

Semestrálna práca z predmetu   
*vývoj aplikácií pre mobilné zariadenia*

FInance Data Manager

**Vypracoval**: Michal Poprac

**Študijná skupina**: 5ZYS21

**Akademický rok:** 2024/2025 V Žiline dňa 1.5.2025

Obsah

[Úvod 2](#_Toc197026920)

[Prehľad podobných aplikácií 2](#_Toc197026921)

[Analýza navrhovanej aplikácie 4](#_Toc197026922)

[Návrh architektúry aplikácie 4](#_Toc197026923)

[Návrh vzhľadu obrazoviek 5](#_Toc197026924)

[Analýza implementovanej aplikácie 9](#_Toc197026925)

[Analýza používateľského rozhrania 9](#_Toc197026926)

[Analýza strednej (validačnej) vrstvy aplikácie 10](#_Toc197026927)

[Analýza jadra aplikácie (backendu) 10](#_Toc197026928)

[Architektúra aplikácie 11](#_Toc197026929)

[12](#_Toc197026930)

[Použité knižnice 13](#_Toc197026931)

[Používateľský scenár (Usage Flow) 14](#_Toc197026932)

[Zoznam zdrojov pri implementácií 14](#_Toc197026933)

[Zoznam zdrojov 14](#_Toc197026934)

# Úvod

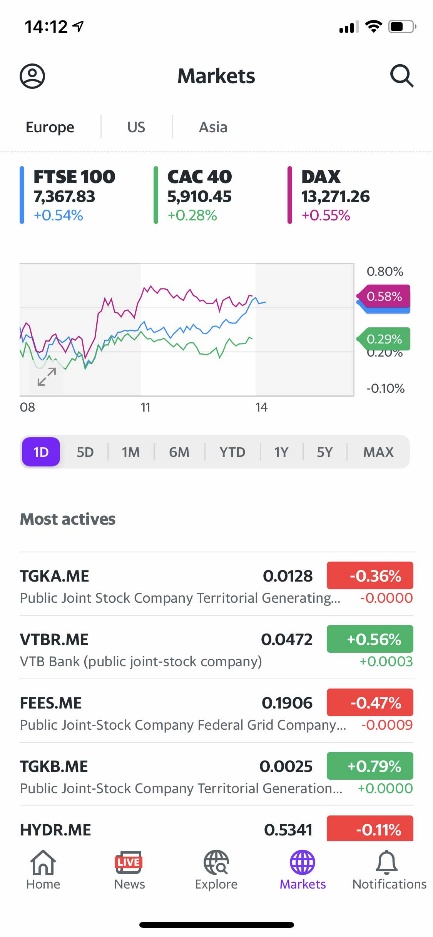
**Finance Data Manager** bude aplikácia určená na jednoduchú prácu s rôznymi finančnými dátami, ako sú akcie. Používateľom poskytne možnosť vizualizácie údajov prostredníctvom grafov a tabuliek. Súčasťou aplikácie budú aj matematické modely na identifikáciu trendov v dátach a ich predikciu.

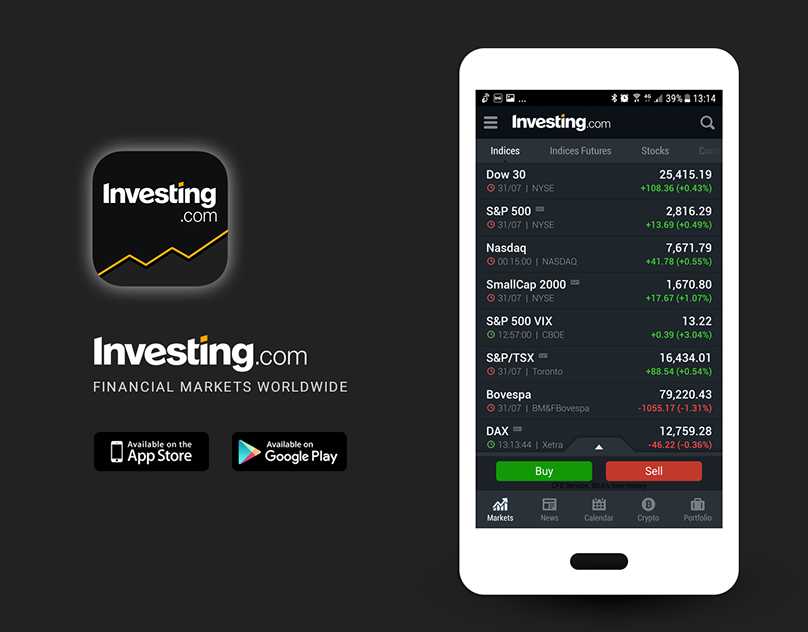
Aplikácia bude klásť dôraz na minimalistický a elegantný dizajn, pričom hlavným cieľom je intuitívnosť a prehľadnosť poskytovaných funkcií.

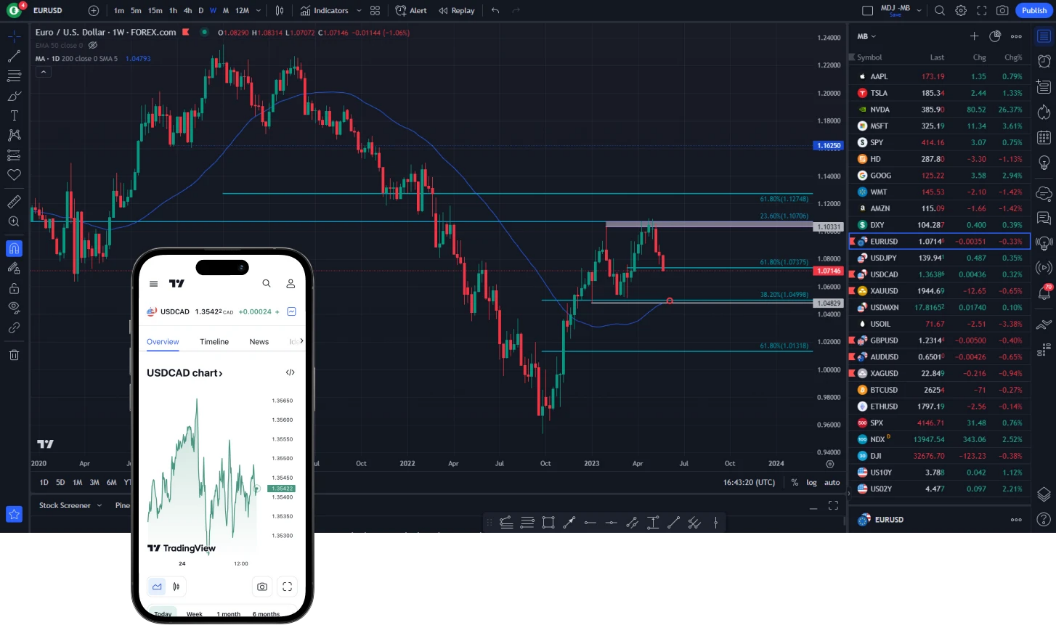
Tému som si vybral, pretože som už v minulosti vytváral podobnú aplikáciu v inom programovacom jazyku. **Kotlin** ponúka rozsiahle knižnice na matematické modelovanie a vizualizáciu dát, čo umožní efektívnu implementáciu požadovaných funkcionalít. Zároveň mi táto téma pripadá originálna a predstavuje zaujímavé spojenie možnosti predikcie trendov reálnych finančných dát.

# Prehľad podobných aplikácií

Ako príklad podobných aplikácií som vybral nasledovné, ktoré slúžia na podobné účely:

**Yahoo Finance** – Umožňuje sledovať finančné trhy, analyzovať akcie a zobrazovať dáta vo forme grafov.

**Investing.com** – Poskytuje nástroje na analýzu trhových trendov, ekonomické kalendáre a technické indikátory.

**TradingView** – Obsahuje pokročilé grafické nástroje na analýzu trhových dát a podporuje predikčné modely.

Tieto aplikácie ma inšpirovali najmä v oblasti vizualizácie dát a analytických funkcií, ktoré by som chcel integrovať aj do **Finance Data Manager**.

# Analýza navrhovanej aplikácie

Možné použitie aplikácie zahŕňa napríklad:

**Analýza vývoja akcií a investícií**

* Používateľ si môže vizualizovať historický vývoj cien akcií vo forme grafov.
* Aplikácia umožní vykonávať **polynomiálnu aproximáciu** trendov a predikovať budúce ceny.
* Pomocou **MA (Moving Average)** a **AR (AutoRegression)** modelov môže používateľ filtrovať výkyvy a analyzovať dlhodobé trendy.

Aplikácia **Finance Data Manager** bude primárne slúžiť na vizualizáciu finančných dát vo forme grafov a tabuliek, ako bolo uvedené vyššie. Okrem vizualizácie bude podporovať aj matematické modely na analýzu dát, napríklad:

* **Polynomiálnu aproximáciu** na identifikáciu trendov v dátach.
* **Predikciu na základe aproximácie**, umožňujúcu odhad budúceho vývoja.
* **Modely MA (Moving Average) a AR (Autoregression)**, ktoré budú slúžiť na filtráciu a predikciu dát.

Aplikácia bude získavať aktuálne finančné dáta prostredníctvom API, čím sa zabezpečí ich aktuálnosť a relevantnosť. O vizualizáciu údajov sa postarajú špecializované knižnice určené na tento účel.

Súčasťou aplikácie bude aj **databáza používateľov**, v ktorej budú uložené ich účty (prihlasovacie údaje), preferované matematické modely a ďalšie nastavenia.

Matematické modely budú implementované s využitím vhodných knižníc, pričom ich konkrétny výber a integrácia budú podrobnejšie popísané v časti **návrh architektúry aplikácie**.

# Návrh architektúry aplikácie

Aplikácia **Finance Data Manager** bude pozostávať z nasledujúcich hlavných častí:

**1. Jadro (Backend)**

Táto časť bude zodpovedná za **spracovanie dát**, **prácu s databázou**, **volanie API** a **vytváranie matematických modelov**. Obsahovať bude nasledujúce triedy:

**DBManager** – správa databázy používateľov a ich preferencií.

**APIManager** – komunikácia s externými API na získavanie aktuálnych finančných dát.

**Model** (supertrieda) – základná trieda pre všetky matematické modely, z ktorej budú dedičné triedy ako:

PolynomialApproximation – polynomiálna aproximácia trendu.

MovingAverageModel – model kĺzavého priemeru (MA).

AutoRegressionModel – autoregresný model (AR).

**DataManager** – centrálna trieda, ktorá bude riadiť spoluprácu medzi vyššie uvedenými triedami.

**2. Stredná vrstva**

Táto vrstva bude zabezpečovať **validáciu požiadaviek** z frontendu a **kontrolu dát** prichádzajúcich z jadra aplikácie. Slúži ako sprostredkovateľ medzi klientskou časťou a backendom. Obsahovať bude:

**RequestManager** – validácia požiadaviek používateľa a správa komunikácie medzi frontendom a backendom.

**3. Frontend**

Používateľské rozhranie, kde sa budú zobrazovať **grafy, tabuľky, modely a vizualizácie dát**. Táto vrstva bude obsahovať implementácie na interakciu s aplikáciou, vrátane:

Vizualizácie finančných dát.

Zobrazenia výsledkov matematických modelov.

Správy používateľských účtov a preferencií.

**Cieľ návrhu**

Navrhnutá **vrstvová architektúra** zabezpečí **robustnosť aplikácie**, **kontrolu vstupov a výstupov** a efektívne oddelenie jednotlivých funkčných častí. Tento prístup uľahčí **údržbu a rozšíriteľnosť aplikácie**.

Prikladám návrh knižníc, ktoré môžem použiť pre jednotlivé časti aplikácie:

|  |  |
| --- | --- |
| **Funkcionalita** | **Knižnice** |
| **Databáza** | Room, Exposed |
| **API** | Retrofit, Ktor Client |
| **Matematické modely** | Kotlin Statistics, Smile, Apache Commons Math |
| **Vizualizácia** | MPAndroidChart, AnyChart, TableView |
| **Architektúra** | Hilt, ViewModel + LiveData, Kotlin Coroutines |

# Návrh vzhľadu obrazoviek

Ako som spomenul vyššie, obrazovky aplikácie majú pôsobiť minimalistickým a elegantným dojmom a poskytovať jednoduché ovládanie. Na tento účel sa navrhuje, aby používateľské rozhranie bolo intuitívne a prívetivé.

**Prikladám príklad aktuálne už naimplementovaných obrazoviek, a to:**

A screenshot of a phone

AI-generated content may be incorrect.**Login Screen** (Prihlasovacia obrazovka)

A screenshot of a black screen

AI-generated content may be incorrect.**New Account Screen** (Obrazovka pre vytvorenie nového účtu)

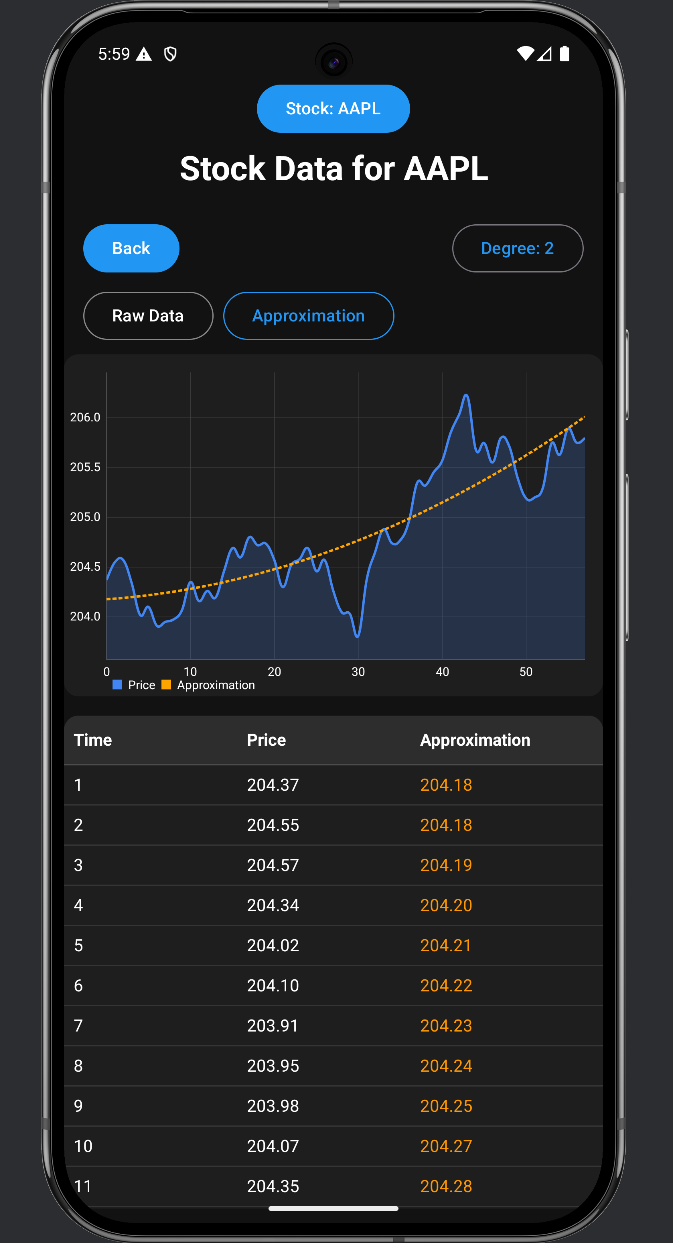
**Main Screen** (Hlavná obrazovka, zatiaľ len hrubý návrh bez pridania ďalších elementov)

A screenshot of a phone

AI-generated content may be incorrect.A screen shot of a phone

AI-generated content may be incorrect.

**Graph Page** (Obrazovka na zobrazenie a výber dát a prácu s mat. Modelmi)



Ako ďaľšie obrazovky by som chcel pridať:

**Settings:** ( Zoznam nastavení pre používateľa )

**Prípadne ďaľšie obrazovky rozširujúce funkcionality aplikácie**

# Analýza implementovanej aplikácie

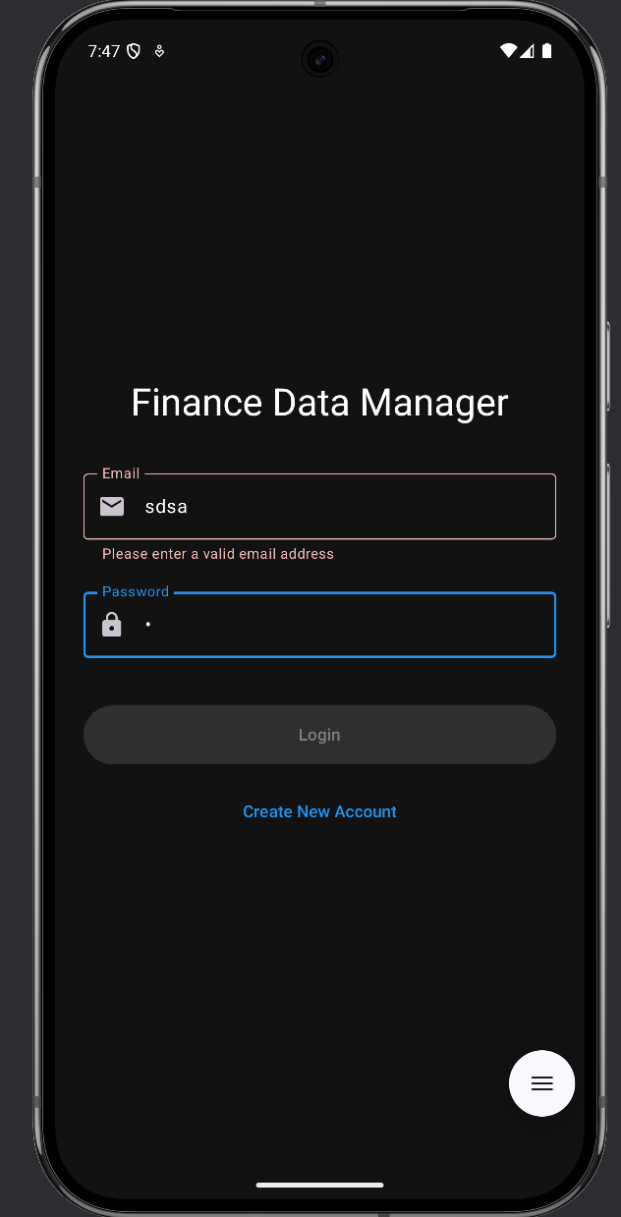
Finálna verzia implementovanej aplikácie zodpovedá pôvodne stanoveným požiadavkám definovaným v analytickej časti. Aplikácia pôsobí elegantne a používateľovi ponúka širokú paletu nástrojov na prácu s dátami spolu s intuitívnym a prehľadným používateľským rozhraním.

Popis jednotlivých častí aplikácie je uvedený v nasledujúcich častiach tohto dokumentu.

# Analýza používateľského rozhrania

Používateľské rozhranie pôsobí jednoduchým a elegantným dojmom, pričom dominuje tmavá farebná schéma. Štruktúra jednotlivých obrazoviek je navrhnutá konzistentne, rovnako ako ich ovládanie, čo prispieva k lepšej orientácii používateľa. Aplikácia kladie dôraz na používateľskú voľnosť a flexibilitu pri práci s dátami, no zároveň obsahuje množstvo kontrolných mechanizmov, ktoré v prípade potreby jasne a zrozumiteľne upozornia na chyby alebo nevhodné vstupy. Tým sa zabezpečuje spoľahlivosť a správnosť spracovávaných údajov.

Ako príklad prikladám prihlasovaciu obrazovku.



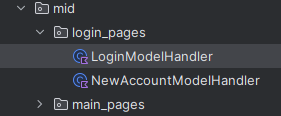
# Analýza strednej (validačnej) vrstvy aplikácie

Stredná vrstva predstavuje kľúčový aspekt celej aplikácie, pretože zabezpečuje sprostredkovanie medzi klientskou časťou (frontendom) a logikou aplikácie (backendom). Jej hlavnou úlohou je validácia toku požiadaviek a dát v oboch smeroch – od používateľa k jadru aplikácie aj späť. Táto vrstva zaručuje, že sa k backendu dostávajú len korektne spracované a overené údaje, a zároveň chráni používateľa pred nesprávnymi alebo neočakávanými výstupmi.

Dôležitou vlastnosťou tejto vrstvy je jej schopnosť abstrahovať komunikáciu medzi frontendom a backendom tak, že je možné jednu z týchto častí nahradiť bez potreby zásahu do druhej. Vďaka tomu je architektúra aplikácie flexibilná, modulárna a lepšie udržiavateľná.

Stredná vrstva je navrhnutá jednotne a systematicky – pre každú obrazovku alebo funkcionalitu používateľského rozhrania existuje adekvátne spracovanie vo forme samostatného funkčného objektu, čo zabezpečuje konzistenciu, opakovateľnosť a jednoduchšiu správu kódu.

Príklad – stredná vrstva pre prihlasovaciu obrazovku



# Analýza jadra aplikácie (backendu)

Táto časť aplikácie predstavuje centrálnu implementáciu všetkej logiky súvisiacej so získavaním, spracovaním a uchovávaním dát, ako aj tvorbou dátových modelov. Jadro aplikácie je striktne oddelené od používateľského rozhrania, s ktorým nekomunikuje priamo – všetka výmena informácií prebieha prostredníctvom definovaných rozhraní v strednej (validačnej) vrstve. Tým sa zabezpečuje modularita a možnosť nezávislého vývoja či výmeny komponentov.

**Funkcie a komponenty backendu:**

* **Logika spracovania dát:** Implementácia algoritmov, matematických modelov a pravidiel na transformáciu vstupov na požadované výstupy.
* **Tvorba dátových modelov:** Definovanie štruktúr, ktoré reprezentujú jednotlivé entity v aplikácii (napr. používateľ, záznam, výpočet).
* **Správa databáz:** Prístup k relačným alebo súborovým databázam, operácie CRUD (create, read, update, delete) a optimalizácia dotazov.
* **API klient:** Zabezpečuje externú komunikáciu s inými systémami alebo službami (napr. na import dát).
* **Autentifikačné rozhranie:** Komunikácia s databázou používateľov, overovanie prístupových údajov a správa bezpečnostných mechanizmov.

Takto navrhnutý backend zaručuje vysokú úroveň bezpečnosti, znovupoužiteľnosti kódu a udržiavateľnosti aplikácie. Vďaka presne definovanému rozhraniu medzi vrstvami je zároveň možné backend prispôsobiť rôznym typom klientskych aplikácií bez potreby zásahov do jeho vnútorného fungovania.

# Architektúra aplikácie

Ako je spomenuté vyššie, aplikácia je rozdelená do troch hlavných úrovní, pričom každá z nich plní špecifické úlohy v rámci celkovej architektúry:

**Prezentačná vrstva (UI):** Slúži na interakciu s používateľom. Zabezpečuje zobrazenie výstupov a zber vstupov cez intuitívne rozhranie. Používateľ prostredníctvom nej ovláda funkcionalitu aplikácie, pričom UI samotné neobsahuje žiadnu aplikačnú logiku.

**Stredná vrstva (validačná):** Predstavuje sprostredkovateľa medzi používateľským rozhraním a aplikačnou logikou. Jej úlohou je kontrolovať korektnosť údajov, filtrovať nevhodné požiadavky a zabezpečiť bezpečný a spoľahlivý prenos medzi ostatnými vrstvami. Táto vrstva podporuje modularitu a uľahčuje udržiavateľnosť systému.

**Backendová vrstva (jadro aplikácie):** Obsahuje hlavnú logiku aplikácie – realizuje výpočty, spracováva dáta, komunikuje s databázou a externými systémami. V tejto vrstve sú implementované dátové modely, výpočtové moduly a funkcionalita zabezpečujúca konzistentné fungovanie celej aplikácie.

Táto viacvrstvová architektúra zabezpečuje nielen čisté oddelenie zodpovedností, ale aj lepšiu bezpečnosť, testovateľnosť a flexibilitu pri rozvoji aplikácie.

# 

# Použité knižnice

Aplikácia využíva viacero moderných knižníc a komponentov, ktoré výrazne uľahčujú vývoj, zvyšujú čitateľnosť kódu a zabezpečujú moderný a konzistentný používateľský zážitok.

**Android Framework & Jetpack**

* **AndroidX Activity** – správa životného cyklu aktivít v súlade s modernými architektonickými odporúčaniami.
* **Jetpack Compose** – deklaratívny nástroj na tvorbu používateľského rozhrania v prostredí Kotlin.
* **Compose Navigation** – komponent pre navigáciu medzi obrazovkami v Compose prostredí.
* **AndroidX ViewModel** – uchovávanie dát medzi zmenami konfigurácie (napr. rotácia obrazovky).
* **AndroidX Lifecycle** – umožňuje reagovať na zmeny životného cyklu komponentov, čím zvyšuje efektivitu správy zdrojov.
* **Material Design 3** – moderné komponenty a vizuálne štýly pre konzistentný dizajn podľa Google zásad.

**Kotlin Libraries**

* **Kotlin Coroutines** – efektívne riadenie asynchrónnych úloh a vlákien.
* **Kotlin StateFlow** – reaktívny nástroj na sledovanie a riadenie stavov v rámci aplikácie.
* **Kotlin DataFrame** – knižnica na prácu s tabuľkovými dátami v štýle podobnom Pandas.

**Networking & Data**

* **Retrofit** – REST klient pre efektívnu komunikáciu s externými API.
* **Room Database** – moderný ORM nástroj na ukladanie a správu lokálnych údajov vo forme relačnej databázy.

**Vizualizácia**

* **MPAndroidChart** – vizualizácia dát prostredníctvom grafov.
  + **LineChart**, **LineData**, **LineDataSet** – komponenty pre vykresľovanie čiarových grafov vrátane konfigurácie dátových množín a ich vlastností.

**Utilities**

* **Android Utils** – pomocné nástroje na spracovanie vzorov, logovanie a zjednodušenie opakujúcich sa operácií.
* **Android Graphics** – nástroje na prácu s farbami a grafickými prvkami.

# Používateľský scenár (Usage Flow)

* **Prihlásenie alebo registrácia používateľa**

Používateľ sa prihlási alebo vytvorí nový účet.

* **Zobrazenie hlavných dát na dashboarde**

Po prihlásení sa zobrazí hlavná obrazovka s trhovými dátami a vývojom cien akcií.

* **Analýza akcií pomocou nástrojov aplikácie**

Aproximačné modely – zobrazenie priebehu aproximácie vrátane MSE (Mean Squared Error).

Klzavý priemer (moving average) – aplikácia filtrovania s vlastnými parametrami (napr. veľkosť okna).

Autoregresívne modely (AR) – generovanie predikcií budúceho vývoja cien na základe historických dát.

* **Nastavenie cenových upozornení**

Používateľ si môže nastaviť alerty na sledované akcie podľa vlastných kritérií.

* **Prijímanie notifikácií**

Pri splnení podmienok alertu používateľ dostane notifikáciu priamo v aplikácii.

# Zoznam zdrojov pri implementácií

Github Copilot: https://github.com/features/copilot

Geeksforgeeks: https://www.geeksforgeeks.org/

Kotlin dokumentácia: <https://kotlinlang.org/>

Android developers: <https://developer.android.com/kotlin>

# Zoznam zdrojov

!!Niektoré časti textu boli preformulované pomocou umelej inteligencie!!

[Kotlin/Native libraries | Kotlin Documentation](https://kotlinlang.org/docs/native-libraries.html)