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 №1–7**

**Дисциплина: Программирование для мобильных платформ**

Работу выполнила: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Д. Н. Баева

Направление подготовки: 02.03.02 Фундаментальная информатика и информационные технологии

Преподаватель: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Т. А. Приходько

**Цель работы:** изучение базового устройства платформы Android и возможностей, которые предоставляется данная платформа для разработки мобильных систем, получение практических навыков по созданию пользовательских интерфейсов, сервисов, а также по использованию сигнализации, аппаратных сенсоров и стандартных хранилищ информации в рамках платформы.

**Ход работы:**

Для разработки было выбрано приложение студенческое расписание для студента группы 46/2 на 7 семестр 4 курса.

Согласно заданию лабораторной работы №1А в Android Studio был создан проект, имеющий следующий структуру, представленную на рисунке 1:

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

Автоматически созданное описание

Рисунок 1 – Структура разработанного проекта.

Файл Android Manifest, имеет следующий вид (рисунок 2).

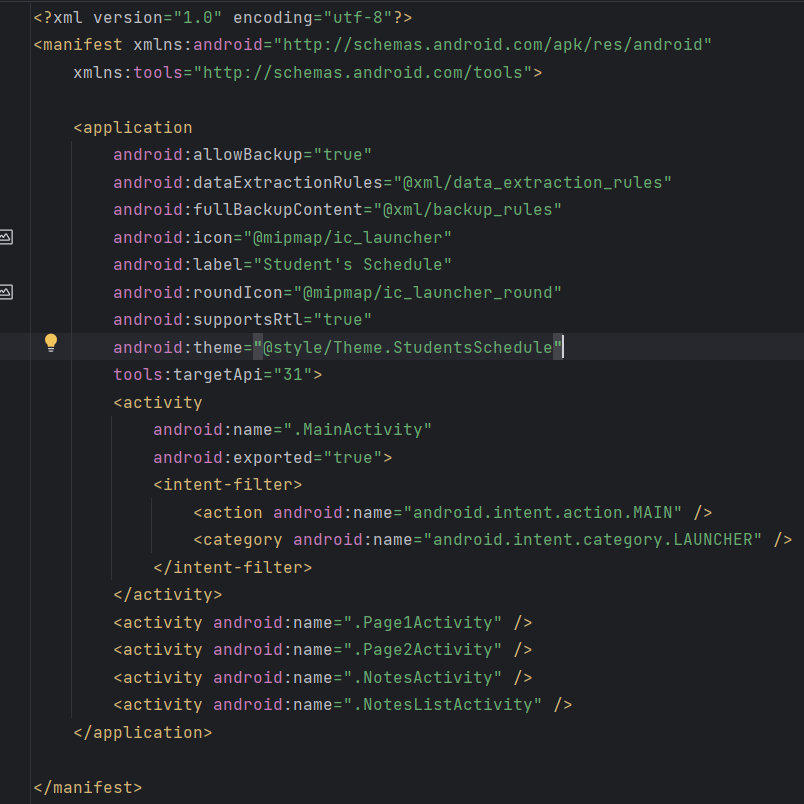


Рисунок 2 – Манифест приложения.

В приложении используется 4 Android-Activity: MainActivity – главная страничка приложения, которая была разработана с помощью фрагментов, Page1Activity – активность, которая отображает расписание по числителю, Page2Activity – активность, которая отображает расписание по знаменателю и NotesActivity – активность, которая отображает учебные заметки.

Согласно заданиям лабораторной работы №1А и лабораторной работы 2А рассмотрим по отдельности каждую из активности.

Класс Page1Activity является наследником класса AppCompatActivity и представляет собой активность, которая отображает расписание занятий студентов. В этом классе определены методы для создания и управления визуальными элементами, такими как TextView, которые представляют дни недели. Метод onCreate() вызывается при создании экземпляра класса. В этом методе происходит инициализация активности, установка макета, который содержит TextView для каждого дня недели, и установка обработчиков событий для каждого TextView. Обработчики событий для каждого TextView определены внутри метода onCreate(). Каждый обработчик событий устанавливает действие, которое должно выполняться при нажатии на соответствующий TextView. Например, обработчик событий для mondayTextView выводит список предметов на понедельник, а обработчик событий для tuesdayTextView выводит список предметов на вторник. Каждый обработчик событий устанавливает действие, которое должно выполняться при нажатии на соответствующий TextView. Когда пользователь нажимает на TextView, вызывается метод onClick(), который содержит код для вывода списка предметов на соответствующий день недели. Для отображения сообщения на экране используется класс Toast. Метод makeText() класса Toast создает новый объект Toast и устанавливает текст сообщения и продолжительность показа. Затем метод show() отображает сообщение на экране. Это XML-файл page\_1, который определяет разметку для активности Page1Activity, содержит TableLayout, который используется для создания таблицы с расписанием занятий студентов. Каждый TextView имеет уникальный идентификатор, который можно использовать для получения ссылки на него в коде Java.

Аналогичным образом работает класс Page2Activity с соответствующей XML-разметкой page\_2 только визуальные элементы TextView заполнены другими значениями которые соответствуют расписанию по знаменателю. В них аналогично, для оповещений используется класс Toast.

На рисунках 3–6 будут представлены фрагменты соответствующих классов и разметок.

Изображение выглядит как текст, снимок экрана

Автоматически созданное описание

Рисунок 3 – Фрагмент класса Page1Activity.

Изображение выглядит как текст, снимок экрана

Автоматически созданное описание

Рисунок 4 – Фрагмент класса Page2Activity.

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

Автоматически созданное описание

Рисунок 5 – Фрагмент разметки для Page1Activity (page\_1.xml).

Изображение выглядит как текст, снимок экрана

Автоматически созданное описание

Рисунок 6 – Фрагмент разметки для Page2Activity (page\_2.xml).

Другим образом, организован класс и соответствующая разметка для MainActivity. Здесь использовались фрагменты в Android. XML-разметка activity\_main содержит LinearLayout, который используется для создания вертикального списка элементов пользовательского интерфейса. LinearLayout имеет следующие атрибуты: android:layout\_width и android:layout\_height определяют ширину и высоту, android:background определяет цвет фона и android:orientation определяет ориентацию элементов в LinearLayout. Также используется Toolbar — это пользовательский интерфейс, который содержит элементы управления, такие как кнопки и текстовые поля. В разметке используются два фрагмента. Fragment - это компонент пользовательского интерфейса, который может использоваться повторно и многократно. Он определяется с помощью fragment. Атрибуты android:id, android:name, android:layout\_width и android:layout\_height определяют идентификатор, имя, ширину и высоту Fragment. В первом фрагменте содержится приветственное сообщение на главном экране приложения, а во втором – кнопки и текстовые надписи перед ними (рисунок 7,8).

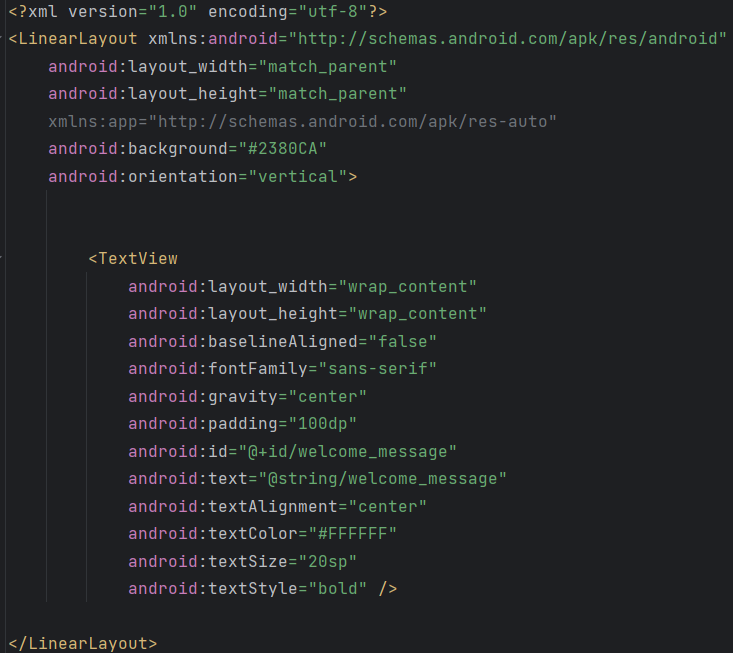


Рисунок 7 – Фрагмент fragment\_1.

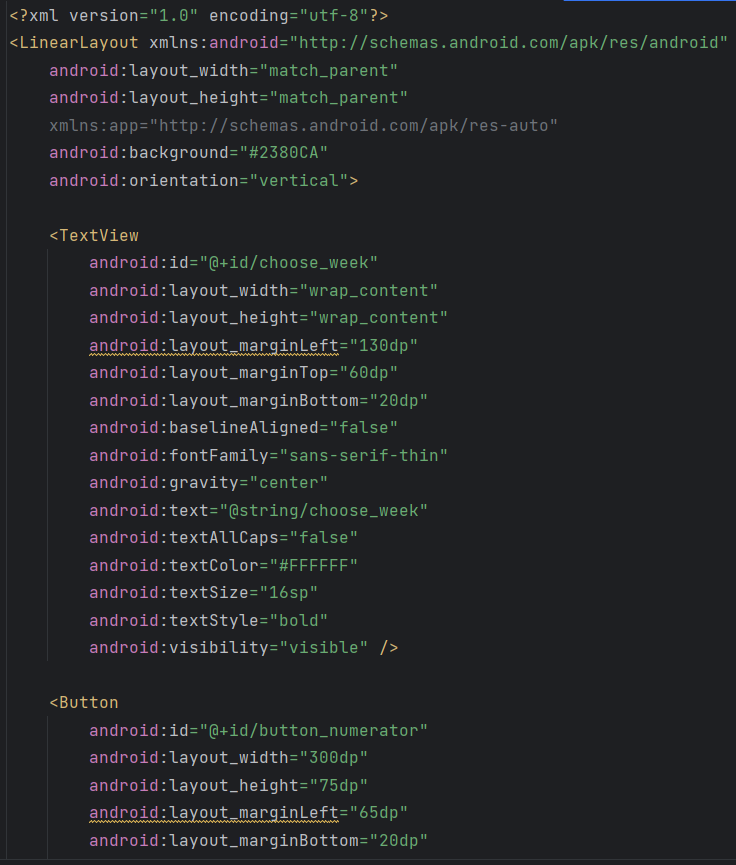


Рисунок 8 – Фрагмент fragment\_2.

Для получившейся разметке был создан пакет под названием Fragment, который содержит 2 соответствующих класса – Fragment1 и Fragment2.

Оба класса является наследниками класса Fragment. Fragment2 представляет собой фрагмент, который содержит три кнопки: buttonNumerator, buttonDenominator и buttonNotes. Каждая кнопка имеет свой обработчик событий, который определяет действие, которое должно выполняться при нажатии на соответствующую кнопку. Метод onCreateView() вызывается при создании экземпляра класса. В этом методе происходит инициализация фрагмента, установка макета, который содержит три кнопки, и установка обработчиков событий для каждой кнопки. Обработчики событий для каждой кнопки определены внутри метода onCreateView(). Каждый обработчик событий устанавливает действие, которое должно выполняться при нажатии на соответствующую кнопку. Например, обработчик событий для buttonNumerator запускает активность Page1Activity, а обработчик событий для buttonDenominator запускает активность Page2Activity. А в классе Fragment1 метод onCreateView() инициализирует фрагмент, устанавливает макет и возвращает корневой элемент макета. На рисунках 9–10 представлены кусочки кодов получившихся классов.

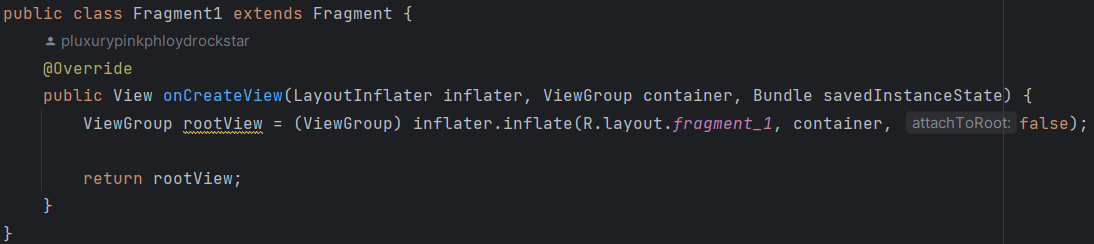


Рисунок 9 – Класс Fragment1.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 10 – Класс Fragment2.

Последней является активность по добавлении учебных заметок. При её реализации использовалась работа с БД и работа с адаптерами (лабораторные работы №4А и №7А).

Сперва был написан класс для создания БД. Класс NoteDataBase является наследником класса SQLiteOpenHelper и представляет собой базу данных SQLite, которая содержит таблицу notes\_data\_base. Конструктор NoteDataBase вызывается при создании экземпляра класса. В этом конструкторе происходит инициализация базы данных, установка имени базы данных, версии базы данных и других параметров. Метод onCreate() вызывается при создании базы данных. В нём происходит создание таблицы notes\_data\_base в базе данных. Метод onUpgrade() вызывается при обновлении базы данных. В этом методе происходит удаление старой таблицы `notes` и создание новой таблицы. Метод addTask() выполняет вставку данных в таблицу. Cначала происходит получение ссылки на базу данных, создание объекта ContentValues, который содержит данные, которые необходимо вставить в таблицу, и выполнение вставки данных в таблицу. Метод deleteTask() выполняет удаление данных из таблицы по идентификатору. Метод editTask() выполняет изменение данных в таблице. В этом методе происходит получение ссылки на базу данных, создание объекта ContentValues, который содержит данные, которые необходимо изменить в таблице, и выполнение изменения данных в таблице. Последний метод getTasks() возвращает список всех заметок из таблицы notes. В нём происходит получение ссылки на базу данных, выполнение запроса на выборку всех данных из таблицы notes, создание объекта Task для каждой записи в таблице и добавление объекта Task в список taskList. Фрагмент кода этого класса представлен на рисунке 11.

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Рисунок 11 – Фрагмент кода класса NoteDataBase.

В качестве модели данных, которая используется для хранения информации о заметках в базе данных, использовался класс Task. Он содержит три поля: mId, mSubject и mNote. Поле mId содержит уникальный идентификатор заметки. Поля mSubject и mNote содержат соответственно тему и текст заметки. Конструктор Task() создает новый объект Task с пустыми полями.Конструктор Task(int id, String subject, String note) создает новый объект Task с заданными значениями полей. Методы getId(), getSubject() и getNote() возвращают значения соответствующих полей. Методы setId(int id), setSubject(String subject) и setNote(String note) устанавливают значения соответствующих полей. Фрагмент кода этого класса представлен на рисунке 12.

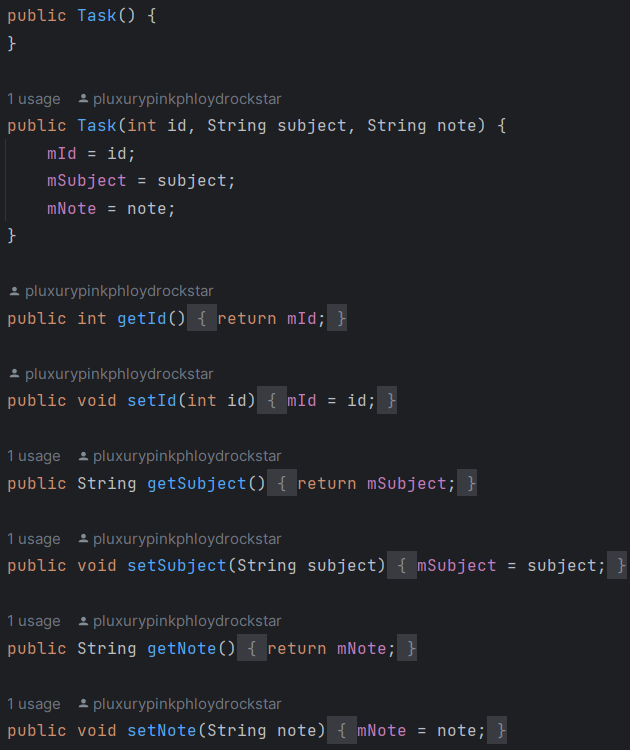


Рисунок 12 – Фрагмент кода класса Task.

Класс NotesAdapter является наследником класса BaseAdapter и представляет собой адаптер, который используется для отображения списка заметок в пользовательском интерфейсе. В этом классе определены методы для создания и управления адаптером. Конструктор NotesAdapter создает новый объект NotesAdapter с заданными значениями полей. Метод getCount() возвращает количество элементов в списке. Метод getItem(int position) возвращает элемент списка по указанной позиции. Метод getItemId(int position) возвращает идентификатор элемента списка по указанной позиции. Метод getView(int position, View convertView, ViewGroup parent) возвращает представление элемента списка по указанной позиции. Если представление уже существует, оно переиспользуется. В противном случае создается новое представление. Фрагмент кода этого класса представлен на рисунке 13.

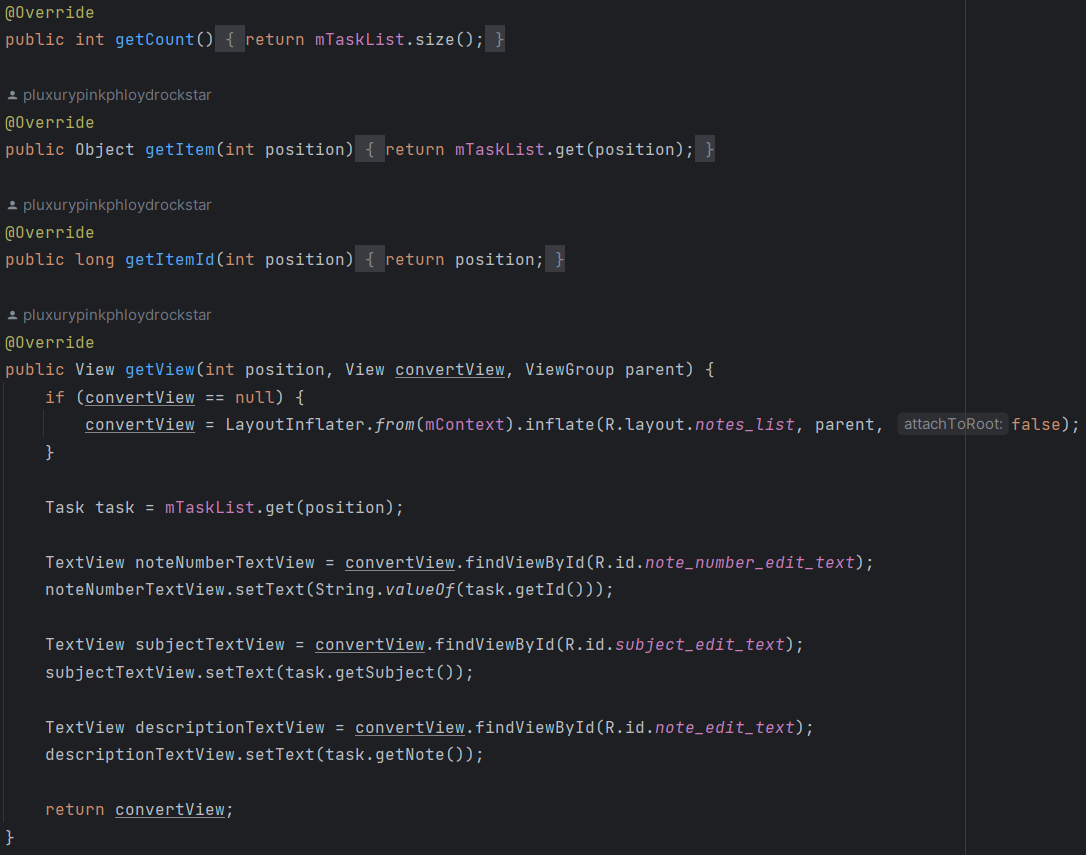


Рисунок 13 – Фрагмент кода класса-адаптера NotesAdapter.

Наконец, последним является код, который определяет активность NotesListActivity, которая отображает список заметок. В этом коде используется класс NoteDataBase для получения списка заметок из базы данных, класс NotesAdapter для отображения списка заметок в пользовательском интерфейсе и класс ListView для отображения списка заметок. В нем определены 3 кнопки – добавление, удаление или изменение записи. Для каждой кнопки устанавливается слушатель нажатий. При нажатии на любую кнопку, сперва данные из полей ввода считываются и проверяются на пустоту. Если хотя бы одно из полей пустое, то выводится сообщение об ошибке. В противном случае выполняется добавление заметок, удаление заметки по номеру или изменение заметки. После этого поля ввода очищаются и выводится сообщение об успешном выполнении той или иной манипуляции с данными. Фрагмент кода класса, на котором демонстрируются кнопки, представлен на рисунке 14.

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Рисунок 14 – Фрагмент кода класс NotesActivity c определением и работой кнопок.

Разметка page\_note.xml состоит из трех объектов EditView, которые соответствуют номеру заметки, названия предмету и тексту самой заметки. После определены разметка 3 кнопок для манипуляций с заметками. Для отображения списка заметок используется ListView. Верстка LinearLayout. Фрагмент разметки представлен на рисунке 15.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 15 – Фрагмент разметки активности NotesActivity.

Согласно заданию лабораторной работы №3А 3 в приложение было добавлено меню, с помощью которого выполнялось изменение локализации рус/англ. Для этого сначала в папке res было создана папка menu, в которой была написана разметка для меню, отображающегося на главном окне приложения (рисунок 16).

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 16 – Разметка меню приложения.

Далее были определены альтернативные ресурсы. В проекте были созданы две папки value-ru и value-en, соответствующие русской и английской локализации. В файле string.xml были определены Default Value для надписей главной страницы приложения. На рисунке 17 представлена её русская и английская локализация.

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

Автоматически созданное описание

Рисунок 17 – Локализация.

Обработка локализации происходит в классе MainActivity. В методе onCreateOptionsMenu создается меню, а в методе onOptionsItemSelected обрабатываются действия, которые пользователь выбирает в меню. Если пользователь выбирает русский язык, то вызывается метод setLocale с параметром “ru”, который устанавливает русский язык для приложения. Если пользователь выбирает английский язык, то вызывается метод setLocale с параметром “en”, который устанавливает английский язык для приложения. Метод applyLanguage устанавливает язык приложения в соответствии с сохраненным языком. Метод getAppLanguage возвращает сохраненный язык приложения. Метод setLocale устанавливает язык приложения в соответствии с переданным параметром, сохраняет язык в настройках приложения и перезапускает главную активность приложения. Фрагмент кода, выполняющий это, представлен на рисунке 18.

Изображение выглядит как текст, снимок экрана

Автоматически созданное описание

Рисунок 18 – Обработка локализации.

На рисунках 19- представлены скриншоты из разработанного приложения.

Изображение выглядит как текст, снимок экрана, программное обеспечение, веб-страница

Автоматически созданное описание

Рисунок 19 – Главное окно программы (русская локализация).

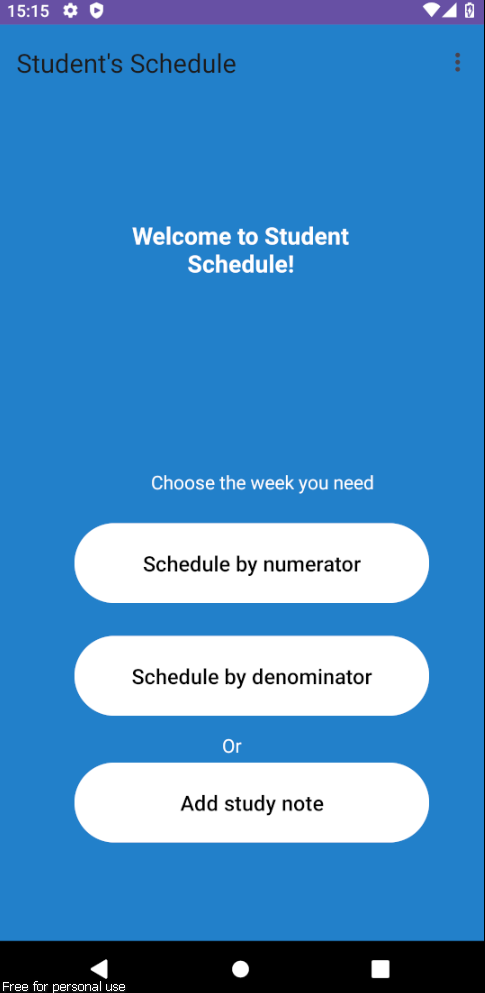


Рисунок 20 – Главное окно программы (английская локализация).

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

Автоматически созданное описание

Рисунок 21 – Окно с расписанием по числителю.

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

Автоматически созданное описание

Рисунок 22 – Окно с расписанием по знаменателю.

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

Автоматически созданное описание

Рисунок 23 – Окно с добавлением учебных заметок.