

return Response(serializer.data, status=status.HTTP\_201\_CREATED)

return Response(serializer.errors, status=status.HTTP\_400\_BAD\_REQUEST)

@api\_view(['GET', 'PUT', 'DELETE'])

def notification\_detail(request, pk):

try:

notification = Notification.objects.get(pk=pk)

except Notification.DoesNotExist:

return Response(status=status.HTTP\_404\_NOT\_FOUND)

if request.method == 'GET':

serializer = NotificationSerializer(notification)

return Response(serializer.data)

elif request.method == 'PUT':

data = JSONParser().parse(request)

serializer = NotificationSerializer(notification, data=data)

if serializer.is\_valid():

serializer.save()

return Response(serializer.data)

return Response(serializer.errors, status=status.HTTP\_400\_BAD\_REQUEST)

elif request.method == 'DELETE':

notification.delete()

return Response(status=status.HTTP\_204\_NO\_CONTENT)

@api\_view(['GET', 'POST'])

def author\_list\_create(request):

if request.method == 'GET':

authors = Author.objects.all()

serializer = AuthorSerializer(authors, many=True)

return Response(serializer.data)

elif request.method == 'POST':

data = JSONParser().parse(request)

serializer = AuthorSerializer(data=data)

if serializer.is\_valid():

serializer.save()

return Response(serializer.data, status=status.HTTP\_201\_CREATED)

return Response(serializer.errors, status=status.HTTP\_400\_BAD\_REQUEST)

@api\_view(['GET', 'PUT', 'DELETE'])

def author\_detail(request, pk):

try:

author = Author.objects.get(pk=pk)

except Author.DoesNotExist:

return Response(status=status.HTTP\_404\_NOT\_FOUND)

if request.method == 'GET':

serializer = AuthorSerializer(author)

return Response(serializer.data)

elif request.method == 'PUT':

data = JSONParser().parse(request)

serializer = AuthorSerializer(author, data=data)

if serializer.is\_valid():

serializer.save()

return Response(serializer.data)

return Response(serializer.errors, status=status.HTTP\_400\_BAD\_REQUEST)

elif request.method == 'DELETE':

author.delete()

return Response(status=status.HTTP\_204\_NO\_CONTENT)

@api\_view(['GET', 'POST'])

def authorbooks\_list\_create(request):

if request.method == 'GET':

author\_books = AuthorBooks.objects.all()

serializer = AuthorBooksSerializer(author\_books, many=True)

return Response(serializer.data)

elif request.method == 'POST':

data = JSONParser().parse(request)

serializer = AuthorBooksSerializer(data=data)

if serializer.is\_valid():

serializer.save()

return Response(serializer.data, status=status.HTTP\_201\_CREATED)

return Response(serializer.errors, status=status.HTTP\_400\_BAD\_REQUEST)

@api\_view(['GET', 'PUT', 'DELETE'])

def authorbooks\_detail(request, pk):

try:

author\_book = AuthorBooks.objects.get(pk=pk)

except AuthorBooks.DoesNotExist:

return Response(status=status.HTTP\_404\_NOT\_FOUND)

if request.method == 'GET':

serializer = AuthorBooksSerializer(author\_book)

return Response(serializer.data)

elif request.method == 'PUT':

data = JSONParser().parse(request)

serializer = AuthorBooksSerializer(author\_book, data=data)

if serializer.is\_valid():

serializer.save()

return Response(serializer.data)

return Response(serializer.errors, status=status.HTTP\_400\_BAD\_REQUEST)

elif request.method == 'DELETE':

author\_book.delete()

return Response(status=status.HTTP\_204\_NO\_CONTENT)

@api\_view(['GET', 'POST'])

def books\_list\_create(request):

if request.method == 'GET':

books = Book.objects.all()

serializer = BookSerializer(books, many=True)

return Response(serializer.data, status=status.HTTP\_200\_OK)

elif request.method == 'POST':

data = JSONParser().parse(request)

serializer = BookSerializer(data=data)

if serializer.is\_valid():

serializer.save()

return Response(serializer.data, status=status.HTTP\_201\_CREATED)

return Response(serializer.errors, status=status.HTTP\_400\_BAD\_REQUEST)

@api\_view(['GET', 'PUT', 'DELETE'])

def books\_detail(request, pk):

try:

book = Book.objects.get(pk=pk)

except Book.DoesNotExist:

return Response(status=status.HTTP\_404\_NOT\_FOUND)

if request.method == 'GET':

serializer = BookSerializer(book)

return Response(serializer.data)

elif request.method == 'PUT':

data = JSONParser().parse(request)

serializer = BookSerializer(book, data=data)

if serializer.is\_valid():

serializer.save()

return Response(serializer.data)

return Response(serializer.errors, status=status.HTTP\_400\_BAD\_REQUEST)

elif request.method == 'DELETE':

book.delete()

return Response(status=status.HTTP\_204\_NO\_CONTENT)

@api\_view(['GET', 'POST'])

def genres\_list\_create(request):

if request.method == 'GET':

genres = Genre.objects.all()

serializer = GenreSerializer(genres, many=True)

return Response(serializer.data)

elif request.method == 'POST':

data = JSONParser().parse(request)

image\_data = data.get('icon\_path')

# Декодируем изображение из base64

image = Image.open(io.BytesIO(base64.b64decode(image\_data)))

# Сохраняем изображение в формате PNG

buffer = io.BytesIO()

image.save(buffer, format='PNG')

# Получаем байтовое представление изображения

image\_png = buffer.getvalue()

# Модифицируем данные, чтобы сохранить байтовое представление изображения

data['icon\_path'] = base64.b64encode(image\_png).decode('utf-8')

serializer = GenreSerializer(data=data)

if serializer.is\_valid():

serializer.save()

return Response(serializer.data, status=status.HTTP\_201\_CREATED)

return Response(serializer.errors, status=status.HTTP\_400\_BAD\_REQUEST)

@api\_view(['GET', 'PUT', 'DELETE'])

def genres\_detail(request, pk):

try:

genre = Genre.objects.get(pk=pk)

except Genre.DoesNotExist:

return Response(status=status.HTTP\_404\_NOT\_FOUND)

if request.method == 'GET':

serializer = GenreSerializer(genre)

return Response(serializer.data)

elif request.method == 'PUT':

data = JSONParser().parse(request)

image\_data = data.get('icon\_path')

# Декодируем изображение из base64

image = Image.open(io.BytesIO(base64.b64decode(image\_data)))

# Сохраняем изображение в формате PNG

buffer = io.BytesIO()

image.save(buffer, format='PNG')

# Получаем байтовое представление изображения

image\_png = buffer.getvalue()

# Модифицируем данные, чтобы сохранить байтовое представление изображения

data['icon\_path'] = base64.b64encode(image\_png).decode('utf-8')

serializer = GenreSerializer(genre, data=data)

if serializer.is\_valid():

serializer.save()

return Response(serializer.data)

return Response(serializer.errors, status=status.HTTP\_400\_BAD\_REQUEST)

elif request.method == 'DELETE':

genre.delete()

return Response(status=status.HTTP\_204\_NO\_CONTENT)

@api\_view(['GET', 'POST'])

def userbooks\_list\_create(request):

if request.method == 'GET':

user\_books = UserBooks.objects.all()

serializer = UserBooksSerializer(user\_books, many=True)

return Response(serializer.data)

elif request.method == 'POST':

data = JSONParser().parse(request)

book\_id = data.get('book')

user\_id = data.get('user')

try:

user\_book = UserBooks.objects.get(book\_id=book\_id, user\_id=user\_id)

if not user\_book.is\_active:

user\_book.is\_active = True

user\_book.save()

serializer = UserBooksSerializer(user\_book)

return Response(serializer.data, status=status.HTTP\_200\_OK)

else:

return Response({'error': 'User book already exists and is active'}, status=status.HTTP\_400\_BAD\_REQUEST)

except UserBooks.DoesNotExist:

serializer = UserBooksSerializer(data=data)

if serializer.is\_valid():

serializer.save()

return Response(serializer.data, status=status.HTTP\_201\_CREATED)

return Response(serializer.errors, status=status.HTTP\_400\_BAD\_REQUEST)

@api\_view(['GET', 'PUT', 'DELETE'])

def userbooks\_detail(request, pk):

try:

user\_book = UserBooks.objects.get(pk=pk)

except UserBooks.DoesNotExist:

return Response(status=status.HTTP\_404\_NOT\_FOUND)

if request.method == 'GET':

serializer = UserBooksSerializer(user\_book)

return Response(serializer.data)

elif request.method == 'PUT':

data = JSONParser().parse(request)

serializer = UserBooksSerializer(user\_book, data=data)

if serializer.is\_valid():

serializer.save()

return Response(serializer.data)

return Response(serializer.errors, status=status.HTTP\_400\_BAD\_REQUEST)

elif request.method == 'DELETE':

user\_book.delete()

return Response(status=status.HTTP\_204\_NO\_CONTENT)

@api\_view(['GET', 'POST'])

def users\_list\_create(request):

if request.method == 'GET':

user = User.objects.all()

serializer = UserSerializer(user, many=True)

return Response(serializer.data)

elif request.method == 'POST':

data = JSONParser().parse(request)

serializer = UserSerializer(data=data)

if serializer.is\_valid():

serializer.save()

return Response(serializer.data, status=status.HTTP\_201\_CREATED)

return Response(serializer.errors, status=status.HTTP\_400\_BAD\_REQUEST)

@api\_view(['GET', 'PUT', 'DELETE'])

def users\_detail(request, pk):

try:

user = User.objects.get(pk=pk)

except User.DoesNotExist:

return Response(status=status.HTTP\_404\_NOT\_FOUND)

if request.method == 'GET':

serializer = UserSerializer(user)

return Response(serializer.data)

elif request.method == 'PUT':

data = JSONParser().parse(request)

serializer = UserSerializer(user, data=data)

if serializer.is\_valid():

serializer.save()

return Response(serializer.data)

return Response(serializer.errors, status=status.HTTP\_400\_BAD\_REQUEST)

elif request.method == 'DELETE':

user.delete()

return Response(status=status.HTTP\_204\_NO\_CONTENT)

@api\_view(['GET', 'POST'])

def roles\_list\_create(request):

if request.method == 'GET':

role = Role.objects.all()

serializer = RoleSerializer(role, many=True)

return Response(serializer.data)

elif request.method == 'POST':

data = JSONParser().parse(request)

serializer = RoleSerializer(data=data)

if serializer.is\_valid():

serializer.save()

return Response(serializer.data, status=status.HTTP\_201\_CREATED)

return Response(serializer.errors, status=status.HTTP\_400\_BAD\_REQUEST)

@api\_view(['GET', 'PUT', 'DELETE'])

def roles\_detail(request, pk):

try:

role = Role.objects.get(pk=pk)

except Role.DoesNotExist:

return Response(status=status.HTTP\_404\_NOT\_FOUND)

if request.method == 'GET':

serializer = RoleSerializer(role)

return Response(serializer.data)

elif request.method == 'PUT':

data = JSONParser().parse(request)

serializer = RoleSerializer(role, data=data)

if serializer.is\_valid():

serializer.save()

return Response(serializer.data)

return Response(serializer.errors, status=status.HTTP\_400\_BAD\_REQUEST)

elif request.method == 'DELETE':

role.delete()

return Response(status=status.HTTP\_204\_NO\_CONTENT)

## Код сериализаторов моделей

from rest\_framework import serializers

from .models import Action, Notification, Author, AuthorBooks, Book, Genre, UserBooks, User, Role

from drf\_extra\_fields.fields import Base64ImageField, Base64FileField

class ActionSerializer(serializers.ModelSerializer):

class Meta:

model = Action

fields = '\_\_all\_\_'

class NotificationSerializer(serializers.ModelSerializer):

class Meta:

model = Notification

fields = '\_\_all\_\_'

class AuthorSerializer(serializers.ModelSerializer):

image\_path = Base64ImageField()

class Meta:

model = Author

fields = '\_\_all\_\_'

class AuthorBooksSerializer(serializers.ModelSerializer):

class Meta:

model = AuthorBooks

fields = '\_\_all\_\_'

class BookSerializer(serializers.ModelSerializer):

image\_path = Base64ImageField()

#pdf\_path = Base64FileField()

class Meta:

model = Book

fields = '\_\_all\_\_'

class GenreSerializer(serializers.ModelSerializer):

icon\_path = Base64ImageField()

class Meta:

model = Genre

fields = '\_\_all\_\_'

class UserBooksSerializer(serializers.ModelSerializer):

class Meta:

model = UserBooks

fields = '\_\_all\_\_'

class UserSerializer(serializers.ModelSerializer):

avatar\_path = Base64ImageField()

class Meta:

model = User

fields = '\_\_all\_\_'

class RoleSerializer(serializers.ModelSerializer):

class Meta:

model = Role

fields = '\_\_all\_\_'