# Регистрация/авторизация пользователя

Для регистрации и авторизации пользователя мы можем использовать базу данных SQLite или Firebase Authentication.

Изображение выглядит как текст, снимок экрана

Автоматически созданное описание  
Вот подробное описание того, что происходит в коде активности регистрации пользователя:

1. В начале кода объявляются переменные etUsername, etPassword и btnRegister, которые будут использоваться для получения ссылок на соответствующие элементы пользовательского интерфейса (поля для ввода логина и пароля, а также кнопка "Зарегистрироваться").
2. В методе onCreate() вызывается функция setContentView(), которая устанавливает макет для активности регистрации, определенный в файле activity\_register.xml.
3. Затем вызываются методы findViewById(), чтобы найти элементы пользовательского интерфейса по их идентификаторам и присвоить найденные элементы соответствующим переменным.
4. При нажатии на кнопку "Зарегистрироваться" (btnRegister), вызывается обработчик клика, определенный с помощью метода setOnClickListener().
5. Внутри обработчика клика получаются значения из полей ввода логина и пароля (etUsername и etPassword) с помощью метода text.toString().
6. Затем происходит проверка наличия данных в полях ввода. Если оба поля заполнены (т.е. username и password не пустые), то вызывается функция registerUser(username, password), передавая ей значения логина и пароля.
7. Если одно из полей не заполнено, то выводится короткое сообщение (Toast) с просьбой ввести логин и пароль.
8. Метод registerUser(username, password) является заглушкой, где вы можете реализовать логику регистрации пользователя в базе данных (например, с использованием Firebase Authentication или другой библиотеки). В этой функции вы можете вызвать соответствующий метод для регистрации пользователя и обработать результаты операции (например, успешное завершение или возникновение ошибки).  
    **Реализация логики регистрации пользователя (8 пунтк):**

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

1. При открытии активности регистрации RegisterActivity, пользователь видит поля для ввода логина и пароля, а также кнопку "Зарегистрироваться".
2. Когда пользователь нажимает на кнопку "Зарегистрироваться", вызывается обработчик setOnClickListener, который получает значения из полей ввода логина и пароля.
3. Затем происходит проверка, заполнены ли оба поля. Если одно из полей не заполнено, то выводится сообщение об ошибке.
4. Если оба поля заполнены, вызывается функция registerUser(username, password).
5. Внутри функции registerUser(username, password) происходит реализация логики регистрации пользователя. В этом месте вы можете использовать подходящий вам метод регистрации пользователя в базе данных (например, с использованием Firebase Authentication).

Примерно так может выглядеть код для регистрации пользователя с использованием Firebase Authentication.  
В этом примере мы используем createUserWithEmailAndPassword() из Firebase Authentication для создания нового пользователя с указанным логином и паролем. Если регистрация проходит успешно, мы можем получить текущего пользователя с помощью FirebaseAuth.getInstance().currentUser и выполнить необходимые действия после успешной регистрации. Если возникает ошибка при регистрации, мы можем получить информацию об ошибке с помощью task.exception и обработать ее соответствующим образом.

# Отслеживание денежных средств пользователя и связь между счетами

Для отслеживания денежных средств пользователя и связи между счетами мы можем использовать базу данных SQLite или Firebase Realtime Database.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

В приведенном коде определен класс Account, который представляет информацию о счете пользователя. Класс Account имеет три свойства: accountId, accountName и balance.

* accountId - идентификатор счета (тип String).
* accountName - название счета (тип String).
* balance - баланс на счете (тип Double).

# Ручное введение и изменение транзакций

Для ручного ввода и изменения транзакций мы можем использовать базу данных SQLite или Firebase Realtime Database.  
Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Класс Transaction предназначен для хранения информации о транзакции пользователя. Он имеет пять свойств:

* transactionId - идентификатор транзакции (тип String).
* accountId - идентификатор счета, на котором была проведена транзакция (тип String).
* amount - сумма транзакции (тип Double). Если значение отрицательное, то это означает расход, если положительное - доход.
* date - дата и время транзакции (тип Date).
* description - описание транзакции (тип String).

Для пользователя это означает, что он может использовать класс Transaction, чтобы хранить информацию о своих транзакциях. Например, когда пользователь проводит покупки или получает зарплату, он может создавать объекты класса Transaction, чтобы отразить эти операции (Добавление транзакции) или изменить уже созданную (Изменение транзакции)

**Добавление транзакции:**Пользователь может использовать функцию addTransaction(), чтобы добавить новую транзакцию в базу данных. Например, когда пользователь проводит покупки в магазине, он может создать новый объект Transaction с отрицательной суммой (расход), указав идентификатор своего счета, дату и описание. Затем он может вызвать функцию addTransaction(), чтобы сохранить эту транзакцию в базе данных.  
  
**Изменение транзакции:**Пользователь также может использовать функцию updateTransaction(), чтобы изменить существующую транзакцию в базе данных. Например, если пользователь ошибочно указал неправильную сумму для своей покупки, он может вызвать функцию updateTransaction() и указать идентификатор транзакции и новую сумму.

# Категории расходов и доходов

Для того, чтобы расходы и доходы после добавления автоматически распределялись по заданным пользователем категориям, можно использовать следующий подход:  
  
1) Добавить новое свойство category в класс Transaction, которое будет хранить информацию о категории транзакции. Например:  
Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

2) Создать класс Category, который будет представлять категорию расходов или доходов. Класс может содержать следующие свойства:

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

3) Создать список категорий, заданных пользователем. Например:

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

4) После добавления новой транзакции в базу данных, определить ее тип (доход или расход) и категорию на основе описания транзакции. Например, можно использовать словарь с ключами-словами, соответствующими категориям, и значениями-списками слов, связанными с каждой категорией. Затем можно написать функцию, которая будет проверять описание транзакции на наличие слов из словаря и устанавливать соответствующую категорию.

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

В этом примере функция categorizeTransaction() принимает объект Transaction и список категорий в качестве аргументов и возвращает новый объект Transaction с установленной категорией. Функция проверяет тип транзакции и описание транзакции на наличие слов из списка слов каждой категории. Если слово из списка найдено, то устанавливается соответствующая категория. Если категория не найдена, то устанавливается значение "Без категории".