ANDROID STUDIO

DeepSeek GUI Plugin

v0.2.0 for Windows

ПОЛНЫЙ ТЕХНИЧЕСКИЙ ОТЧЁТ

Только DeepSeek API • Только онлайн • Никаких локальных моделей

Платформа	Windows 10/11
IDE	Android Studio Ladybug 2024.2+
AI-бэкенд	DeepSeek Chat + Coder API
Лицензия	MIT (Open Source)
Дата	Февраль 2026

1. Концепция проекта

Android Studio DeepSeek GUI — плагин для Android Studio (IntelliJ Platform), интегрирующий DeepSeek Chat и DeepSeek Coder API непосредственно в процесс Android-разработки.

Основная задача: предоставить Android-разработчикам на Windows визуальный AI-ассистент, понимающий специфику Android — Kotlin/Java, XML layouts, Gradle, Jetpack Compose, архитектурные паттерны (MVVM, Clean Architecture, MVI).

КЛЮЧЕВЫЕ ПРИНЦИПЫ

- **Строго DeepSeek API** — никаких локальных моделей, Ollama или других LLM-провайдеров
- **Изоляция AI-логики** — Node.js прослойка (ai-bridge) между IDE и DeepSeek API
- **Отзывчивый UI** — стриминг ответов через SSE, неблокирующие операции даже на слабых Windows-машинах
- **Android-first** — контекст проекта автоматически обогащает промпты (minSdk, зависимости, Compose)

Целевая аудитория: Android-разработчики на Windows 10/11, использующие Android Studio Ladybug (2024.2.1) и новее.

2. Архитектура проекта

Проект построен на трёхуровневой архитектуре, изолирующей AI-логику от UI и IDE-интеграции.

🎨 FRONTEND React + WebView2	⚙️ BACKEND Java/Kotlin + IntelliJ SDK	🤖 AI-BRIDGE Node.js → DeepSeek API
Chat Window (React 18) Code Preview + Diff Theme sync (Dark/Light) Android resource hints	AndroidContextCollector Manifest/Gradle parser IntelliJ Actions/Services Session management (SQLite)	DeepSeek Chat endpoint DeepSeek Coder endpoint SSE streaming proxy Android prompt templates

3. Функциональные блоки

3.1. DeepSeek Android Engine

Плагин использует два API-эндпоинта DeepSeek для разных задач:

Подсистема	Endpoint	Назначение
DeepSeek Chat	`/v1/chat/completions`	Архитектура, объяснения, review
DeepSeek Coder	`/v1/code/completions`	Генерация, рефакторинг, миграция

Android-специфичные шаблоны промптов

```
// ai-bridge/templates/android-templates.js
const ANDROID_TEMPLATES = {
  generateActivity: `
    Создай Activity на Kotlin для Android.
    Экран: {{features}}. ViewBinding + Material 3.`,

  generateComposable: `
    Jetpack Compose компонент: {{description}}
    Material Design 3 + модификаторы + корутины.`,

  generateViewModel: `
    ViewModel для {{screen}}.
    StateFlow + viewModelScope. Clean/MVVM.`,

  migrateToCompose: `
    Мигрируй XML layout в Compose:
    {{xml_content}}`,

  generateTests: `
```

```
    Напиши тесты для {{target}}.
    JUnit 5 + MockK + Turbine для Flow.`
};
```

3.2. Android Context Analyzer

Автоматический сбор контекста проекта для обогащения промптов. Собираются данные из AndroidManifest.xml, build.gradle(.kts) и структуры модулей.

```
// AndroidContextCollector.java
public class AndroidContextCollector {
    public AndroidProjectContext collectContext(Project project) {
        AndroidProjectContext ctx = new AndroidProjectContext();

        // Manifest
        ctx.manifest     = readManifest(project);
        ctx.packageName  = extractPackageName(ctx.manifest);
        ctx.minSdk       = extractMinSdk(ctx.manifest);
        ctx.targetSdk    = extractTargetSdk(ctx.manifest);

        // Gradle
        ctx.dependencies = extractDependencies(project);
        ctx.compileSdk   = extractCompileSdk(project);
        ctx.jdkVersion   = extractJdkVersion(project);
        ctx.kotlinVersion = extractKotlinVersion(project);

        // Структура
        ctx.modules       = getAndroidModules(project);
        ctx.hasCompose    = detectCompose(project);
        ctx.hasViewBinding = detectViewBinding(project);
        ctx.hasHilt       = detectHilt(project);
        ctx.hasRoom       = detectRoom(project);

        return ctx;
    }
}
```

3.3. Умные команды

Быстрые слэш-команды для генерации Android-кода:

Команда	Описание	Результат
`/init-android`	Новый проект с чистой архитектурой	Полный каркас
`/activity`	Генерация Activity или Fragment	Kotlin + XML/Compose
`/compose`	Создать Composable компонент	Material 3 UI
`/viewmodel`	ViewModel + StateFlow + UseCase	MVVM готовый
`/repository`	Repository с Room или Retrofit	Data layer
`/di`	Hilt/Dagger модуль	DI конфигурация
`/test`	Unit/UI тесты	JUnit + MockK
`/migrate-compose`	Конвертация XML → Compose	Миграция UI
`/lint-fix`	Исправить Lint-предупреждения через AI	Авто-фикс
`/explain`	Объяснить выделенный код	Документация

3.4. Работа с Android-кодом

@file reference с Android-спецификой. Подсказки ресурсов прямо в чате: @layout/activity_main.xml, @drawable/ic_launcher, @string/app_name, @color/primary, @dimen/margin_standard.

Code DIFF с превью. Визуальное сравнение XML layouts, preview Compose кода до применения, проверка совместимости версий SDK.

Batch Write. Массовое создание MVVM-компонентов (Activity + ViewModel + Layout), генерация Room Entity + DAO + Database, создание Retrofit-сервисов + моделей данных.

3.5. Управление сессиями

Данные хранятся локально в Windows:

```
%APPDATA%\AndroidStudioDeepSeek\sessions.db      # SQLite — история диалогов
%APPDATA%\AndroidStudioDeepSeek\favorites.json   # Избранное
%APPDATA%\AndroidStudioDeepSeek\api_keys.enc     # AES-256 зашифрованные ключи
%APPDATA%\AndroidStudioDeepSeek\context_cache.db # Кэш контекста проектов
```

Автоматическая фильтрация истории по package name текущего проекта. Контекстное переключение между модулями multi-module проекта.

4. Технический стек

Компонент	Технология	Назначение
Ядро плагина	`Java 17 + Kotlin 2.0`	Интеграция с IntelliJ Platform / Android Studio
	`IntelliJ Platform SDK 2024.2+`	API для работы с IDE и проектом
	`Android Plugin API`	Доступ к Android Facet, Gradle, Manifest
AI Backend	`Node.js 22 LTS`	Прокси-сервер DeepSeek API (ai-bridge)
	`Express 5 + Axios`	REST/SSE эндпоинты, HTTP-клиент
	`node-cache`	Кэширование ответов DeepSeek
Frontend	`TypeScript 5.6+`	Типизированная логика WebView
	`React 19`	UI-компоненты чата
	`Fluent UI React v9`	Визуальный стиль Windows 11
	`Edge WebView2`	Рендеринг (встроен в Win11)
Сборка	`Gradle IntelliJ Plugin 2.x`	Компиляция и упаковка плагина
	`Vite 6`	Быстрая сборка фронтенда
Хранение	`SQLite 3.45`	История, сессии, кэш контекста
Безопасность	`AES-256-GCM (Java JCE)`	Шифрование API-ключей DeepSeek

5. Детальный план реализации

ФАЗА 0: ПОДГОТОВКА WINDOWS-СРЕДЫ

```
# PowerShell — проверка и установка
if (-not (Test-Path "C:\Program Files\Android\Android Studio")) {
    Write-Error "Установите Android Studio Ladybug+"
    exit 1
}
winget install Microsoft.OpenJDK.17
winget install OpenJS.NodeJS.LTS
winget install Git.Git

git clone https://github.com/your-org/android-studio-deepseek-plugin
cd android-studio-deepseek-plugin
```

ФАЗА 1: СТРУКТУРА ПРОЕКТА (build.gradle.kts)

```
plugins {
    id("java")
    id("org.jetbrains.intellij.platform") version "2.2.1"
    id("org.jetbrains.kotlin.jvm") version "2.0.21"
}

group = "com.deepseek.androidstudio"
version = "0.2.0-windows"

repositories {
    mavenCentral()
    intellijPlatform { defaultRepositories() }
    google()
}

dependencies {
    intellijPlatform {
        create("AI", "2024.2.1")  // Android Studio Ladybug
        plugin("com.intellij.java")
        plugin("org.jetbrains.android")
        instrumentationTools()
    }
    implementation("org.xerial:sqlite-jdbc:3.45.1.0")
    implementation("com.google.code.gson:gson:2.11.0")
    implementation("com.squareup.okhttp3:okhttp:4.12.0")
    implementation("com.squareup.okhttp3:okhttp-sse:4.12.0")
```

```
}

tasks {
    patchPluginXml {
        sinceBuild.set("242.*")  // Ladybug
        untilBuild.set("252.*")  // До Narwhal
    }
}
```

ФАЗА 2: AI-BRIDGE (DeepSeek API Proxy, Node.js)

Структура ai-bridge:

```
ai-bridge/
├── package.json
├── server.js              # Express + SSE
├── deepseek/
│   ├── chat.js            # DeepSeek Chat wrapper
│   └── coder.js           # DeepSeek Coder wrapper
├── android/
│   ├── templates.js       # Android prompt templates
│   └── context-builder.js # Обогащение контекста
└── utils/
    ├── tokenizer.js       # Подсчёт токенов
    ├── cache.js           # node-cache
    └── rate-limiter.js    # Rate limiting
```

server.js — главный файл ai-bridge:

```
const express = require('express');
const cors = require('cors');
const axios = require('axios');

const app = express();
app.use(cors());
app.use(express.json({ limit: '50mb' }));

const DEEPSEEK = {
  chat:  'https://api.deepseek.com/v1/chat/completions',
  coder: 'https://api.deepseek.com/v1/code/completions',
};

// DeepSeek Chat (синхронный)
app.post('/api/chat', async (req, res) => {
  const { message, context, apiKey, model='deepseek-chat' } = req.body;
  try {
    const resp = await axios.post(DEEPSEEK.chat, {
```

```
      model,
      messages: [
        { role:'system', content: 'Ты Android-эксперт. Kotlin, Compose, XML.' },
        { role:'user', content: enrichPrompt(message, context) }
      ],
      stream: false, temperature: 0.7
    }, { headers: { Authorization: `Bearer ${apiKey}` } });
    res.json(resp.data);
  } catch (err) { res.status(500).json({ error: err.message }); }
});

// DeepSeek Coder (SSE streaming)
app.post('/api/code', async (req, res) => {
  res.setHeader('Content-Type', 'text/event-stream');
  res.setHeader('Cache-Control', 'no-cache');
  const { prompt, apiKey, androidContext } = req.body;
  try {
    const resp = await axios({ method:'post', url:DEEPSEEK.coder,
      data: { prompt: enrichPrompt(prompt, androidContext),
              max_tokens:4096, temperature:0.3 },
      headers: { Authorization: `Bearer ${apiKey}` },
      responseType: 'stream' });
    resp.data.on('data', c => res.write(`data: ${c}\n\n`));
    resp.data.on('end', () => { res.write('data: [DONE]\n\n'); res.end(); });
  } catch (err) {
    res.write(`data: ${JSON.stringify({error:err.message})}\n\n`);
    res.end();
  }
});

function enrichPrompt(prompt, ctx) {
  if (!ctx) return prompt;
  let e = prompt;
  if (ctx.minSdk) e += `\nMin SDK: ${ctx.minSdk}`;
  if (ctx.hasCompose) e += '\nПроект использует Jetpack Compose';
  if (ctx.hasHilt) e += '\nDI: Hilt';
  if (ctx.packageName) e += `\nPackage: ${ctx.packageName}`;
  return e;
}

app.listen(3012, '127.0.0.1', () =>
  console.log('DeepSeek Android Bridge on :3012'));
```

ФАЗА 3: JAVA/KOTLIN BACKEND (Ядро плагина)

Структура src/main/java/com/deepseek/androidstudio/:

```
├── services/
│   ├── DeepSeekService.java          # HTTP-клиент к ai-bridge
│   ├── AndroidContextService.java    # Сбор контекста проекта
│   ├── SessionService.java           # SQLite сессии
│   └── StreamingService.java         # SSE-клиент для стриминга
├── actions/
│   ├── GenerateActivityAction.java
│   ├── GenerateComposableAction.java
│   ├── GenerateViewModelAction.java
│   ├── ExplainCodeAction.java
│   └── MigrateToComposeAction.java
├── listeners/
│   └── AndroidProjectListener.java
├── ui/
│   ├── DeepSeekToolWindowFactory.java
│   └── settings/
│       └── DeepSeekSettingsConfigurable.java
└── utils/
    ├── AndroidXmlParser.java
    ├── ApiKeyEncryption.java     # AES-256-GCM
    └── GradleFileAnalyzer.java
```

DeepSeekService.java (ключевой сервис):

```
@Service
public final class DeepSeekService {
    private static final String BRIDGE = "http://127.0.0.1:3012";
    private final OkHttpClient client = new OkHttpClient.Builder()
        .connectTimeout(30, TimeUnit.SECONDS)
        .readTimeout(120, TimeUnit.SECONDS).build();

    public CompletableFuture<String> chat(String msg, Project project) {
        return CompletableFuture.supplyAsync(() -> {
            var ctx = AndroidContextService.getInstance()
                .collectContext(project);
            var key = DeepSeekSettings.getInstance().getApiKey();
            var body = RequestBody.create(
                MediaType.parse("application/json"),
                new Gson().toJson(Map.of(
                    "message", msg,
                    "androidContext", ctx,
                    "apiKey", key
```

```
                ")));
            var req = new Request.Builder()
                .url(BRIDGE + "/api/chat").post(body).build();
            try (var resp = client.newCall(req).execute()) {
                return resp.body().string();
            } catch (IOException e) {
                return "Ошибка: " + e.getMessage();
            }
        });
    }
}
```

plugin.xml:

```
<idea-plugin>
  <id>com.deepseek.androidstudio.windows</id>
  <name>DeepSeek AI for Android Studio</name>
  <vendor>DeepSeek</vendor>

  <depends>com.intellij.modules.platform</depends>
  <depends>com.intellij.modules.androidstudio</depends>
  <depends>org.jetbrains.android</depends>

  <extensions defaultExtensionNs="com.intellij">
    <toolWindow id="DeepSeek Android AI"
      icon="/icons/deepseek-android.svg"
      anchor="right"
      factoryClass="...ui.DeepSeekToolWindowFactory"/>
    <applicationService
      serviceImpl="...services.DeepSeekService"/>
    <projectService
      serviceImpl="...services.AndroidContextService"/>
    <applicationConfigurable parentId="tools"
      instance="...settings.DeepSeekSettingsConfigurable"
      displayName="DeepSeek AI"/>
  </extensions>

  <actions>
    <action id="DeepSeek.GenerateComposable"
      class="...actions.GenerateComposableAction"
      text="Generate Compose UI with DeepSeek">
      <add-to-group group-id="NewGroup" anchor="last"/>
    </action>
    <action id="DeepSeek.ExplainCode"
      class="...actions.ExplainCodeAction"
      text="Explain with DeepSeek">
      <add-to-group group-id="EditorPopupMenu"/>
```

```
    </action>
  </actions>
</idea-plugin>
```

ФАЗА 4: WEBVIEW FRONTEND (React 19 + Fluent UI)

```
webview/
├── src/
│   ├── components/
│   │   ├── AndroidChatWindow.tsx    # Главное окно чата
│   │   ├── CodeBlockWithDiff.tsx    # Код + diff превью
│   │   ├── ComposePreview.tsx       # Preview Compose UI
│   │   ├── XmlTreeView.tsx          # XML layout viewer
│   │   └── CommandPalette.tsx       # Слэш-команды
│   ├── hooks/
│   │   ├── useDeepSeekStream.ts     # SSE hook
│   │   └── useAndroidStudioTheme.ts # Theme sync
│   └── styles/
│       └── tokens.css               # CSS custom properties
├── vite.config.ts
└── package.json
```

useAndroidStudioTheme.ts — синхронизация темы:

```
import { useState, useEffect } from 'react';

export const useAndroidStudioTheme = () => {
  const [isDark, setIsDark] = useState(false);
  const [accent, setAccent] = useState('#3DDC84');

  useEffect(() => {
    const bg = getComputedStyle(document.body).backgroundColor;
    const [r, g, b] = bg.match(/\d+/g)?.map(Number) ?? [255,255,255];
    setIsDark((r + g + b) / 3 < 128);

    const link = getComputedStyle(document.body)
      .getPropertyValue('--link-color');
    if (link) setAccent(link);
  }, []);

  return { isDark, accent };
};
```

ФАЗА 5-6: ANDROID-ИНТЕГРАЦИЯ + ТЕСТИРОВАНИЕ

Manifest Parser — парсинг package, SDK версий, списка Activity через javax.xml.parsers.
Compose Generator — шаблонная генерация @Composable функций с Modifier,

collectAsState, Material 3. **Lint Integration** — IntelliJ LocalInspectionTool для проверки устаревших API и производительности layout через DeepSeek.

Unit-тесты (JUnit 5 + MockK):

```
class DeepSeekAndroidServiceTest {
    @Test fun testContextCollection() {
        val project = mock(Project::class.java)
        val ctx = AndroidContextService.getInstance()
            .collectContext(project)
        assertNotNull(ctx)
        assertTrue(ctx.packageName.isNotEmpty())
    }

    @Test fun testComposeGeneration() {
        val result = DeepSeekService.getInstance()
            .generateComposable("Кнопка + иконка", null)
            .get(10, TimeUnit.SECONDS)
        assertTrue(result.contains("@Composable"))
    }
}
```

Интеграционные тесты Windows (PowerShell):

```
# test-android-plugin.ps1
$androidStudio = "C:\Program Files\Android\Android Studio"
if (Test-Path $androidStudio) { Write-Host "✅ Android Studio" }

$api = Invoke-RestMethod -Uri "http://127.0.0.1:3012/api/health"
if ($api.status -eq "ok") { Write-Host "✅ DeepSeek Bridge" }

./gradlew buildPlugin
if ($LASTEXITCODE -eq 0) { Write-Host "✅ Plugin build OK" }
```

ФАЗА 7: СБОРКА ДИСТРИБУТИВА

```
@echo off
echo Building DeepSeek Android Studio Plugin...
call .\gradlew clean buildPlugin
cd webview && call npm ci && call npm run build && cd ..
mkdir build\distributions\bridge
xcopy ai-bridge build\distributions\bridge /E /I
cd build\distributions
jar -cf deepseek-android-studio-0.2.0.zip *
echo Done! Size: ~35 MB
```

6. Системные требования

Параметр	Минимальные	Рекомендуемые
ОС	Windows 10 1909+	Windows 11 23H2+
IDE	Android Studio Ladybug 2024.2.1	Android Studio Meerkat 2025.1+
RAM	4 ГБ (рекомендуется 8 ГБ)	16+ ГБ
Node.js	18 LTS	22 LTS
Диск	HDD, 200 МБ свободно	SSD, 1 ГБ для кэша
Сеть	Стабильное подключение	Широкополосный интернет
WebView2	Edge WebView2 Runtime	Встроен в Windows 11

7. Конфигурация

Получение API-ключа DeepSeek: зарегистрируйтесь на platform.deepseek.com, создайте ключ в разделе «Api Keys» (формат: sk-...), вставьте в настройки плагина (Settings → Tools → DeepSeek AI).

```
// DeepSeekSettings.java
public class DeepSeekSettings {
    private String apiKey;                    // Шифруется AES-256-GCM
    private String model    = "deepseek-coder"; // Модель по умолчанию
    private boolean stream  = true;            // SSE стриминг
    private int maxTokens   = 4096;            // Макс. токены ответа
    private double temp     = 0.5;             // Temperature

    public void save() {
        Preferences prefs = Preferences.userRoot()
            .node("com/deepseek/androidstudio");
        prefs.put("apiKey", encrypt(apiKey));
        prefs.put("model", model);
        prefs.putBoolean("stream", stream);
    }
}
```

8. Чек-лист перед релизом

Задача	Статус	Приоритет
Совместимость с Android Studio Ladybug / Meerkat	☐	P0
Поддержка Java 17 + Kotlin 2.0 проектов	☐	P0
Анализ AndroidManifest.xml	☐	P0
Парсинг build.gradle(.kts) зависимостей	☐	P0
Генерация Activity / Fragment	☐	P0
Генерация Jetpack Compose UI	☐	P0
Генерация ViewModel + StateFlow	☐	P1
Генерация Room Entity / DAO	☐	P1
Генерация Retrofit-сервисов	☐	P1
DI-генерация (Hilt)	☐	P1
XML layout анализ и исправление	☐	P1
Миграция XML → Compose	☐	P2
Unit/UI тест генерация	☐	P2
Шифрование API-ключей AES-256-GCM	☐	P0
SSE streaming с прогрессом	☐	P0
Русская + English локализация	☐	P1
Windows 11 Mica / темы	☐	P2
Автообновление через JetBrains Marketplace	☐	P2
Документация (RU + EN)	☐	P1

9. Заключение

Android Studio DeepSeek GUI Plugin — профессиональный инструмент для Android-разработчиков, построенный на следующих принципах:

- **Строго DeepSeek API** — никаких локальных моделей, высокая скорость ответа, актуальные модели
- **Android-first** — Compose, XML, Gradle, Room, Retrofit, Hilt, архитектурные паттерны
- **Windows-оптимизация** — Edge WebView2, нормализация путей, Fluent UI, Mica
- **Безопасность** — AES-256-GCM для API-ключей, контроль записи файлов
- **Удобство** — слэш-команды, контекстное меню, SSE-стриминг, история сессий
- **Двуязычность** — полная поддержка русского и английского языков

Размер пакета	~35 МБ
Распространение	JetBrains Marketplace + GitHub Releases
Лицензия	MIT (Open Source)
Версия	v0.2.0