# 项目总结报告

项目名称：随手记App

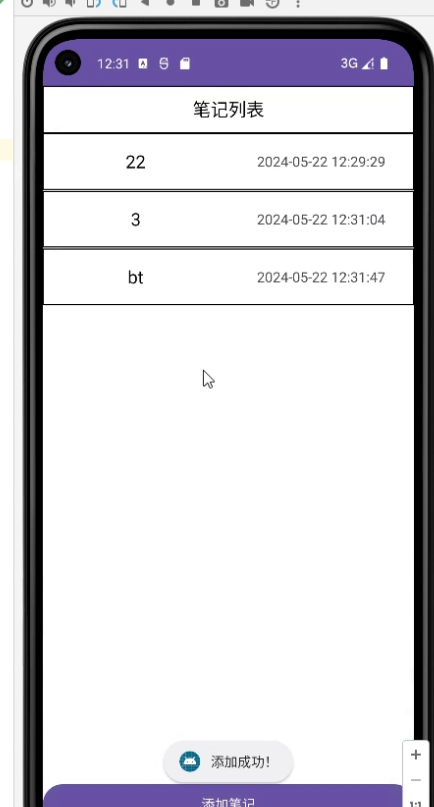
学号姓名：

功能设计及运行结果：

实现了登陆界面；笔记列表；添加笔记，并且添加笔记时会自动获取并记录当前时间；删除笔记。使用了Room将数据进行持久化保存，使用了Fragment，并且可以从明细页返回列表页

登陆界面 添加笔记

笔记列表 删除笔记

**核心代码：**

这段代码实现了一个数据库的创建和访问功能。使用了Android的Room Persistence Library来创建一个名为MyDatabase的数据库，并定义了一个名为MyNoteDao的数据访问对象。数据库中包含一个名为"MyNote"的实体类，用于存储笔记的相关信息。通过调用getInstance()方法，可以获取数据库的单例实例。如果数据库实例为空，则会使用Room.databaseBuilder()方法创建一个数据库实例，并指定数据库名称、上下文和实体类。然后，可以通过调用myNoteDao()方法获取用于执行数据库操作的MyNoteDao对象。

|  |
| --- |
| *//创建数据库* @Database(entities = {MyNote.class},version = 1) public abstract class MyDatabase extends RoomDatabase {  *//定义数据库名称* private static final String *DATABASE\_NAME* = "my\_db";   private static MyDatabase *databaseInstance*;   public static synchronized MyDatabase getInstance(Context context){  if(*databaseInstance* == null){  *databaseInstance* = Room.*databaseBuilder*(  context.getApplicationContext(),  MyDatabase.class,  *DATABASE\_NAME*)  .build();  }  return *databaseInstance*;  }  public abstract MyNoteDao myNoteDao(); } |

这段代码实现的是笔记的主界面，主界面采用了Fragment技术，分别包含了两个Fragment：One和Two。其中One Fragment用于展示用户的笔记列表，Two Fragment用于添加新的笔记。在MainActivity中，首先定义了一个FragmentManager对象和两个Fragment对象one和two。其中FragmentManager用于管理Fragment对象的生命周期，one和two则用于存储对应的Fragment对象。在onCreate方法中，通过调用setContentView方法设置了布局文件activity\_main作为主界面的内容。接着，调用setChioceItem方法，该方法根据传入的参数index选择显示不同的Fragment。如果index为0，则显示One Fragment；如果index为1，则显示Two Fragment。在setChioceItem方法中，首先创建一个FragmentTransaction对象，该对象用于对Fragment进行添加、移除、替换、隐藏等操作。然后通过hideFragments方法来隐藏其他的Fragment，只显示当前选择的Fragment。最后根据传入的index参数，判断要显示哪个Fragment，并执行相应的操作。在hideFragments方法中，如果one对象或two对象不为空，则将其隐藏。MainActivity类还提供了一个setview方法，该方法用于在其他地方调用来切换Fragment的显示。

|  |
| --- |
| @Override protected void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  setContentView(R.layout.*activity\_main*);  *fragmentManager* = getSupportFragmentManager();   *setChioceItem*(0); *// 初始化页面加载时显示第一个选项卡* }  @SuppressLint("ResourceAsColor") private static void setChioceItem(final int index) {  FragmentTransaction fragmentTransaction = *fragmentManager*.beginTransaction();  *hideFragments*(fragmentTransaction);  switch (index) {  case 0:  if (*one* == null) {  *one* = new One();  fragmentTransaction.add(R.id.*content*, *one*);  }  else {  fragmentTransaction.show(*one*);  *one*.onResume();  }   break;  case 1:  if (*two* == null) {  *two* = new Two();  fragmentTransaction.add(R.id.*content*, *two*);  } else {  fragmentTransaction.show(*two*);  *two*.onResume();  }  break;  }  fragmentTransaction.commit(); } private static void hideFragments(FragmentTransaction fragmentTransaction) {  if (*one* != null) {  fragmentTransaction.hide(*one*);  }  if (*two*!= null) {  fragmentTransaction.hide(*two*);  }  } public static void setview(int n){  *setChioceItem*(n); } |

这段代码定义了一个数据访问对象，用于执行与MyNote实体相关的数据库操作。@Insert注解标记的insertMyNote方法用于向数据库中插入新的 MyNote 对象。@Delete注解标记的deleteMyNote方法用于从数据库中删除指定的 MyNote 对象。@Update注解标记的updateMyNote方法用于更新数据库中已有的 MyNote 对象。@Query("SELECT \* FROM mynote")注解标记的getMyNoteList方法用于从数据库中查询并返回所有的 MyNote对象列表。@Query("SELECT \* FROM mynote WHERE id = :id")注解标记的getMyNoteById方法用于根据给定的 id 从数据库中查询并返回对应的 MyNote 对象。

|  |
| --- |
| @Dao public interface MyNoteDao {  @Insert  void insertMyNote(MyNote myNote);  @Delete  void deleteMyNote(MyNote myNote);   @Update  void updateMyNote(MyNote myNote);   @Query("SELECT \* FROM mynote")  List<MyNote> getMyNoteList();   @Query("SELECT \* FROM mynote WHERE id = :id")  MyNote getMyNoteById(int id);  } |

这段代码是一个Fragment类的实现，onCreateView方法中，设置了布局文件R.layout.one并初始化了界面元素。其中，通过调用setListView方法设置了列表视图，并为添加按钮add添加了点击事件监听器。点击添加按钮时，会将标识MainActivity.sign置为0，并调用MainActivity.setview(1)方法。setListView方法中，首先获取了数据库实例 ，然后创建了一个新线程，在该线程中从数据库中获取笔记列表数据，并将其显示在列表视图中。对于每个笔记，都将标题和时间存储到数据列表中，并使用SimpleAdapter进行数据绑定。同时，为列表项设置了点击事件监听器，当用户点击某个笔记时，会将选中的笔记信息存储到MainActivity.myNote中，并将标识MainActivity.sign置为1，最后调用MainActivity.setview(1)方法。onResume方法中，重新调用setListView方法，以便在Fragment恢复显示时刷新列表数据。

|  |
| --- |
| import androidx.fragment.app.Fragment; import com.fuchenkai.Note.MainActivity; import com.fuchenkai.Note.MyDatabase; import com.fuchenkai.Note.MyNote; import com.fuchenkai.Note.R; import java.util.ArrayList; import java.util.Calendar; import java.util.HashMap; import java.util.List; import java.util.Map;  public class One extends Fragment {  private View view;  ListView listView;  private Button add;  private List<Map<String, Object>> data;  private List<MyNote> list;  @Override  public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {  view = inflater.inflate(R.layout.*one*, container, false);   setListView();  add = view.findViewById(R.id.*add*);  add.setOnClickListener(new View.OnClickListener() {  @Override  public void onClick(View v) {  MainActivity.*sign*=0;  MainActivity.*setview*(1);  }  });  return view;  }   private void setListView() {  listView =view.findViewById(R.id.*listView*);  MyDatabase myDatabase = MyDatabase.*getInstance*(getContext());  new Thread(new Runnable() {  @Override  public void run() {  list = myDatabase.myNoteDao().getMyNoteList();  if(list!=null){  data = new ArrayList<>();  for (int i = 0; i < list.size(); i++) {  Map<String, Object> map = new HashMap<String, Object>();  map.put("title",list.get(i).title);  map.put("time", list.get(i).time);  data.add(map);  }  SimpleAdapter simpleAdapter = new SimpleAdapter(getContext(),  data,  R.layout.*item*,  new String[]{"title", "time"},  new int[]{R.id.*item\_title*, R.id.*item\_time*});  listView.setAdapter(simpleAdapter);  listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {  @Override  public void onItemClick(AdapterView<?> parent, View view, int position, long id) {  MainActivity.*myNote* = list.get(position);  MainActivity.*sign*=1;  MainActivity.*setview*(1);  }  });  }  }  }).start();  }    @Override  public void onResume() {  super.onResume();  setListView();  } } |

实现思路及心得：

程序的设计思路就是先创建一个数据库用于存储笔记数据，使用一个类表示笔记对象，包含笔记的属性，再创建一个接口用于访问数据库，提供增删改查等操作。还需要仓库类用于管理笔记接口和提供业务逻辑，如获取笔记列表、添加笔记、删除笔记等操作，最后就是用Fragment与用户进行交互，提供获取笔记列表、添加笔记、删除笔记等方法。这学期学习了安卓开发Roo数据库让我受益匪浅，了解了android的基础知识，如Activity、Fragment、布局等，以及掌握了相关的编程语言。通过这个大作业还深入了解了一下Room数据库，更好地理解和掌握了Room数据库的各种功能和用法。本应用不足的是没有对登录时的用户进行检验，也没有记录用户的登录信息，将来可以优化登录功能，可以向用户提供注册的接口，并在登陆时进行校验，并记录用户数据。