Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КубГУ»)**

**Факультет компьютерных технологий и прикладной математики**

**Кафедра вычислительных технологий**

**ЛАБОРАТОРНАЯ РАБОТА №6**

**Дисциплина: Программирование мобильных платформ**

Работу выполнил: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ К. В. Стасюк

Направление подготовки: 02.03.02 Фундаментальная информатика и информационные технологии

Преподаватель: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Т. А. Приходько

**Постановка задачи:**

Написание мобильного приложения на Android для добавления планов на день.В работе должны быть зафиксированы следующие пункты :

1)Создание разметки страниц приложения.

2) Создание Активностей. Отслеживание состояний Активности. Реализация Toast.

3)Работа с ресурсами. Добавить меню и изменение локализации рус/англ, используя меню пользователя.

4)Адаптеры в Андроид .

5) Намерения в Android. В этом задании должны быть спроектированы 3-4 Activity для своего приложения.

6)Фрагменты.

7)Хранилища данных. Работа с базой данных SQLite. Дополнить свою задачу базой данных.

**Ход работы:**

Планировщик дел — это удобное приложение, разработанное для эффективного управления вашим временем и задачами. С помощью этого приложения вы можете легко добавлять, редактировать и удалять задачи, а также просматривать планы на каждый день. Интуитивный интерфейс, возможность отображение задач по дням делают использование приложения максимально удобным.

**Создание разметки страниц приложения:**

В приложении было разработаны разметки для трех страниц:

1)Главная страница приложения:

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

Рисунок 1 – главная страница приложения.

**Описание разметки:**

1. **ConstraintLayout:** Основной контейнер, использующий ConstraintLayout для гибкого позиционирования элементов.
2. **Toolbar:** Верхняя полоса, служащая для размещения элементов управления. Содержит текущую дату и кнопку добавления задач.
   * **TextView (tvCurrentDate)**
   * **Button (addButton):** Кнопка добавления новой задачи. При клике на эту кнопку вызывается метод **onAddButtonClick**.
   * **LinearLayout:** Горизонтальный контейнер для дополнительных элементов управления (меню).
3. **ListView (tasksListView):** Виджет списка, предназначенный для отображения списка задач.
4. **LinearLayout (datesBottomBar):** Нижняя панель с датами и кнопками для переключения между днями.
   * **ImageButton (btnPreviousDay):** Кнопка для переключения на предыдущий день. При клике вызывается метод **onNextDayClick**.
   * **TextView (tvYesterday):** Отображение даты вчерашнего дня.
   * **TextView (tvToday):** Отображение текущей даты.
   * **TextView (tvTomorrow):** Отображение даты завтрашнего дня.
   * **ImageButton (btnNextDay):** Кнопка для переключения на следующий день. При клике вызывается метод **onNextDayClick**.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Операционная система

Автоматически созданное описание

Рисунок 2 – страница редактирования.

**Описание разметки :**

1. **ConstraintLayout:** Основной контейнер, использующий ConstraintLayout для гибкого позиционирования элементов.
   * **include:** Включение содержимого другой разметки (fragment\_note\_form.xml). Таким образом, содержимое **fragment\_note\_form.xml** добавляется в текущую разметку.

**Описание разметки элемента (fragment\_note\_form.xml):**

1. **ConstraintLayout:** Основной контейнер, использующий ConstraintLayout для гибкого позиционирования элементов.
2. **LinearLayout:** Вертикальный контейнер для группировки элементов в столбец .
   * **EditText (etNoteName):** Поле для ввода названия плана. Вводится текст.
   * **EditText (etDate):** Поле для ввода даты плана. Используется для ввода даты.
   * **EditText (etTime):** Поле для ввода времени плана. Используется для ввода времени.
   * **EditText (description):** Поле для ввода описания плана. Вводится текст.
   * **Button (btnAction):** Кнопка "Сохранить", которая вызывает действие сохранения плана.
   * **Button (btnClose):** Кнопка "Закрыть", которая вызывает метод **closePage**. Размещена внизу.

Изображение выглядит как текст, снимок экрана, программное обеспечение, число

Автоматически созданное описание

Рисунок 3 – страница добавления.

**Описание разметки :**

1. **ConstraintLayout:** Основной контейнер, использующий ConstraintLayout для гибкого позиционирования элементов.
   * **include:** Включение содержимого другой разметки (fragment\_note\_form.xml). Данный фрагмент описан выше.

**Toast:**

При добавлении и редактировании заметки появляется сообщение Toast о сохранении заметки. Для этого был использован метод **Toast.makeText().**

Изображение выглядит как текст, снимок экрана, программное обеспечение, мультимедиа

Автоматически созданное описание

Рисунок 4 – Вывод сообщения о добавлении.

**Работа с ресурсами:**

Для реализации локализации в приложении используется класс Locale.Выбранный язык сохраняется в SharedPreferences, и приложение перезапускается с учетом новой локали. Метод applyLanguage устанавливает новую локаль в конфигурации, что позволяет приложению использовать соответствующие ресурсы для выбранного языка. Для корректной работы с датами и текстовыми ресурсами учитывается текущая локаль. Обновление меток дней недели и текста в интерфейсе происходит с учетом текущей локали.

Меню используется для предоставления пользователю возможности выбора языка приложения. Оно предоставляет пользователю выбор языка: русский или английский.Метод **onOptionsItemSelected** обрабатывает выбор пользователя в меню, изменяя язык приложения и обновляя соответствующие элементы интерфейса.

Начало формы

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 5 – Добавление меню.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 6 – Локализация главной страницы.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 7 – Локализация меню.

Изображение выглядит как снимок экрана, программное обеспечение, компьютер, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 8 – Локализация компонента добавления и редактирования.

**Адаптеры в Андроид:**

Класс **TaskAdapter** служит для связи данных из списка задач (**List<Task> tasks**) с пользовательским интерфейсом, представленным в виде макета **simple\_list\_item\_2.xml**. Он расширяет базовый класс **ArrayAdapter**, который предоставляет стандартные методы для работы с данными и отображения их в элементах пользовательского интерфейса.

1. **Конструктор:**
   * Принимает контекст и список задач.
   * Инициализирует список задач и объект **SharedPreferences** для работы с состоянием чекбоксов.
2. **Метод getView:**
   * Каждый элемент списка отображается в соответствующем макете **simple\_list\_item\_2.xml**.
   * Использует паттерн "ViewHolder" для более эффективного управления элементами интерфейса.
   * Загружает данные из задачи в соответствующие элементы интерфейса (чекбокс, текстовые поля, иконки).
   * Устанавливает слушателей для чекбокса (для обновления состояния в базе данных) и иконки редактирования (для открытия активности редактирования).
3. **Метод openEditActivity:**
   * Создает и отправляет намерение для открытия активности редактирования задачи.
4. **Методы getTasks и sortTasksByTime:**
   * Предоставляют доступ к списку задач и сортируют задачи по времени.

**Разметка simple\_list\_item\_2.xml:**

* Верхний **LinearLayout** содержит горизонтальный контейнер с чекбоксом, названием и иконкой редактирования.
* Ниже идет **TextView** для отображения описания задачи.
* В самом низу расположено горизонтальное размещение для отображения времени и иконки удаления.

**Как это работает:**

1. **Отображение данных:**
   * Когда элемент списка становится видимым, вызывается метод **getView**, который заполняет соответствующий макет данными из объекта **Task**.
   * Данные отображаются в соответствующих элементах пользовательского интерфейса.
2. **Взаимодействие с пользователями:**
   * При нажатии на чекбокс вызывается слушатель, который обновляет состояние задачи в базе данных (**isChecked**).
   * При нажатии на иконку удаления вызывается слушатель, который удаляет задачу из базы данных и обновляет список.
3. **Редактирование задачи:**
   * При нажатии на иконку редактирования вызывается слушатель, который открывает активность редактирования с передачей идентификатора задачи.
4. **Сортировка задач:**
   * Задачи могут быть отсортированы по времени.
5. **MainActivity:**

* В методе **displayTasks** создается экземпляр **TaskAdapter** и устанавливается для **tasksListView**. Это позволяет отобразить список задач для выбранной даты.

Класс **TaskAdapter** играет ключевую роль в связывании данных и их отображении в пользовательском интерфейсе. Он также обеспечивает взаимодействие с базой данных для обновления и получения данных о задачах.

Начало формы

**5) Намерения в Android. В этом задании должны быть спроектированы 3-4 Activity для своего приложения.**

Описание использования активити:

**MainActivity:**

1. **Добавление задачи:**
   * Открывает **AddActivity** при нажатии кнопки "Добавить".
   * Передает управление в **AddActivity** для добавления новой задачи.
2. **Редактирование задачи:**
   * Обрабатывает результат из **EditActivity** для обновления списка задач при редактировании задачи.

**TaskAdapter:**

1. **Действия удаления и редактирования:**
   * Предоставляет возможность удалять задачи.
   * Открывает **EditActivity**, когда щелкается значок редактирования.

**EditActivity:**

1. **Загрузка данных о задаче:**
   * Загружает данные о задаче для редактирования с использованием идентификатора задачи.
   * Заполняет поля интерфейса пользователя существующими данными о задаче.
2. **Обновление задачи:**
   * Обновляет задачу в базе данных новыми данными.
   * Возвращает результат в **MainActivity** для обновления списка задач.

**AddActivity:**

1. **Добавление новой задачи:**
   * Обрабатывает добавление новой задачи в базу данных.
   * Показывает всплывающее сообщение в зависимости от успеха операции.
   * Переходит обратно на главную страницу.

**6)Фрагменты.**

Программа использует фрагменты для создания модульной структуры интерфейса. Основная активность (**MainActivity**) содержит фрагмент (**FragmentList**), который отображает список задач в **ListView**. Фрагмент добавляется динамически в методе **onCreate** активности. Когда задачи обновляются, вызывается метод фрагмента (**updateTasks**), который обновляет содержимое **ListView** новым списком задач.

Фрагмент **FragmentList** содержит **ListView** и отвечает за отображение списка задач. Он используется для разделения логики отображения задач и управления интерфейсом от основной логики активности. Кроме того, макеты фрагментов (**list\_fragment.xml**) и активности (**activity\_main.xml**) разделены для более четкой структуры кода.

**7)Хранилища данных. Работа с базой данных SQLite. Дополнить свою задачу базой данных.**

1. **Создание базы данных:**
   * Определен класс **Database**, унаследованный от **SQLiteOpenHelper**.
   * В методе **onCreate** создается таблица "tasks" с полями "id", "title", "description", "date", "time" и "isChecked".
   * В случае изменения версии базы данных (метод **onUpgrade**), происходит пересоздание таблицы.
2. **Операции CRUD (Create, Read, Update, Delete):**
   * **Добавление новой задачи (addTask):** Вставляет новую задачу в таблицу и возвращает идентификатор новой строки.
   * **Удаление задачи (deleteTask):** Удаляет задачу из таблицы по идентификатору.
   * **Получение списка задач по дате (getTasks):** Возвращает список задач для конкретной даты.
   * **Обновление задачи (updateTask):** Обновляет данные задачи в таблице по идентификатору.
   * **Получение задачи по идентификатору (getTaskById):** Возвращает объект задачи по идентификатору.
3. **Использование Cursor:**
   * При выборке данных используется **Cursor**, который позволяет итерироваться по результатам запроса.
4. **Поток выполнения:**
   * Операции с базой данных выполняются в фоновом потоке (**getWritableDatabase**, **getReadableDatabase**).

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 9 – Диаграмма бд.

**Основная функциональность и дополнительные возможности приложения:**

1. **Сортировка задач по времени:**
   * Реализован метод **sortTasksByTime()**, который использует **Collections.sort()** для сортировки списка задач по времени. Сравнение времени задач выполняется с использованием **SimpleDateFormat**.
2. **Переключение дат:**
   * Имеются кнопки и текстовые поля, которые обрабатывают нажатия и обновляют выбранную дату. Дни могут быть переключены вперед и назад.
3. **Перевод всех дат:**
   * Даты отображаются с использованием локали, что позволяет перевести текстовое представление даты на разные языки. Используется **SimpleDateFormat** с указанием локали.
4. **Обновление и изменение сразу:**
   * После изменения данных, таких как добавление новой задачи или изменение даты, вызывается метод **displayTasks()**, который обновляет отображение задач.
5. **Работа с CheckBoxes:**
   * В адаптере задач (**TaskAdapter**), **CheckBox** связан с состоянием задачи. Изменения сохраняются в базе данных.
6. **Запоминание языка:**
   * Язык приложения сохраняется в **SharedPreferences**. При изменении языка происходит его применение, и приложение перезапускается с новым языком.
7. **Обновление интерфейса:**
   * Метод **updateDateLabels()** обновляет текстовые метки, отображая текущую дату и дни "вчера" и "завтра" с учетом выбранной локали и времени.

Эти функции обеспечивают полноценное управление задачами, а также улучшают пользовательский опыт за счет сортировки, переключения дат и поддержки различных языков.

**Вывод**

Приложение планировщика дел предоставляет удобное и эффективное средство для организации задач и планов на каждый день. Оно отличается простым и интуитивно понятным интерфейсом, поддержкой сортировки задач по времени, мультиязычной поддержкой и гибкостью в редактировании задач. Система сохранения настроек и оперативное обновление интерфейса при внесении изменений обеспечивают удобство и персонализированный опыт пользователей.

Начало формы