ГУАП

КАФЕДРА № 43

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Старший преподаватель |  |  |  | С.В. Щекин |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №8 |
| Освоение работы с базами данных в мобильном приложении |
| по курсу: Программирование мобильных устройств |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | 4134к |  |  |  | Д.В. Самарин |
|  |  |  | подпись, дата |  | инициалы, фамилия |

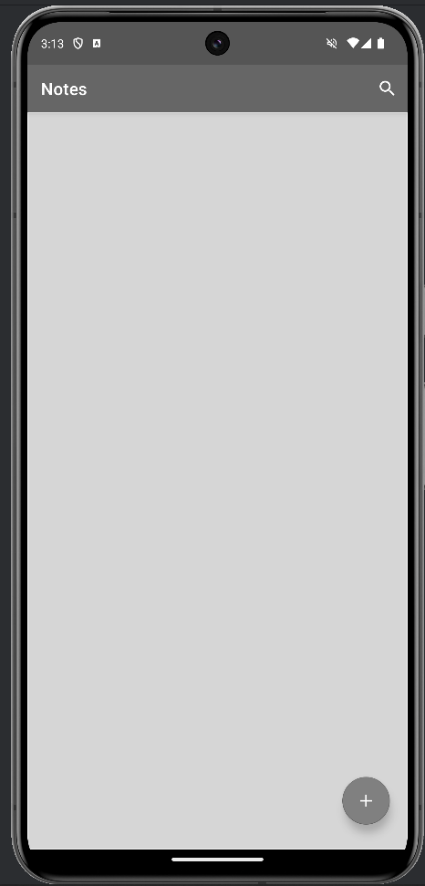
Санкт-Петербург 2024

Задание:

Разработать и отладить мобильное приложение, обеспечивающее подключение и работу с базой данных (например, SQLite для ОС Android). Приложение должно реализовывать создание, чтение, изменение и удаление отдельных записей.

Ход работы

В ходе работы было реализовано приложение заметок:



## Листинг

|  |
| --- |
| MainActivity  package com.example.notes  import android.os.Bundle import android.view.View.OnCreateContextMenuListener import androidx.activity.enableEdgeToEdge import androidx.appcompat.app.AppCompatActivity import androidx.core.view.ViewCompat import androidx.core.view.WindowInsetsCompat import androidx.lifecycle.ViewModelProvider import com.example.notes.database.NoteDatabase import com.example.notes.repository.NoteRepository import com.example.notes.viewmodel.NoteViewModel import com.example.notes.viewmodel.NoteViewModelFactory  class MainActivity : AppCompatActivity() {   lateinit var noteViewModel: NoteViewModel   override fun onCreate(savedInstanceState: Bundle?){  super.onCreate(savedInstanceState)  setContentView(R.layout.*activity\_main*)   setupViewModel()    }      private fun setupViewModel(){  val noteRepository = NoteRepository(NoteDatabase( this))  val viewModelProviderFactory = NoteViewModelFactory(*application*, noteRepository)  noteViewModel = ViewModelProvider( this, viewModelProviderFactory) [NoteViewModel::class.*java* ]     }  }  AddNoteFragment  package com.example.notes.fragments  import android.os.Bundle import androidx.fragment.app.Fragment import android.view.LayoutInflater import android.view.Menu import android.view.MenuInflater import android.view.MenuItem import android.view.View import android.view.ViewGroup import android.widget.Toast import androidx.core.view.MenuHost import androidx.core.view.MenuProvider import androidx.lifecycle.Lifecycle import androidx.navigation.findNavController import com.example.notes.MainActivity import com.example.notes.R import com.example.notes.databinding.FragmentAddNoteBinding import com.example.notes.model.Note import com.example.notes.viewmodel.NoteViewModel   class AddNoteFragment : Fragment(R.layout.*fragment\_add\_note*),MenuProvider {    private var addNoteBinding:FragmentAddNoteBinding? = null  private val binding get() = addNoteBinding!!    private lateinit var notesViewModel: NoteViewModel   private lateinit var addNoteView : View    override fun onCreateView(  inflater: LayoutInflater, container: ViewGroup?,  savedInstanceState: Bundle?  ): View? {  // Inflate the layout for this fragment    addNoteBinding = FragmentAddNoteBinding.inflate(inflater,container, false)  return binding.*root* }   override fun onViewCreated(view: View, savedInstanceState: Bundle?) {  super.onViewCreated(view, savedInstanceState)   val menuHost: MenuHost = requireActivity()  menuHost.addMenuProvider(this, *viewLifecycleOwner*, Lifecycle.State.*RESUMED*)   notesViewModel = (*activity* as MainActivity).noteViewModel   addNoteView = view  }    private fun saveNote(view: View){  val noteTitle = binding.addNoteTitle.*text*.toString().*trim*()  val noteDesc = binding.addNoteDesc.*text*.toString().*trim*()   if (noteTitle.*isNotEmpty*()){  val note = Note (0, noteTitle, noteDesc)  notesViewModel.addNote(note)   Toast.makeText(addNoteView.*context*, "Note Saved", Toast.*LENGTH\_SHORT*).show()  view.*findNavController*().popBackStack(R.id.*homeFragment*, false)   } else {  Toast.makeText(addNoteView.*context*, "Pls enter note title", Toast.*LENGTH\_SHORT*).show()  }    }   override fun onCreateMenu(menu: Menu, menuInflater: MenuInflater) {   menu.clear()  menuInflater.inflate(R.menu.*menu\_add\_note*, menu)     }   override fun onMenuItemSelected(menuItem: MenuItem): Boolean {   return when(menuItem.*itemId*){  R.id.*saveMenu* -> {  saveNote(addNoteView)  true  }  else -> false  }   }    override fun onDestroy() {  super.onDestroy()  addNoteBinding = null  }     }  EditNoteFragment  package com.example.notes.fragments  import android.app.AlertDialog import android.os.Bundle import androidx.fragment.app.Fragment import android.view.LayoutInflater import android.view.Menu import android.view.MenuInflater import android.view.MenuItem import android.view.View import android.view.ViewGroup import android.widget.Toast  import androidx.core.view.MenuHost import androidx.core.view.MenuProvider import androidx.lifecycle.Lifecycle   import androidx.navigation.findNavController import androidx.navigation.fragment.navArgs   import com.example.notes.MainActivity import com.example.notes.R  import com.example.notes.databinding.FragmentEditNoteBinding import com.example.notes.model.Note import com.example.notes.viewmodel.NoteViewModel    class EditNoteFragment : Fragment(R.layout.*fragment\_edit\_note*),MenuProvider {     private var editNoteBinding : FragmentEditNoteBinding? = null  private val binding get() = editNoteBinding!!   private lateinit var notesViewModel : NoteViewModel  private lateinit var currentNote:Note   private val args: EditNoteFragmentArgs by *navArgs*()     override fun onCreateView(  inflater: LayoutInflater, container: ViewGroup?,  savedInstanceState: Bundle?  ): View {    editNoteBinding = FragmentEditNoteBinding.inflate(inflater,container,false)   return binding.*root* }   override fun onViewCreated(view: View, savedInstanceState: Bundle?) {  super.onViewCreated(view, savedInstanceState)    val menuHost: MenuHost = requireActivity()  menuHost.addMenuProvider(this, *viewLifecycleOwner*, Lifecycle.State.*RESUMED*)   notesViewModel = (*activity* as MainActivity).noteViewModel   currentNote = args.*note*!!    binding.editNoteTitle.setText(currentNote.noteTitle)  binding.editNoteDesc.setText(currentNote.noteDesc)   binding.editNoteFab.setOnClickListener**{** val noteTitle = binding.editNoteTitle.*text*.toString().*trim*()  val noteDesc = binding.editNoteDesc.*text*.toString().*trim*()   if (noteTitle.*isNotEmpty*()){  val note = Note(currentNote.id, noteTitle, noteDesc)  notesViewModel.updateNote(note)  view.*findNavController*().popBackStack(R.id.*homeFragment*, false)  }else {  Toast.makeText(*context*,"Pls enter note Titile", Toast.*LENGTH\_SHORT*).show()  }    **}** }    private fun deleteNote(){  AlertDialog.Builder(*activity*).*apply* **{** setTitle("Удаление заметки")  setMessage("Вы действительно хотите удалить заметку?")  setPositiveButton("Удалить")**{**\_,\_**->** notesViewModel.deleteNote(currentNote)  Toast.makeText(*context*,"Note Deleted", Toast.*LENGTH\_SHORT*).show()  *view*?.*findNavController*()?.popBackStack(R.id.*homeFragment*, false )  **}** setNegativeButton("Отмена", null)  **}**.create().show()   }   override fun onCreateMenu(menu: Menu, menuInflater: MenuInflater) {  menu.clear()  menuInflater.inflate(R.menu.*menu\_edit\_note*, menu)  }   override fun onMenuItemSelected(menuItem: MenuItem): Boolean {  return when(menuItem.*itemId*){  R.id.*deleteMenu* -> {  deleteNote()  true  } else -> false  }  }   override fun onDestroy() {  super.onDestroy()  editNoteBinding = null  }   }  HomeFragment  package com.example.notes.fragments  import android.os.Bundle import androidx.fragment.app.Fragment import android.view.LayoutInflater import android.view.Menu import android.view.MenuInflater import android.view.MenuItem import android.view.View import android.view.ViewGroup import androidx.appcompat.widget.SearchView //import androidx.widget.SearchView import androidx.core.view.MenuHost import androidx.core.view.MenuProvider import androidx.lifecycle.Lifecycle import androidx.navigation.findNavController import androidx.recyclerview.widget.StaggeredGridLayoutManager import com.example.notes.MainActivity import com.example.notes.R import com.example.notes.adapter.NoteAdapter import com.example.notes.databinding.FragmentHomeBinding import com.example.notes.model.Note import com.example.notes.viewmodel.NoteViewModel   class HomeFragment : Fragment(R.layout.*fragment\_home*), SearchView.OnQueryTextListener,MenuProvider {    private var homeBinding: FragmentHomeBinding? = null  private val binding get() = homeBinding!!     private lateinit var notesViewModel: NoteViewModel   private lateinit var noteAdapter: NoteAdapter   override fun onCreateView(  inflater: LayoutInflater, container: ViewGroup?,  savedInstanceState: Bundle?  ): View? {  // Inflate the layout for this fragment   homeBinding = FragmentHomeBinding.inflate(inflater, container,false)  return binding.*root* }    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {  super.onViewCreated(view, savedInstanceState)   val menuHost: MenuHost = requireActivity()  menuHost.addMenuProvider(this, *viewLifecycleOwner*, Lifecycle.State.*RESUMED*)   notesViewModel = (*activity* as MainActivity).noteViewModel  setupHomeRecyclerView()   binding.addNoteFab.setOnClickListener**{  it**.*findNavController*().navigate(R.id.*action\_homeFragment\_to\_addNoteFragment*)   **}** }     private fun updateUI(note:List<Note>?){  if (note != null) {  if (note.*isNotEmpty*()){  binding.emptyNotesImage.*visibility* = View.*GONE* binding.homeRecyclerView.*visibility* = View.*VISIBLE* }else {  binding.emptyNotesImage.*visibility* = View.*VISIBLE* binding.homeRecyclerView.*visibility* = View.*GONE* }   }  }    private fun setupHomeRecyclerView(){  noteAdapter = NoteAdapter()  binding.homeRecyclerView.*apply* **{** *layoutManager* = StaggeredGridLayoutManager ( 2, StaggeredGridLayoutManager.*VERTICAL*)  setHasFixedSize(true)  *adapter* = noteAdapter  **}** *activity*?.*let***{** notesViewModel.getAllNotes().observe(*viewLifecycleOwner*)**{** note **->** noteAdapter.differ.submitList(note)  updateUI(note)  **}  }** }    private fun searchNote(query: String?) {   val searchQuery = "%$query"   notesViewModel.searchNote(searchQuery).observe(this )**{**list **->** noteAdapter.differ.submitList(list)  **}** }   override fun onQueryTextSubmit(p0: String?): Boolean {   return false   }   override fun onQueryTextChange(newText:String?): Boolean {   if(newText != null){  searchNote(newText)  }  return false  }   override fun onDestroy() {  super.onDestroy()  homeBinding = null  }    override fun onCreateMenu(menu: Menu, menuInflater: MenuInflater) {  menu.clear()  menuInflater.inflate(R.menu.*home\_menu*, menu)   val menuSearch = menu.findItem(R.id.*searchMenu*).*actionView* as SearchView  menuSearch.*isSubmitButtonEnabled* = false  menuSearch.setOnQueryTextListener(this)  }   override fun onMenuItemSelected(menuItem: MenuItem): Boolean {  return false  }   }  NoteDatabase  package com.example.notes.database  import android.content.Context import androidx.room.Database import androidx.room.Room import androidx.room.RoomDatabase import com.example.notes.model.Note import java.util.concurrent.locks.Lock  @Database(entities = [Note::class], version = 1)  abstract class NoteDatabase: RoomDatabase() {  abstract fun getNoteDao(): NoteDao  companion object  {  @Volatile  private var instance: NoteDatabase? = null  private var LOCK = Any()   operator fun invoke(context: Context) = instance ?:  *synchronized*(LOCK)**{** instance ?:  createDatabase(context).*also***{** instance = **it  }  }** private fun createDatabase(context: Context) =  Room.databaseBuilder(  context.*applicationContext*,  NoteDatabase:: class.*java*,  "note\_db"  ).build()   }  }  NoteDao  package com.example.notes.database  import androidx.lifecycle.LiveData import androidx.room.Dao import androidx.room.Delete import androidx.room.Insert import androidx.room.OnConflictStrategy import androidx.room.Query import androidx.room.Update import com.example.notes.model.Note   @Dao interface NoteDao {  @Insert(onConflict = OnConflictStrategy.REPLACE)  suspend fun insertNote(note: Note)   @Update  suspend fun updateNote(note: Note)   @Delete  suspend fun deleteNote(note: Note)   @Query("SELECT \* FROM NOTES ORDER BY id DESC")  fun getAllNotes(): LiveData<List<Note>>   @Query("SELECT \* FROM NOTES WHERE noteTitle LIKE :query OR noteDesc LIKE:query " )  fun searchNote(query: String?): LiveData<List<Note>>   }  NoteAdapter  package com.example.notes.adapter  import android.view.LayoutInflater import android.view.ViewGroup import androidx.navigation.findNavController import androidx.recyclerview.widget.AsyncListDiffer import androidx.recyclerview.widget.DiffUtil import androidx.recyclerview.widget.RecyclerView import com.example.notes.databinding.NoteLayoutBinding import com.example.notes.fragments.HomeFragmentDirections import com.example.notes.model.Note  class NoteAdapter: RecyclerView.Adapter<NoteAdapter.NoteViewHolder>() {   class NoteViewHolder(val itemBinding: NoteLayoutBinding): RecyclerView.ViewHolder(itemBinding.*root*)   private val differCallback = object : DiffUtil.ItemCallback<Note>(){  override fun areItemsTheSame(oldItem: Note, newItem: Note): Boolean {  return oldItem.id == newItem.id &&  oldItem.noteDesc == newItem.noteDesc &&  oldItem.noteTitle == newItem.noteTitle  }   override fun areContentsTheSame(oldItem: Note, newItem: Note): Boolean {  return oldItem == newItem  }   }   val differ = AsyncListDiffer( this, differCallback)    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): NoteViewHolder {  return NoteViewHolder(  NoteLayoutBinding.inflate(LayoutInflater.from(parent.*context*), parent, false)  )  }   override fun getItemCount(): Int {  return differ.*currentList*.size   }   override fun onBindViewHolder(holder: NoteViewHolder, position: Int) {   val currentNote = differ.*currentList*[position]   holder.itemBinding.noteTitle.*text* = currentNote.noteTitle  holder.itemBinding.noteDesc.*text* = currentNote.noteDesc   holder.itemView.setOnClickListener**{** val direction = HomeFragmentDirections.actionHomeFragmentToEditNoteFragment(currentNote)  **it**.*findNavController*().navigate(direction)  **}** }  } |