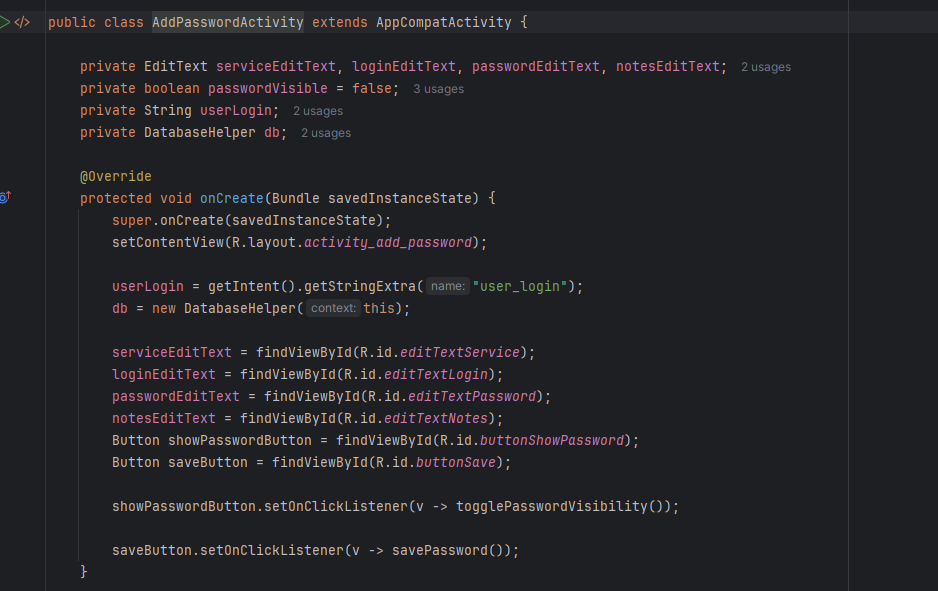
Отчет по аттестационной работе.

Выполнил: Мордовской Алексей.

Создаем класс AddPasswordActivity.

  
Рисунок 1.

public class AddPasswordActivity extends AppCompatActivity — объявление класса активности, наследующей от AppCompatActivity.

private EditText serviceEditText, loginEditText, passwordEditText, notesEditText; — объявление переменных для ввода текста.

private boolean passwordVisible = false; — флаг видимости пароля.

private String userLogin; — переменная для хранения логина пользователя.

private DatabaseHelper db; — объект для работы с базой данных.

onCreate() — метод, запускаемый при создании активности:

setContentView(R.layout.activity\_add\_password); — устанавливает макет активности.

userLogin = getIntent().getStringExtra("user\_login"); — получает логин пользователя из переданных данных.

db = new DatabaseHelper(this); — создаёт помощник для работы с базой данных.

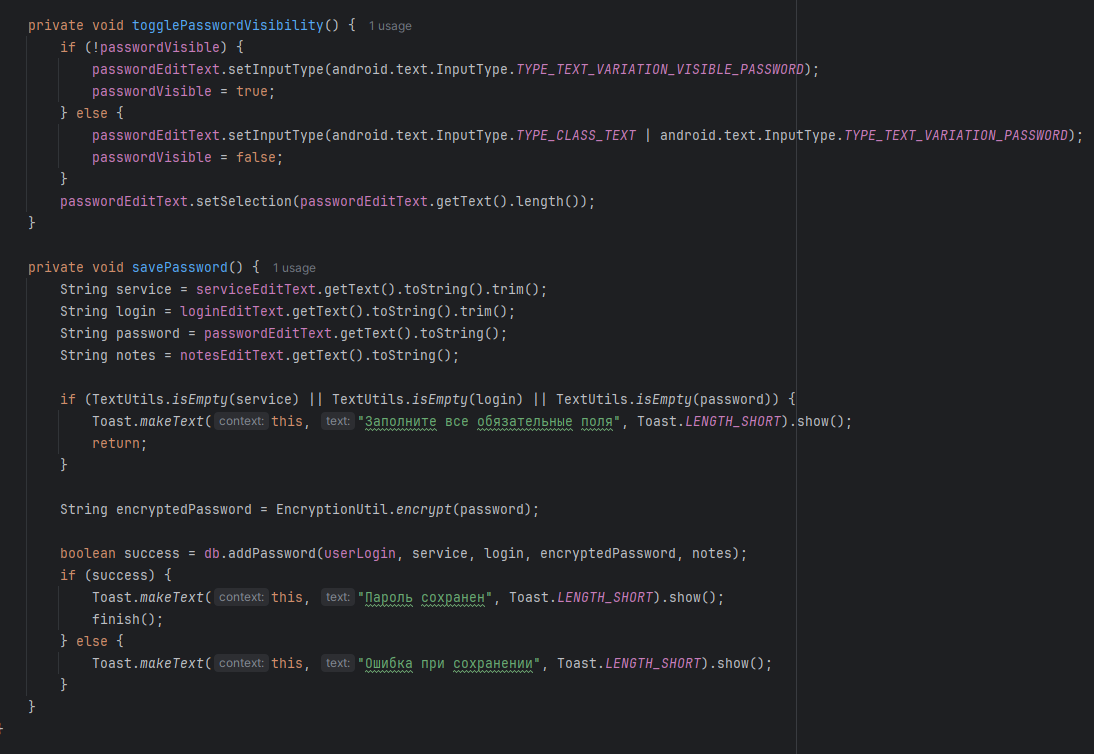
findViewById(R.id.editTextService); и подобные — ищут элементы интерфейса по ID.

Button showPasswordButton = findViewById(R.id.buttonShowPassword); — находит кнопку показа пароля.

Button saveButton = findViewById(R.id.buttonSave); — находит кнопку сохранения.

showPasswordButton.setOnClickListener(v -> togglePasswordVisibility()); — при клике вызывается метод для переключения видимости пароля.

saveButton.setOnClickListener(v -> savePassword()); — при клике вызывается метод для сохранения пароля.

  
Рисунок 2.

Проверяет флаг passwordVisible.

Если пароль скрыт (false), меняет тип ввода InputType на видимый пароль, и устанавливает passwordVisible = true.

Если пароль видимый (true), меняет тип на скрытый (маскируемый), и устанавливает passwordVisible = false.

setSelection() перемещает курсор в конец текста пароля.

savePassword()

Получает текст из полей ввода (service, login, password, notes), удаляет лишние пробелы (trim()).

Проверяет, что обязательные поля (service, login, password) не пустые; если пустые — показывает сообщение и выходит.

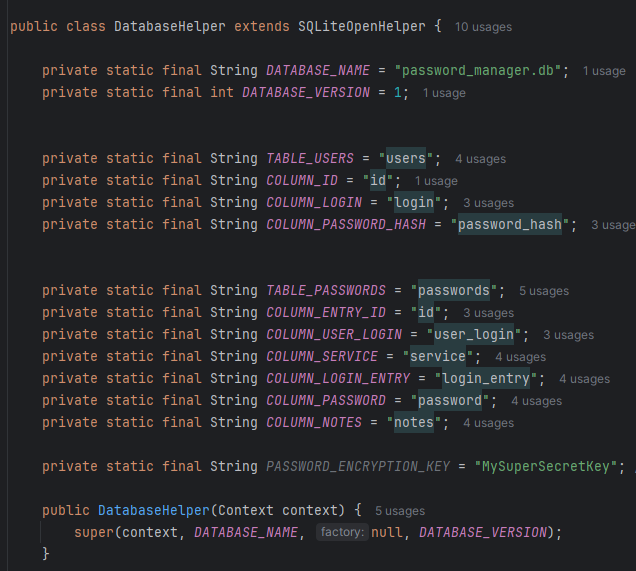
Шифрует пароль с помощью EncryptionUtil.encrypt().

Вызывает db.addPassword() для сохранения данных в базу.

Если сохранение прошло успешно, показывает сообщение и закрывает активность (finish()).

Если ошибка, показывает сообщение об ошибке.

Создаем класс DatabaseHelper.

  
Рисунок 3.

public class DatabaseHelper extends SQLiteOpenHelper — объявление класса, который помогает работать с базой данных SQLite.

private static final String DATABASE\_NAME = "password\_manager.db"; — название базы данных.

private static final int DATABASE\_VERSION = 1; — версия базы данных для управления обновлениями.

private static final String TABLE\_USERS = "users"; — название таблицы пользователей.

private static final String COLUMN\_ID = "id"; — имя столбца для идентификатора пользователя.

private static final String COLUMN\_LOGIN = "login"; — имя столбца для логина пользователя.

private static final String COLUMN\_PASSWORD\_HASH = "password\_hash"; — имя столбца для хранения хеша пароля пользователя.

private static final String TABLE\_PASSWORDS = "passwords"; — название таблицы для хранения паролей.

private static final String COLUMN\_ENTRY\_ID = "id"; — имя столбца для уникального идентификатора записи пароля.

private static final String COLUMN\_USER\_LOGIN = "user\_login"; — имя столбца для логина владельца пароля.

private static final String COLUMN\_SERVICE = "service"; — название сервиса/сайта.

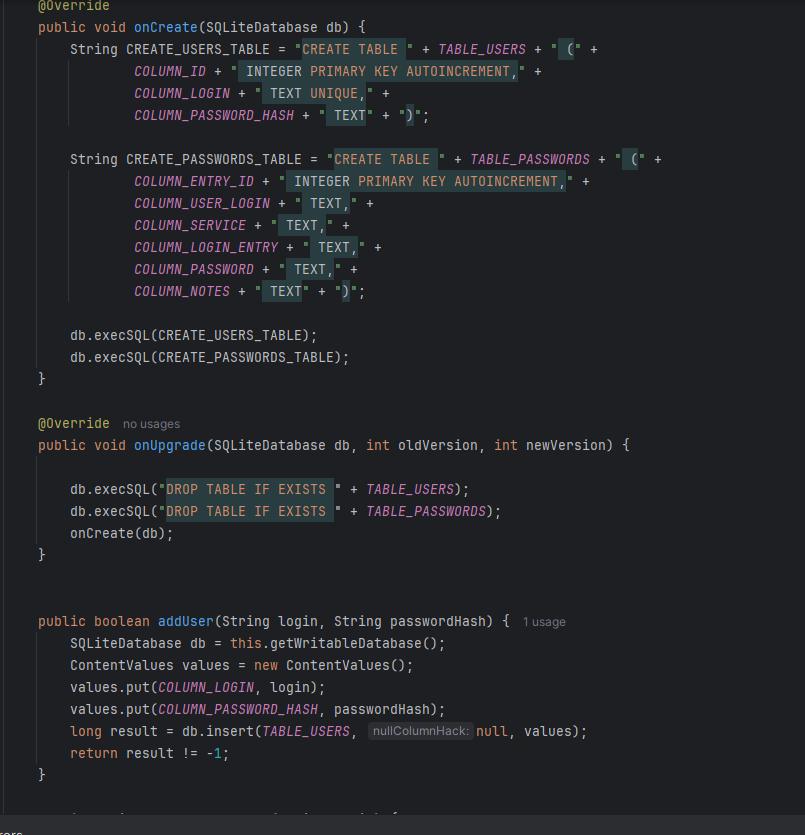
private static final String COLUMN\_LOGIN\_ENTRY = "login\_entry"; — логин для входа в сервис.

private static final String COLUMN\_PASSWORD = "password"; — зашифрованный пароль.

private static final String COLUMN\_NOTES = "notes"; — заметки/примечания.

private static final String PASSWORD\_ENCRYPTION\_KEY = "MySuperSecretKey"; — ключ для шифрования паролей (упрощенно).

public DatabaseHelper(Context context) — конструктор, вызывающий родительский для инициализации базы данных.

  
Рисунок 4.

onCreate(SQLiteDatabase db)

Создаёт таблицы в базе данных:

CREATE TABLE users с полями:

id — автоинкрементный первичный ключ.

login — уникальный логин.

password\_hash — хеш пароля.

CREATE TABLE passwords с полями:

id — автоинкрементный первичный ключ.

user\_login — логин владельца пароля.

service — название сервиса.

login\_entry — логин для сервиса.

password — зашифрованный пароль.

notes — заметки.

db.execSQL() — выполняет SQL-запросы для создания таблиц.

onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion)

Обновляет структуру базы данных:

Удаляет существующие таблицы (DROP TABLE IF EXISTS).

Вызывает onCreate() для пересоздания таблиц.

addUser(String login, String passwordHash)

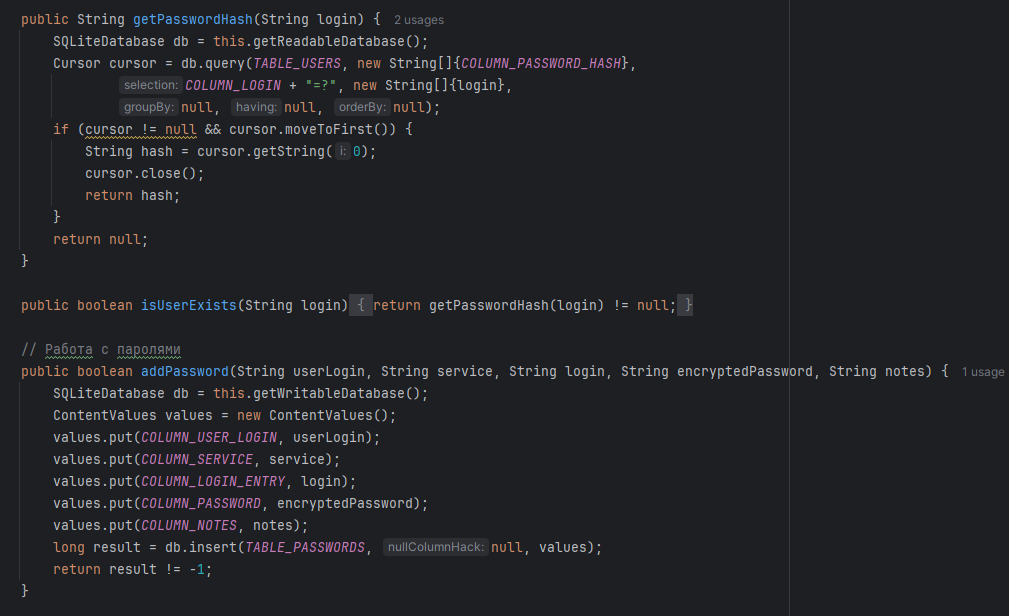
Добавляет нового пользователя:

Получает writable базу данных.

Создаёт ContentValues, вставляя логин и хеш пароля.

Вставляет данные в таблицу users.

Возвращает true, если вставка успешна (result != -1), иначе — false.

  
Рисунок 5.

getPasswordHash(String login)

Получает хеш пароля пользователя по логину:

Открывает чтение базы данных.

Выполняет запрос к таблице users, выбирая поле password\_hash, где login совпадает.

Если есть результат, возвращает его; иначе — возвращает null.

isUserExists(String login)

Проверяет, существует ли пользователь по логину:

Вызывает getPasswordHash(login).

Если результат не null, пользователь есть; иначе — нет.

addPassword(String userLogin, String service, String login, String encryptedPassword, String notes)

Добавляет новую запись пароля:

Открывает запись в базу данных.

Создаёт ContentValues с данными:

Логин владельца (userLogin),

Название сервиса (service),

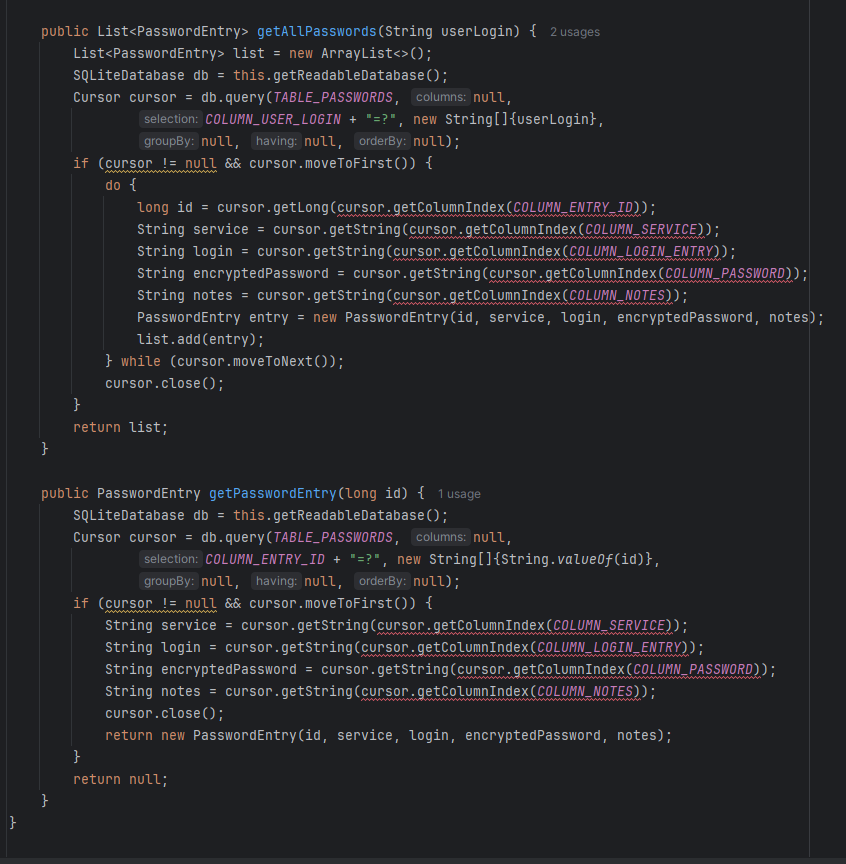
Логин для сервиса (login),

Зашифрованный пароль (encryptedPassword),

Заметки (notes).

Вставляет данные в таблицу passwords.

Возвращает true, если вставка прошла успешно (result != -1), иначе — false.

  
Рисунок 6.

getAllPasswords(String userLogin)

Получает все записи паролей для указанного пользователя:

Создаёт пустой список list.

Открывает чтение базы данных.

Выполняет запрос к таблице passwords, выбирая все записи (null означает все столбцы), где user\_login совпадает.

Если есть результаты, перебирает их:

Извлекает значения полей (id, service, login, encryptedPassword, notes) через getColumnIndex().

Создаёт объект PasswordEntry с этими данными.

Добавляет его в список.

После обхода закрывает курсор и возвращает список.

getPasswordEntry(long id)

Получает конкретную запись пароля по id:

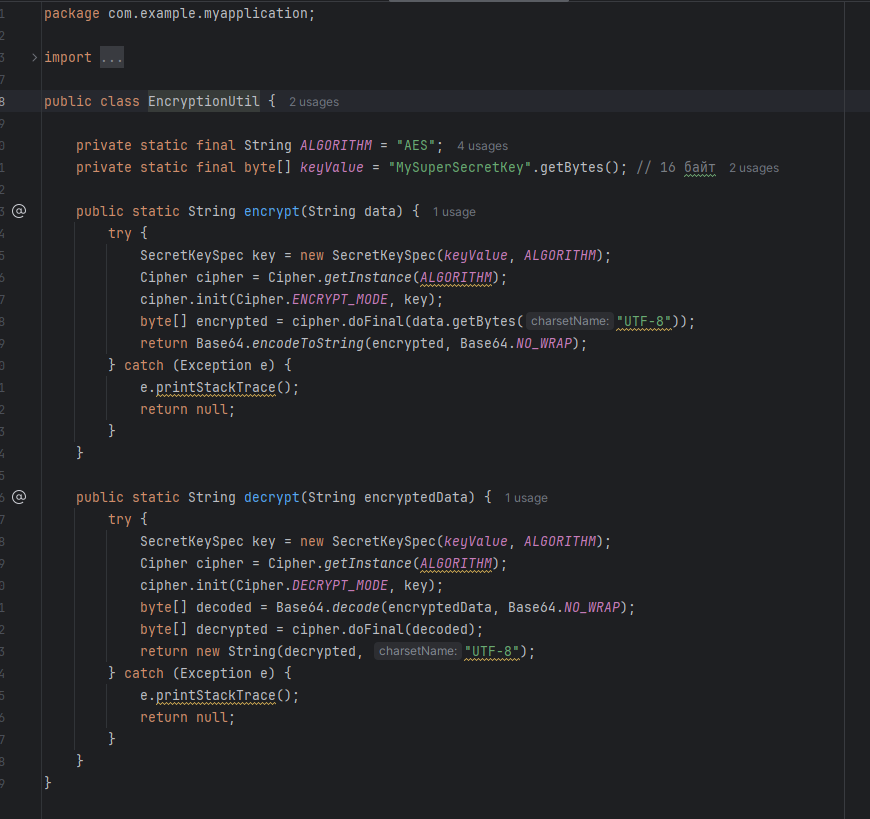
Открывает чтение базы данных.

Выполняет запрос к таблице passwords, выбирая все поля, где entry\_id равно заданному id.

Если есть результат, извлекает поля (service, login, encryptedPassword, notes).

Создаёт и возвращает объект PasswordEntry.

Если запись не найдена, возвращает null.

  
Рисунок 7.

Создаем класс EncryptionUtil.

ALGORITHM: строка "AES" — алгоритм шифрования.

keyValue: байты ключа из строки "MySuperSecretKey" (16 байт).

SecretKeySpec key = new SecretKeySpec(keyValue, ALGORITHM);

Создаёт ключ для шифрации из байтового массива.

Cipher cipher = Cipher.getInstance(ALGORITHM);

Создаёт объект шифра для алгоритма AES.

cipher.init(Cipher.ENCRYPT\_MODE, key);

Инициализирует шифр для шифрования с этим ключом.

byte[] encrypted = cipher.doFinal(data.getBytes("UTF-8"));

Шифрует байтовое представление входных данных.

return Base64.encodeToString(encrypted, Base64.NO\_WRAP);

Кодирует зашифрованные байты в строку Base64 без переносов и возвращает её.

SecretKeySpec key = new SecretKeySpec(keyValue, ALGORITHM);

Создаёт ключ для шифрации из байтового массива.

Cipher cipher = Cipher.getInstance(ALGORITHM);

Создаёт объект шифра для алгоритма AES.

cipher.init(Cipher.ENCRYPT\_MODE, key);

Инициализирует шифр для шифрования с этим ключом.

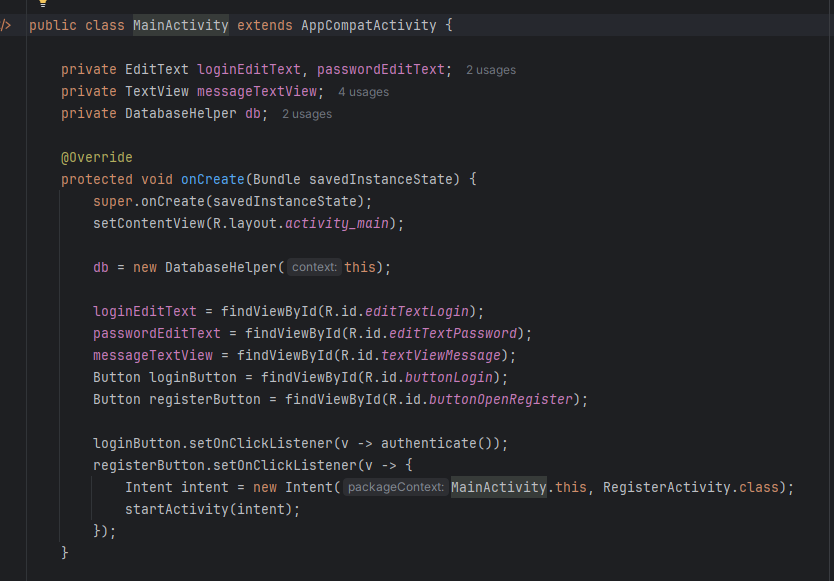
byte[] encrypted = cipher.doFinal(data.getBytes("UTF-8"));

Шифрует байтовое представление входных данных.

return Base64.encodeToString(encrypted, Base64.NO\_WRAP);

Кодирует зашифрованные байты в строку Base64 без переносов и возвращает её.

Прописываем код в MainActivity.

  
Рисунок 8.

loginEditText, passwordEditText: поля для ввода логина и пароля.

messageTextView: текстовое поле для сообщений.

db: объект помощника базы данных DatabaseHelper.

onCreate(Bundle savedInstanceState)

super.onCreate(savedInstanceState);

Вызывает базовую реализацию метода, чтобы подготовить активность.

setContentView(R.layout.activity\_main);

Устанавливает макет активности из файла activity\_main.xml.

db = new DatabaseHelper(this);

Создаёт экземпляр базы данных для работы с данными.

findViewById(R.id.editTextLogin);

Находит и присваивает поле для логина по ID.

findViewById(R.id.editTextPassword);

Находит и присваивает поле для пароля.

findViewById(R.id.textViewMessage);

Находит и присваивает текстовое поле для сообщений.

findViewById(R.id.buttonLogin);

Находит кнопку входа по ID.

findViewById(R.id.buttonOpenRegister);

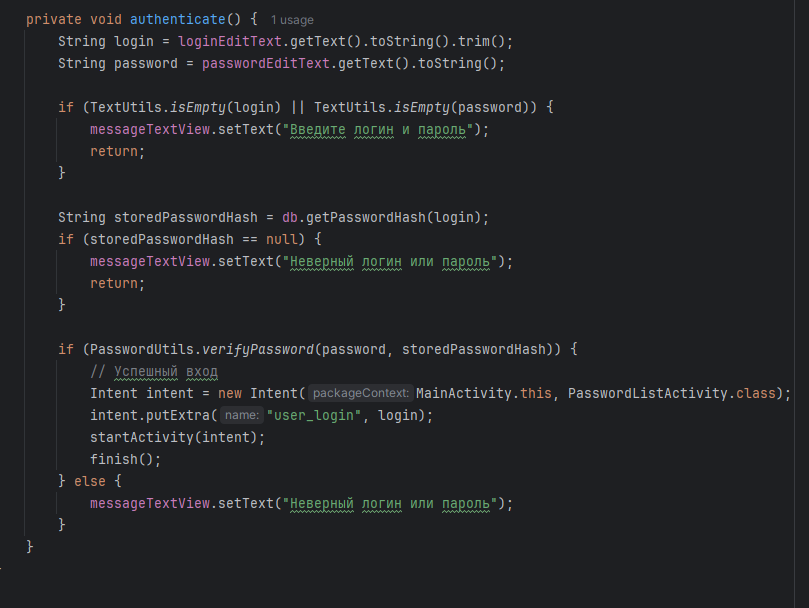
Находит кнопку регистрации.

loginButton.setOnClickListener(v -> authenticate());

Назначает обработчик клика для кнопки входа, вызывающий метод authenticate() (предположительно, для проверки логина и пароля).

registerButton.setOnClickListener(v -> { ... });

Назначает обработчик для кнопки регистрации, который запускает RegisterActivity.

  
Рисунок 9.

String login = loginEditText.getText().toString().trim();

Получает текст из поля логина, удаляет пробелы вначале и в конце.

String password = passwordEditText.getText().toString();

Получает текст из поля пароля.

if (TextUtils.isEmpty(login) || TextUtils.isEmpty(password)) { ... }

Проверяет, пусты ли поля. Если да, выводит сообщение и возвращает.

String storedPasswordHash = db.getPasswordHash(login);

Получает из базы хеш пароля для введённого логина.

if (storedPasswordHash == null) { ... }

Если хеш не найден (логин неправильный), выводит сообщение и возвращает.

if (PasswordUtils.verifyPassword(password, storedPasswordHash)) { ... }

Проверяет, совпадает ли введённый пароль с сохранённым хешем.

Intent intent = new Intent(MainActivity.this, PasswordListActivity.class);

Создаёт намерение для перехода к следующей активности — PasswordListActivity.

intent.putExtra("user\_login", login);

Передаёт логин пользователя в следующую активность.

startActivity(intent);

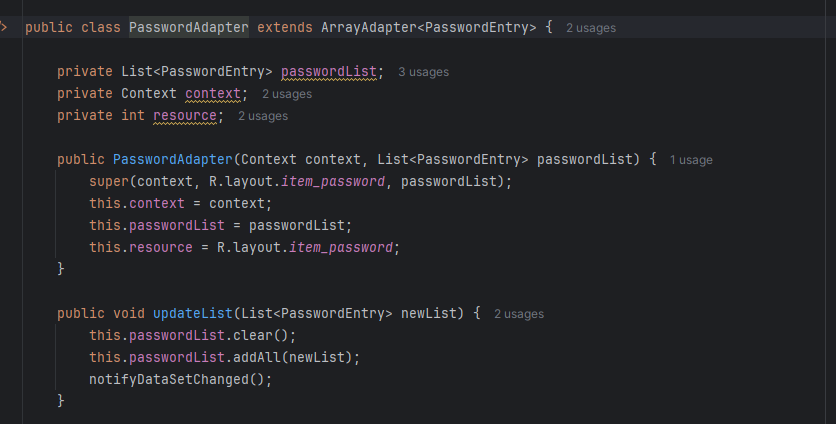
Запускает новую активность.

finish();

Закрывает текущую активность, чтобы пользователь не мог вернуться назад.

В случае несоответствия пароля — выводит сообщение "Неверный логин или пароль".

Создаем класс PasswordAdapter.

  
Рисунок 10.

passwordList: список объектов PasswordEntry, хранящий данные паролей.

context: контекст приложения или активности.

resource: идентификатор макета элемента списка (item\_password).

PasswordAdapter(Context context, List<PasswordEntry> passwordList)

Вызывает конструктор базового класса ArrayAdapter, передавая контекст, макет элемента и список данных.

this.context = context; — сохраняет контекст.

this.passwordList = passwordList; — сохраняет список данных.

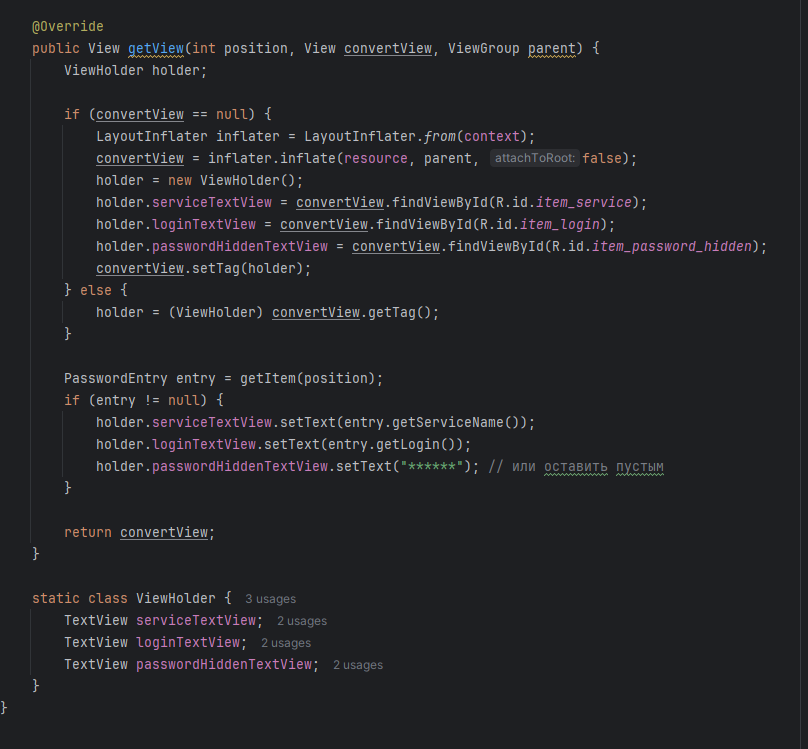
this.resource = R.layout.item\_password; — сохраняет идентификатор макета элемента.

public void updateList(List<PasswordEntry> newList)

Очищает текущий список паролей.

Добавляет новые элементы из newList.

notifyDataSetChanged(); — уведомляет адаптер, что данные изменены, чтобы обновить отображение списка.

  
Рисунок 11.

getView(int position, View convertView, ViewGroup parent)

ViewHolder holder;

Объявление переменной для хранения вспомогательного объекта ViewHolder.

if (convertView == null) { ... }

Если нет пересозданного вида (новый элемент), создаём его.

LayoutInflater inflater = LayoutInflater.from(context);

Получает объект для создания вида из XML.

convertView = inflater.inflate(resource, parent, false);

Создаёт новый вид на основе макета resource.

holder = new ViewHolder();

Создаёт новый объект ViewHolder.

holder.serviceTextView = convertView.findViewById(R.id.item\_service);

Находит и сохраняет ссылку на TextView для сервиса.

holder.loginTextView = convertView.findViewById(R.id.item\_login);

Находит и сохраняет TextView для логина.

holder.passwordHiddenTextView = convertView.findViewById(R.id.item\_password\_hidden);

Находит и сохраняет TextView для скрытого пароля.

convertView.setTag(holder);

Связывает ViewHolder с видом для повторного использования.

else { holder = (ViewHolder) convertView.getTag(); }

Если вид уже создан, извлекает связанный ViewHolder.

PasswordEntry entry = getItem(position);

Получает объект данных для текущей позиции

if (entry != null) { ... }

Если объект существует, заполняет поля:

holder.serviceTextView.setText(entry.getServiceName());

Устанавливает название сервиса.

holder.loginTextView.setText(entry.getLogin());

Устанавливает логин.

holder.passwordHiddenTextView.setText("\*\*\*\*\*\*");

Устанавливает скрытый пароль (может быть пустым).

return convertView;

Возвращает подготовленный вид для отображения.

Создаем класс PasswordDetail.

  
Рисунок 12.

serviceTextView, loginTextView, passwordTextView, notesTextView — ссылки на текстовые элементы интерфейса для отображения данных.

db — объект для работы с базой данных (DatabaseHelper).

entryId — идентификатор записи пароля, переданный через Intent.

super.onCreate(savedInstanceState);

Вызывает метод базового класса для инициализации активити.

setContentView(R.layout.activity\_password\_detail);

Устанавливает макет активити.

findViewById(...)

Получает ссылки на TextView по их ID из макета.

db = new DatabaseHelper(this);

Создаёт объект для работы с базой данных.

entryId = getIntent().getLongExtra("entry\_id", -1);

Получает из Intent ID записи пароля, переданный при запуске активити (если нет — —1).

loadEntry();

Загружает и отображает данные записи.

PasswordEntry entry = db.getPasswordEntry(entryId);

Получает объект PasswordEntry из базы по ID.

if (entry != null) { ... }

Проверяет, существует ли запись.

serviceTextView.setText(entry.getServiceName());

Устанавливает название сервиса.

loginTextView.setText(entry.getLogin());

Устанавливает логин.

String decryptedPassword = EncryptionUtil.decrypt(entry.getEncryptedPassword());

Расшифровывает зашифрованный пароль.

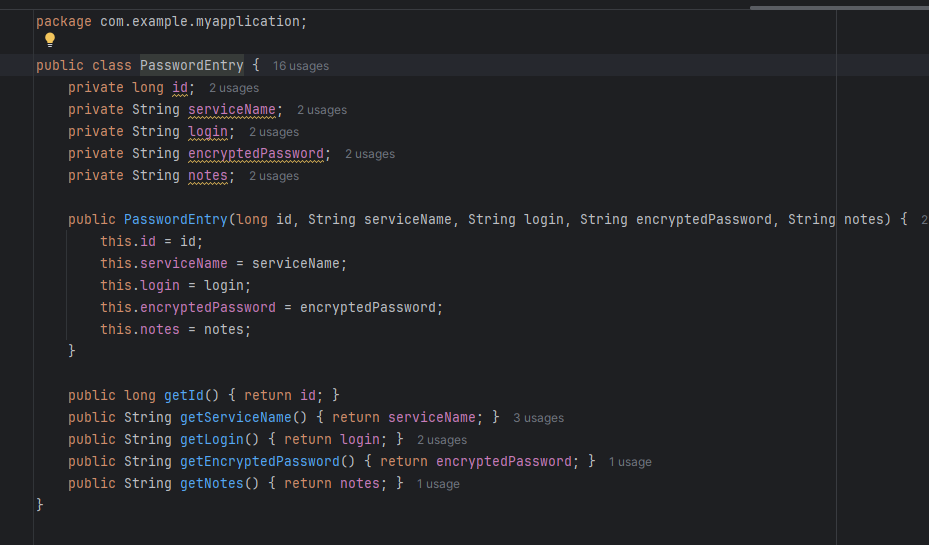
passwordTextView.setText(decryptedPassword);

Отображает расшифрованный пароль.

notesTextView.setText(entry.getNotes());

Отображает заметки.

Создаем класс PasswordEntry.

  
Рисунок 13.

id — уникальный идентификатор записи (тип long).

serviceName — название сервиса или сайта.

login — логин пользователя.

encryptedPassword — зашифрованный пароль.

notes — дополнительные заметки.

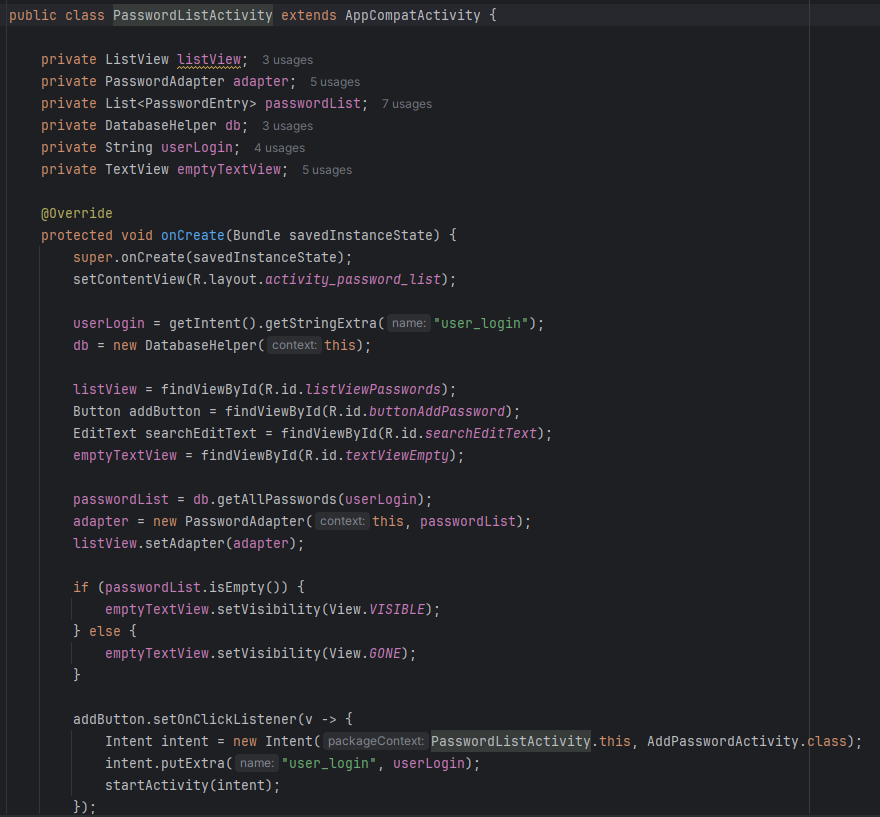
getId() — возвращает id.

getServiceName() — возвращает название сервиса.

getLogin() — возвращает логин.

getEncryptedPassword() — возвращает зашифрованный пароль.

getNotes() — возвращает заметки.

  
Рисунок 14.

listView — компонент для отображения списка паролей.

adapter — адаптер, связывающий список данных с ListView.

passwordList — список объектов PasswordEntry.

db — помощник для работы с базой данных.

userLogin — логин текущего пользователя.

emptyTextView — текстовое сообщение, показываемое, если список пуст.

super.onCreate(savedInstanceState);

Инициализация активности.

setContentView(R.layout.activity\_password\_list);

Установка макета экрана.

userLogin = getIntent().getStringExtra("user\_login");

Получение логина пользователя, переданного через Intent.

db = new DatabaseHelper(this);

Создание объекта для работы с базой данных.

findViewById(...)

Получение ссылок на элементы интерфейса:

listView — список паролей.

addButton — кнопка для добавления новых паролей.

searchEditText — поле поиска (используется, вероятно, для фильтрации).

emptyTextView — сообщение, если список пуст.

passwordList = db.getAllPasswords(userLogin);

Получение всех паролей для текущего пользователя из базы данных.

adapter = new PasswordAdapter(this, passwordList);

Создание адаптера для отображения данных.

listView.setAdapter(adapter);

Установка адаптера в ListView.

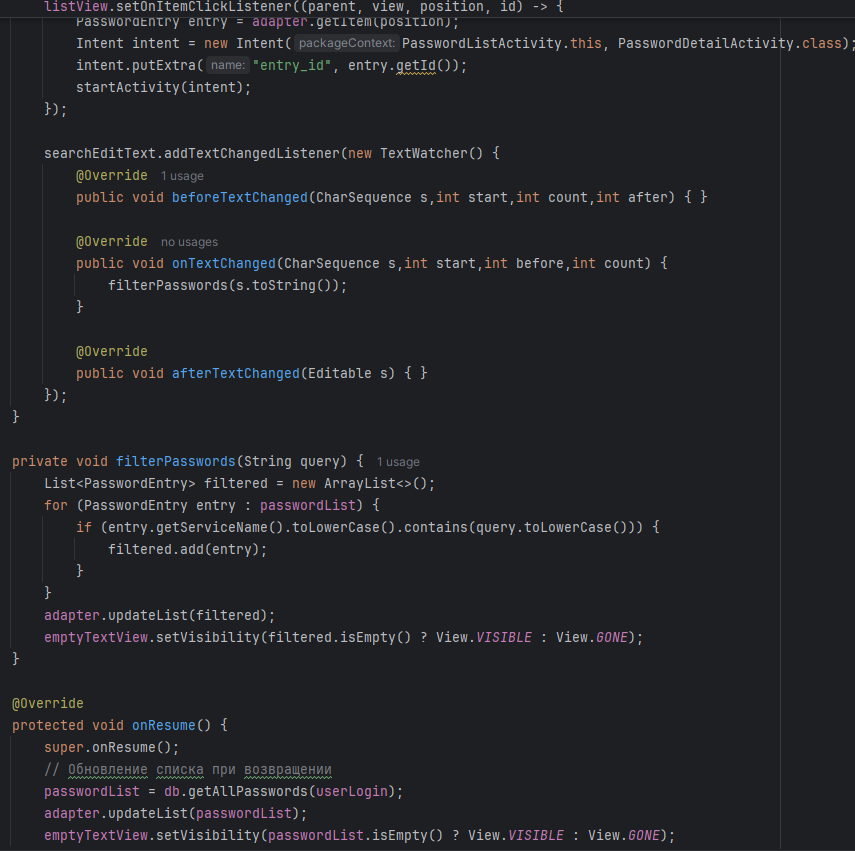
Проверка, пуст ли список:

Если passwordList пуст, показываем emptyTextView.

Иначе скрываем его.

Обработчик нажатия на кнопку:

При нажатии стартует активити AddPasswordActivity, передавая логин пользователя.

  
Рисунок 15.

listView.setOnItemClickListener(...)

Назначает обработчик для кликов по элементам списка.

При клике:

Получает выбранный PasswordEntry по позиции.

Создает Intent для открытия PasswordDetailActivity.

Передает ID выбранной записи через intent.putExtra("entry\_id", entry.getId()).

Запускает активити для отображения деталей пароля.

searchEditText.addTextChangedListener(new TextWatcher() {...})

Добавляет слушатель изменений текста в поле поиска.

Реализует три метода:

beforeTextChanged — вызывается перед изменением текста (пусто).

onTextChanged — вызывается при каждом изменении текста; вызывает метод filterPasswords() с текущим текстом.

afterTextChanged — после изменения текста (пусто).

private void filterPasswords(String query)

Создает новый список filtered.

Проходит по всем паролям (passwordList):

Если название сервиса (ServiceName) содержит введенный запрос (без учета регистра), добавляет запись в filtered.

Обновляет отображаемый список адаптером вызовом adapter.updateList(filtered).

Показывает сообщение emptyTextView, если фильтрованный список пуст.

@Override protected void onResume()

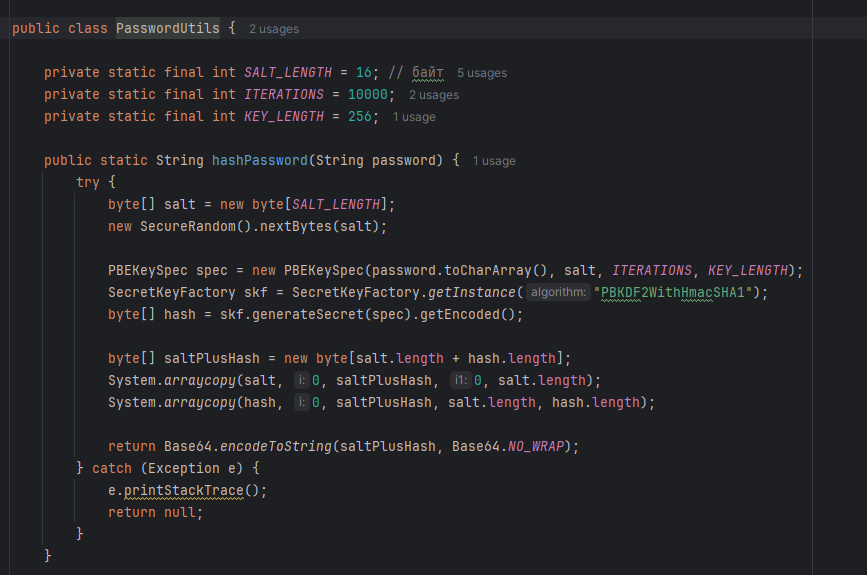
Метод вызывается при возвращении к активности.

Обновляет список паролей из базы данных (db.getAllPasswords(userLogin)).

Обновляет адаптер новым списком (adapter.updateList(passwordList)).

Показывает или скрывает сообщение о пустом списке в зависимости от наличия элементов.

Создаем класс PasswordUtils.

  
Рисунок 16.

SALT\_LENGTH = 16 — длина соли в байтах (16 байт = 128 бит).

ITERATIONS = 10000 — количество итераций для метода PBKDF2 (чем больше — тем сложнее взлом).

KEY\_LENGTH = 256 — длина генерируемого ключа в битах.

Создание соли:

Создается массив байтов salt длиной 16 байт.

Заполняется случайными байтами с помощью SecureRandom.

Генерация хеша:

Создается спецификация PBEKeySpec с паролем, солью, числом итераций и длиной ключа.

Получается SecretKeyFactory для алгоритма PBKDF2WithHmacSHA1.

Генерируется хеш пароля (byte[] hash).

Объединение соли и хеша:

Создается массив saltPlusHash длиной суммы соли и хеша.

Вначале копируется соль, затем хеш.

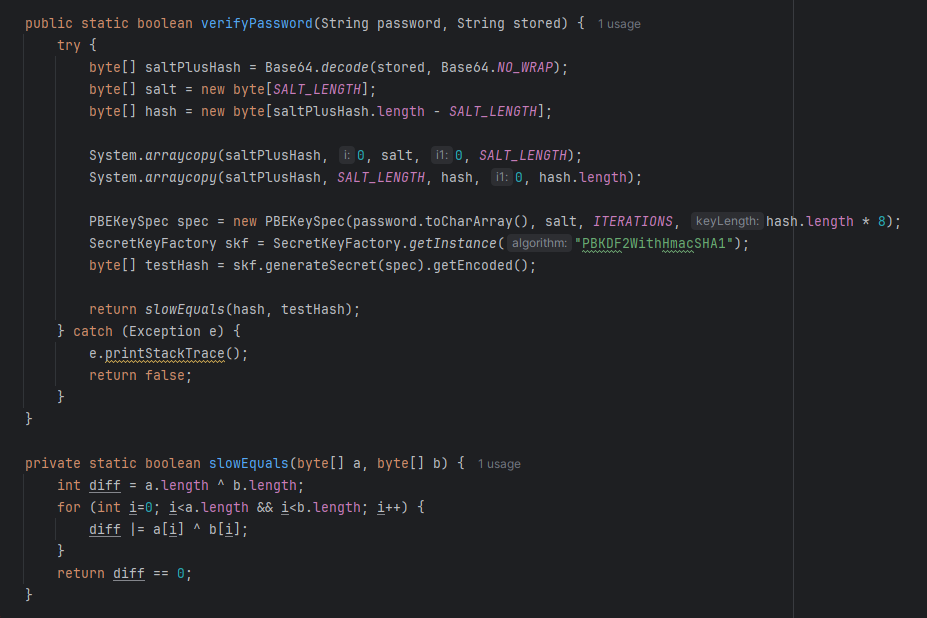
Возвращение результата:

Массив объединенных данных кодируется в строку Base64 без переносов.

Возвращается строка, содержащая соль и хеш, закодированные вместе.

Обработка исключений:

В случае ошибок выводит стек трейса и возвращает null.

  
Рисунок 17.

Метод verifyPassword(String password, String stored)

Декодирование сохраненного пароля:

Раскодирует строку stored из Base64 в байтовый массив saltPlusHash.

Извлечение соли и хеша:

Создает массив salt длиной SALT\_LENGTH.

Создает массив hash длиной оставшейся части saltPlusHash.

Копирует первую часть данных в salt.

Остальную часть — в hash.

Генерация тестового хеша:

Создает PBEKeySpec с введенным паролем, извлеченной солью, количеством итераций и длиной хеша.

Получает SecretKeyFactory для PBKDF2WithHmacSHA1.

Генерирует хеш из введенного пароля (testHash).

Сравнение:

Использует slowEquals для сравнения оригинального хеша (hash) и тестового (testHash).

Возвращает true, если пароли совпадают, иначе — false.

Метод slowEquals(byte[] a, byte[] b)

Безопасное сравнение байтовых массивов:

Инициализирует diff разницей в длинах.

Перебирает байты обоих массивов:

Использует побитовый OR (|=) для сравнения байтов.

Возвращает true, если diff равен 0 (массивы совпадают), иначе — false.

Создаем класс RegisterActivity.

  
Рисунок 18.

public class RegisterActivity extends AppCompatActivity {

Объявление класса активности регистрации, наследует от AppCompatActivity.

private EditText loginEditText, passwordEditText, confirmPasswordEditText;

Объявление переменных для полей ввода логина, пароля и подтверждения пароля.

private TextView messageTextView;

Переменная для текста сообщений пользователю.

private DatabaseHelper db;

Переменная для работы с базой данных.

@Override

Указывает, что метод переопределяет базовый метод.

protected void onCreate(Bundle savedInstanceState) {

Метод, вызываемый при создании активности.

super.onCreate(savedInstanceState);

Вызов базового метода onCreate.

setContentView(R.layout.activity\_register);

Установка макета активности из файла activity\_register.xml.

db = new DatabaseHelper(this);

Создает экземпляр помощника базы данных.

loginEditText = findViewById(R.id.editTextRegisterLogin);

Нахождение элемента для логина по ID.

passwordEditText = findViewById(R.id.editTextRegisterPassword);

Нахождение элемента для пароля.

confirmPasswordEditText = findViewById(R.id.editTextConfirmPassword);

Нахождение элемента для подтверждения пароля.

messageTextView = findViewById(R.id.textViewRegisterMessage);

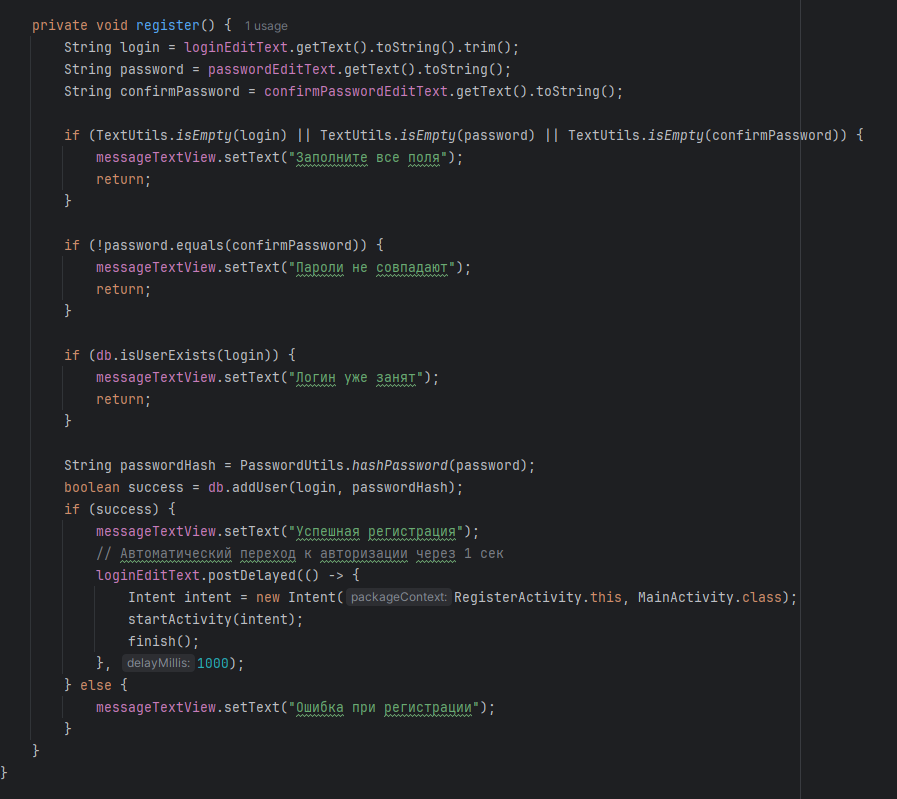
Нахождение элемента для отображения сообщений.

Button registerButton = findViewById(R.id.buttonRegister);

Поиск кнопки регистрации.

registerButton.setOnClickListener(v -> register());

Назначение обработчика нажатия — при клике вызывается метод register().

  
Рисунок 19.

String login = loginEditText.getText().toString().trim();

Получает введённый логин и удаляет пробелы по краям.

String password = passwordEditText.getText().toString();

Получает введённый пароль.

String confirmPassword = confirmPasswordEditText.getText().toString();

Получает подтверждение пароля.

if (TextUtils.isEmpty(login) || TextUtils.isEmpty(password) || TextUtils.isEmpty(confirmPassword)) { ... }

Проверяет, заполнены ли все поля. Если нет, выводит сообщение.

if (!password.equals(confirmPassword)) { ... }

Проверяет, совпадают ли пароли. Если нет, выводит сообщение.

if (db.isUserExists(login)) { ... }

Проверяет, есть ли уже пользователь с таким логином. Если есть, сообщает, что логин занят.

String passwordHash = PasswordUtils.hashPassword(password);

Хеширует пароль для безопасности.

boolean success = db.addUser(login, passwordHash);

Добавляет нового пользователя в базу данных.

if (success) { ... }

Если регистрация успешна:

Выводит сообщение об успехе.

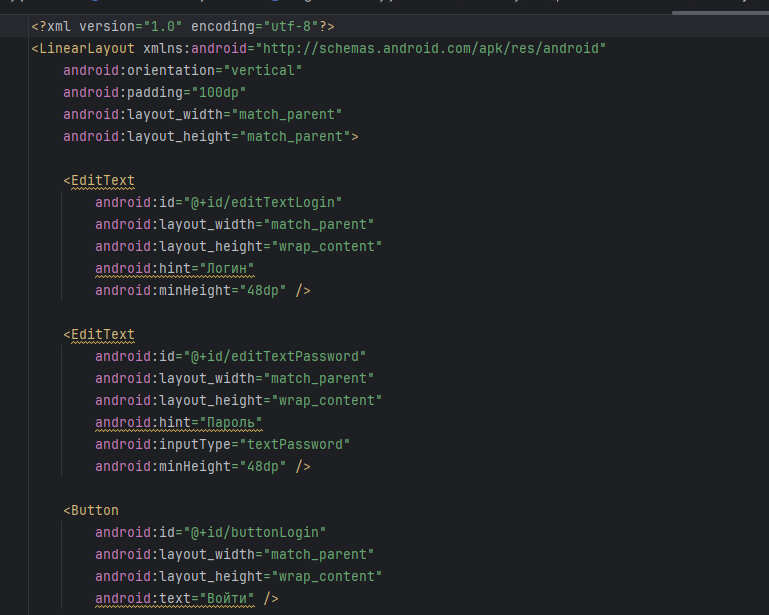
Через 1 секунду автоматически переходит к MainActivity, закрывая текущую.

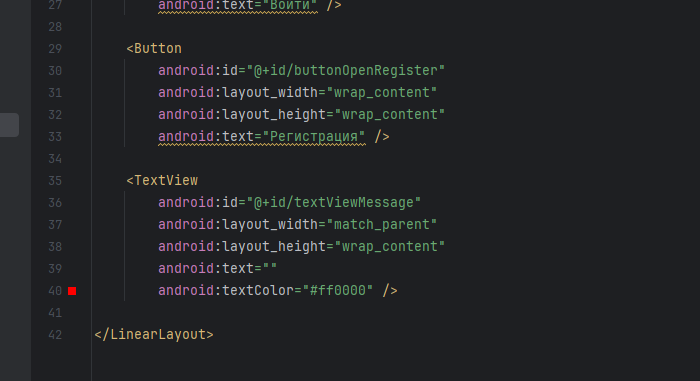
else { messageTextView.setText("Ошибка при регистрации"); }

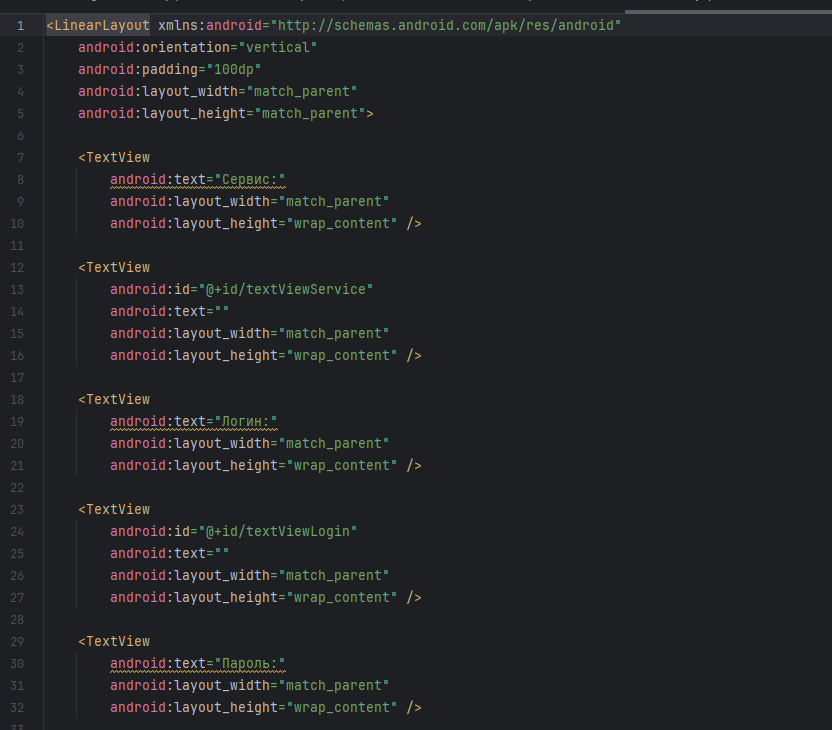
Если что-то пошло не так, показывает сообщение об ошибке.

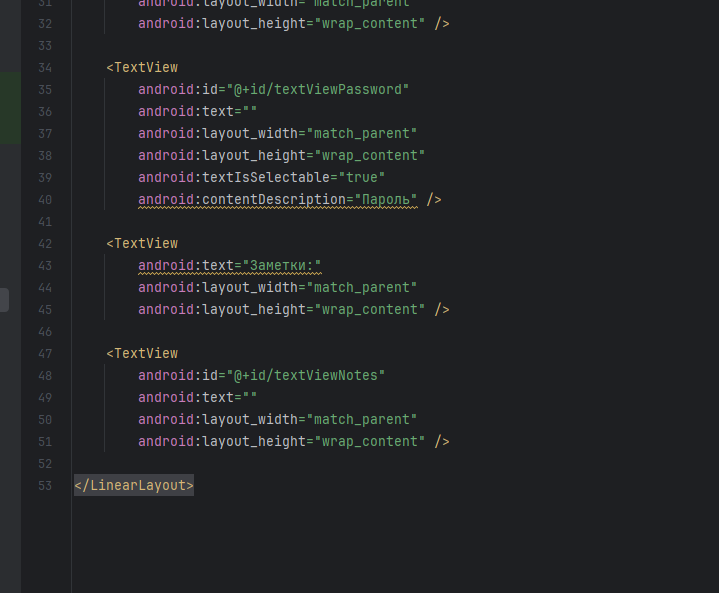
  
Рисунок 20. – activity\_add\_password

  
Рисунок 21. - activity\_add\_password

  
Рисунок 22. – activity\_main

  
Рисунок 23. - activity\_main

  
Рисунок 24. – activity\_password\_detail.

  
Рисунок 25. - activity\_password\_detail

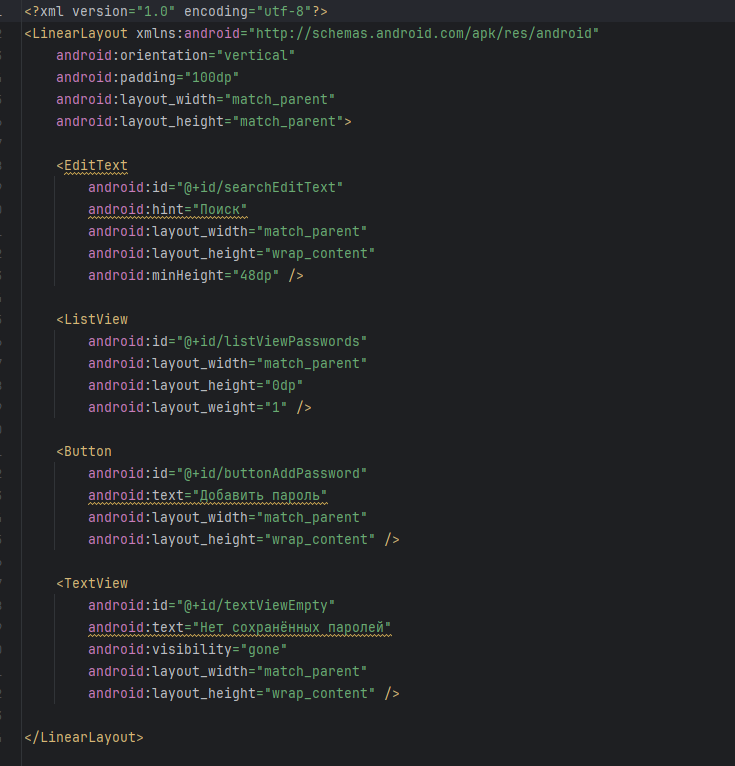
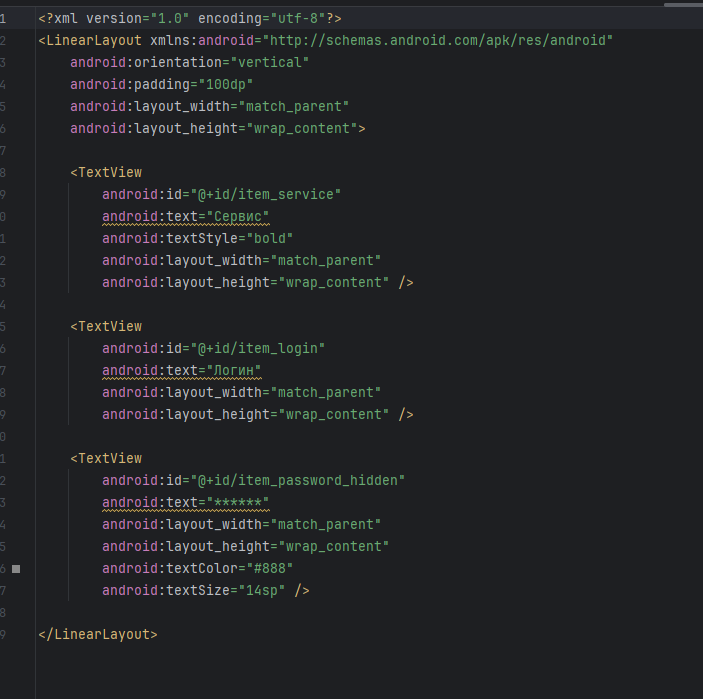
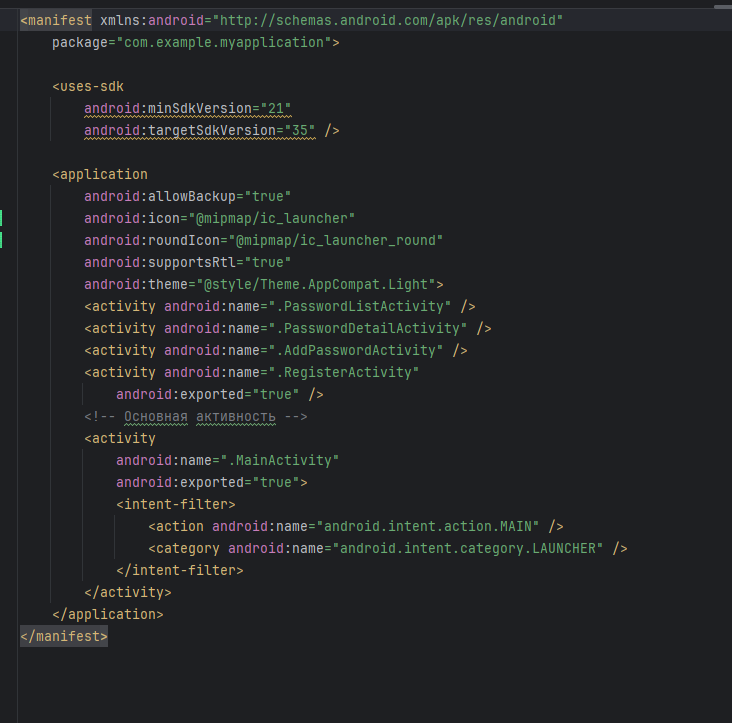
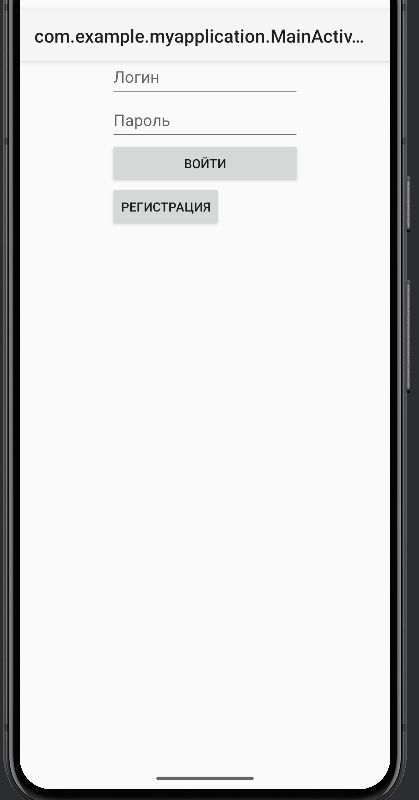
  
Рисунок 26. – activity\_password\_list.

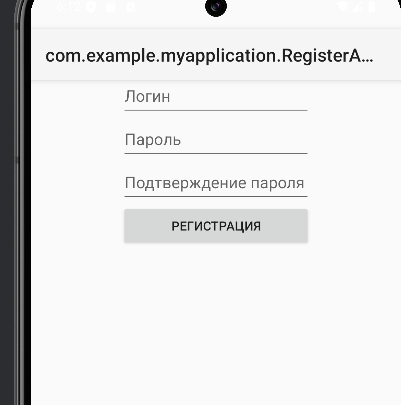
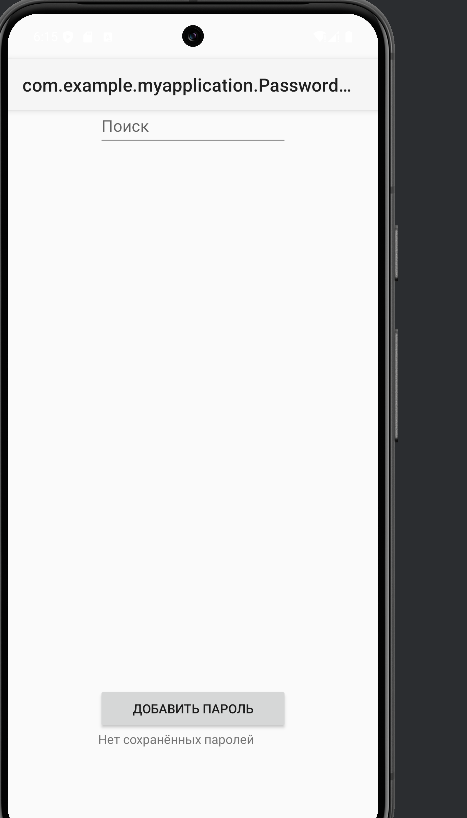
  
Рисунок 27. – activity\_registr.

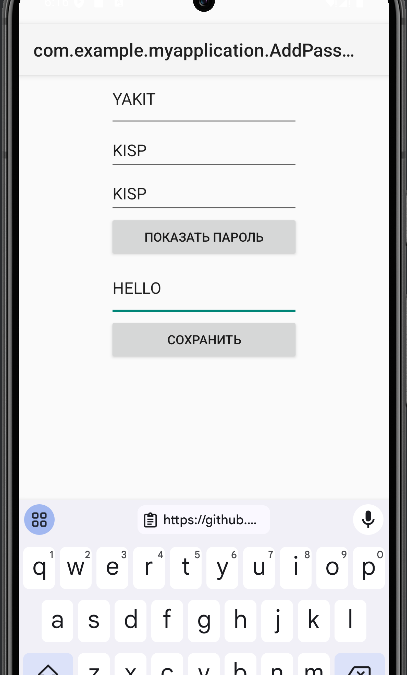
  
Рисунок 28. - activity\_registr.

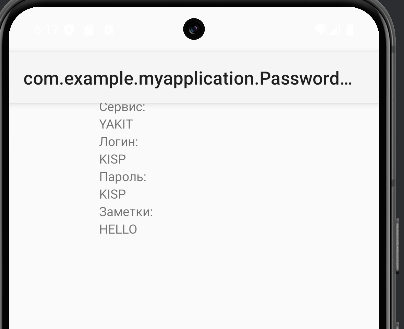
  
Рисунок 29. – item\_password.

  
Рисунок 30. – manifest.

  
Рисунок 31. – Главный экран.

  
Рисунок 32. – Экран Регистрации.  
  
Рисунок 33. – Нажимаем добавить пароль.

  
Рисунок 34. – Вводим данные и нажимаем сохранить.

  
Рисунок 35. – Успешно.