ŽILINSKÁ UNIVERZITA V ŽILINE
Fakulta riadenia
a informatiky

Semestrálna práca z predmetu *vývoj aplikácií pre mobilné zariadenia*

# <CODE FLOW>

**Vypracoval**: <Jakub Gály>

**Študijná skupina**: <5ZYS32>

**Akademický rok:** <2024/2025>

V Žiline dňa <17.5.2025>

# Obsah

# Úvod

Moja navrhovaná aplikácia nesie názov CODE FLOW, ide o aplikáciu, ktorá vizualizuje vykonávanie kódu krok po kroku. No nie je to debugger v pravom slova zmysle.

Žijeme v digitálnej dobe, a preto sa programovanie stáva čoraz viac nevyhnutné v rôznych oblastiach profesie. Dokonca už aj na základných školách majú deti možnosť začať sa učiť túto zručnosť. Programovanie sa vyučuje na stredných IT školách, stredných elektrotechnických školách, školách venujúcich sa automatizácii, ba dokonca aj na gymnáziách. Programovanie je tvorivá činnosť, vďaka ktorej môžeme realizovať naše nápady, no pre veľké množstvo ľudí nie je až také pochopiteľné – aspoň zo začiatku.

Tento problém sa budem snažiť riešiť v mojej aplikácii. Aplikácia CODE FLOW bude komentovať ľudskou rečou vykonávanie kódu krok po kroku. Teda napríklad: "Vytvorila sa premenná auto, jej typ je int, jej počiatočná hodnota je 5. Vstupujeme do cyklu while. Použila sa druhá vetva IF podmienky..." Tieto komentáre budú generované v poradí, v akom sa kód vykonáva. Bude možné sa aj spätne vracať v týchto komentároch. Napríklad ak sa chceme pozrieť na krok 5, dostaneme k nemu príslušný komentár.

Teda aplikácia má nájsť svoje publikum hlavne medzi ľuďmi, ktorí sa budú chcieť učiť programovať, no samozrejme aj medzi skúsenejšími programátormi, ktorí budú chcieť pochopiť nejaký blok kódu.

Aplikácia bude zo začiatku schopná skomentovať kód naprogramovaný v jazyku Java. Bude mať schopnosť komentovať základné vytváranie premenných, spracovanie výrazov, cykly, podmienky, výpisy do terminálu – do budúcna bude škálovateľná.

# Prehľad podobných aplikácií

Aj keď aplikácia, ktorú sa zamýšľam vytvoriť, pravdepodobne na trhu priamo neexistuje, existujú aplikácie na iný účel súvisiaci s programovaním. Nižšie uvádzam prehľad troch podobných aplikácií vrátane ich výhod a nevýhod.

**Code Editor - Compiler & IDE:**

Aplikácia, ktorá poskytuje základné prvky pre vývoj kódu na mobilnej platforme. Obsahuje mobilný editor a kompilátor s podporou viacerých jazykov.

*Výhody:*

- Podporuje rôzne programovacie jazyky
- Funguje offline
- Umožňuje testovať kód priamo v aplikácii Nevýhody:

*Nevýhody*

- Nie je zameraný na výučbu programovania
- Používateľ sa musí orientovať v chybových hláseniach a ladenie nie je jednoduché pre začiatočníkov

- Neponúka vizualizáciu kódu krok po kroku


*Obrázok 1 Code Editor - Compiler & IDE:*

**Mimo**

Mimo je aplikácia zameraná na programátorov začiatočníkov, ale aj mierne pokročilejších, ktorí si chcú zlepšiť svoje zručnosti. Ponúka krátke interaktívne cvičenia, ktoré sa týkajú hlavne jazykov Python. HTML, JavaScript.

*Výhody*

- Krátke a zábavné lekcie, po ktorých dokončení získa užívateľ body
- Flexibilita – možnosť učiť sa kedykoľvek a kdekoľvek, stačí mať so sebou svoje mobilné zariadenie

*Nevýhody*

- Aplikácia sa zameriava na základy a nie je veľmi vhodná pre výrazne pokročilých užívateľov
- Niektoré funkcie aplikácie vyžadujú predplatné
- Neponúka detailnú vizualizáciu vykonávania kódu krok po kroku.


*Obrázok 2 Mimo*

**SoloLearn**

Je to aplikácia podobná aplikácii Mimo – taktiež slúži na výuku. Je určená hlavne pre začiatočníkov. Podporuje programovacie jazyky ako: C++, Java, Python, JavaScript a ďalšie iné.

*Výhody*

- Zahŕňa viac programovacích jazykov
- Interaktívne kvízy a lekcie
- Možnosť zapojenia sa do programátorskej komunity – zdieľať svoj obsah

*Nevýhody*

- Aplikácia nie je veľmi vhodná pre pokročilejších programátorov
- Nedisponuje detailnou vizualizáciou a komentovaním krok po kroku behu programu


*Obrázok 3 SoloLearn*

Po dôkladnom prehľadaní obchodu Play som dospel k záveru, že aplikácia ako tá moja ešte nie je dostupná. Samozrejme, nemám šancu prekontrolovať všetky aplikácie, ktoré nejako súvisia s programovaním, a zistiť, či sa tam náhodou nenachádza taká aplikácia s funkcionalitou, akú zamýšľam. Taktiež som nebol schopný nájsť nejaký odborný článok, ktorý by sa svojím zámerom približoval k mojej myšlienke.

Ako poslednú možnosť som skúsil požiadať AI, aby mi vyhľadala aplikáciu v takom štýle, aký zamýšľam – tiež mi potvrdila, že nič nenašla.

## Analýza navrhovanej aplikácie

Aplikácia bude obsahovať vzorové časti kódu, ku ktorým si môže používateľ vypísať výklad. Bude si ho môcť odkrokovať postupne, vrátiť sa dozadu, keď treba, skočiť na výklad ku konkrétnemu kroku programu alebo si vypísať všetky komentáre k jednotlivým krokom kódu naraz a potom ich prechádzať podľa potreby. Ku každému komentáru bude zvýraznená aj oblasť v kóde, ktorej prináleží.

Rovnako ako krokovanie vzorových kódov, bude mať používateľ možnosť vložiť si aj svoj vlastný kód a ten si vizualizovať. Jediná podmienka bude, že zatiaľ kód nemôže byť príliš komplexný – nemôže obsahovať triedy, funkcie atď. (rozšírenie do budúcna).

## Návrh architektúry aplikácie

**Parser**

Keďže aplikácia bude schopná rozpoznávať a interpretovať Java kód, bude potrebné tento kód nejakým spôsobom spracovať – na to bude slúžiť parser. Rozdelí kód podľa vhodných oddeľovačov. Pre každý príkaz vyhodnotí jeho príslušnosť k rôznym častiam kódu, či sa jedná o for cyklus, výraz alebo niečo iné.

**Vyhodnocovač výrazov**

Parser síce bude schopný vhodne rozdeliť kód a následne ho „označkovať“, no nebude schopný vykonávať matematické operácie. Na to použijem nejakú externú knižnicu, ktorá bude schopná vykonávať matematické výpočty.

**Zoznam príkazov a zoznam premenných**

Zoznam premenných bude dátová štruktúra, ktorá bude uchovávať jednotlivé premenné a ich hodnoty – toto bude veľmi užitočné, keď nejaká funkcia z nejakej externej knižnice bude vyhodnocovať výraz, pretože bude potrebovať jednotlivé premenné a ich hodnoty, ktoré sa vo výraze vyskytnú.

Zoznam príkazov už bude obsahovať jednotlivé kroky v správnom poradí a k nim príslušný komentár, ktorý ich bude vysvetľovať.

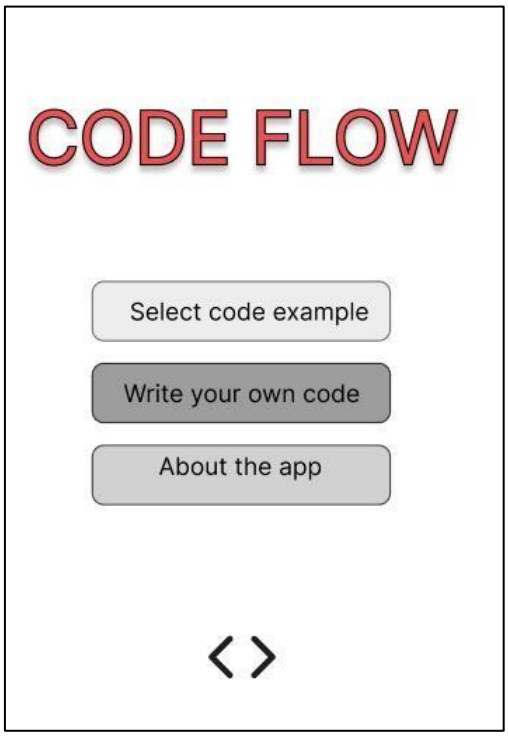
**Databáza ukážkových kódov**

Nejaký List, ktorý bude obsahovať surové kódy, ktoré ešte neprebehli parserom. Bude slúžiť na testovanie aplikácie na rôznych kódoch.

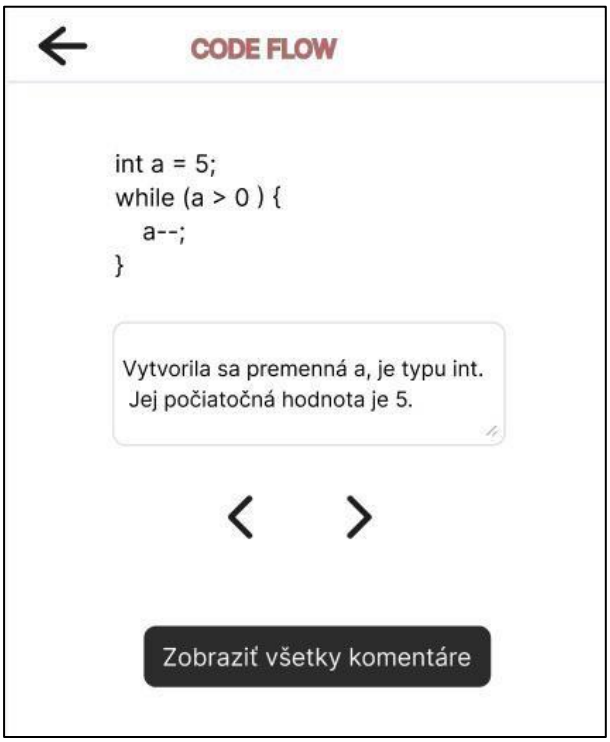
**UI**

- Užívateľské rozhranie bude disponovať oknom pre zdrojový kód
- Tlačidlá, ktoré budú spúšťať komentované kroky ku kódu
- Obrazovka, kde budú tieto správy/komentáre popísané v správnom poradí + k nim bude prislúchať poradové číslo
- Komponent umožňujúci vrátiť sa na dané číslo kroku a ku nemu príslušný komentár
- Ešte premyslím...

## Návrh vzhľadu obrazoviek




*Obrázok 4 CODE FLOW - Screen 2*




*Obrázok 5 CODE FLOW - Screen 1*

Uvítacia obrazovka reprezentovaná obrázkom *CODE FLOW – Screen 1* bude slúžiť ako hlavná navigačná časť aplikácie.

Bude obsahovať:

- Nejakú peknú grafiku, ktorú ešte domyslím
- Navigačné tlačidlá, ktoré budú vykonávať nejaké funkcie, prejavujúce sa buď ako zmena obrazovky, alebo nejaká iná činnosť

Druhý návrh obrazovky sa týka už samotného (*CODE FLOW – Screen 2)* jadra aplikácie. Tu sa bude vykonávať krokovanie programu spolu s príslušným komentárom. Bude tu možnosť

prezrieť si všetky komentáre alebo sa postupne medzi nimi preklikávať, či už dopredu alebo spätne. Bude obsahovať:

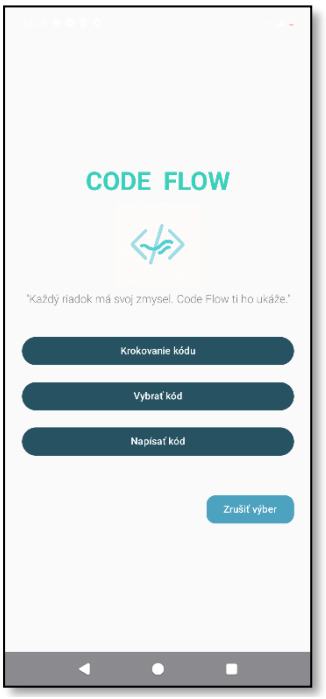
- Šípky na preklikávanie medzi komentármi
- Tlačidlo možnosti zobrazenia všetky komentárov • Šípku späť, ktorá nás vráti na hlavnú úvodnú obrazovku
- Ešte domyslím...

## Finálna verzia aplikácie

Finálna verzia aplikácie zodpovedá naplánovanej schéme. Ako dodatok k základným prvkom toku programu som pridal aj príkazy: *break*, *continue.* Teda aplikácia rozoznáva príkazy výpisu do terminálu, podmienky vetvenia (nie swich), cykly – do while, while, for -, tvorby premenných a príkazy vyhodnocovania výrazov + break, continue.

### Štruktúra aplikácie

#### Hlavná obrazovka

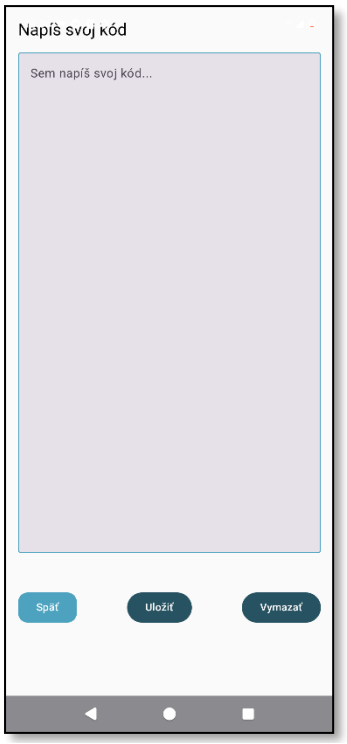



*Obrázok 6 Úvodná obrazovka - FINAL*

Obrazovka slúži ako hlavný spojovací bod medzi ostatnými obrazovkami. Obsahuje názov aplikácie, citát a niekoľko tlačidiel, vďaka ktorým je možné prechádzať do iných obrazoviek.

Celá obrazovka je postavená na Layoute Column, v ktorom sú rozmiestnené jednotlivé prvky.

### Obrazovka pre písanie kódu




Táto časť aplikácie slúži na písanie svojho vlastného kódu, ktorý sa následne bude krokovať. Je možné napísať ľubovoľne dlhý kód, avšak je potrebné dodrať kritéria súvisiace s podporou aplikácie pre dané príkazy.

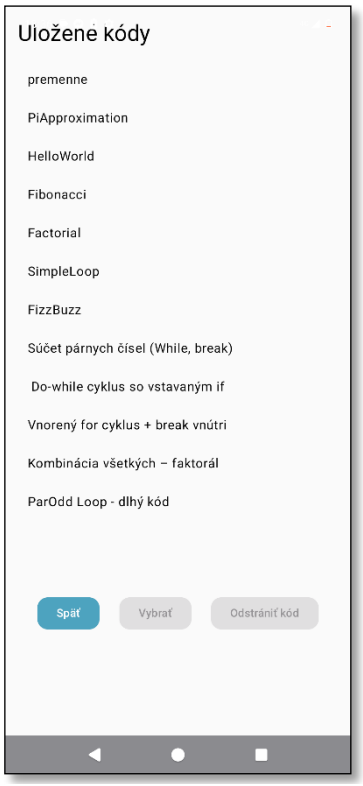
Aplikácie neobsahuje syntax kontrolu, z toho dôvodu je kód potrebné dôkladne prekontrolovať pred jeho finálnym stavom.

Ďalej sú tu tlačidlá *Späť*, ktoré nás naviguje naspäť na úvodnú obrazovku, *Uložiť,* pomocou ktorého vieme kód uložiť do databázy kódov a následne ho krokovať, *Vymazať,* ktoré vynuluje textové pole.

Ako aj pri úvodnej obrazovke aj, tu je hlavným prvkom rozloženia Column.

*Obrázok 7 Obrazovka písania kódu - FINAL*

### Obrazovka pre zvolenie kódu na krokovanie



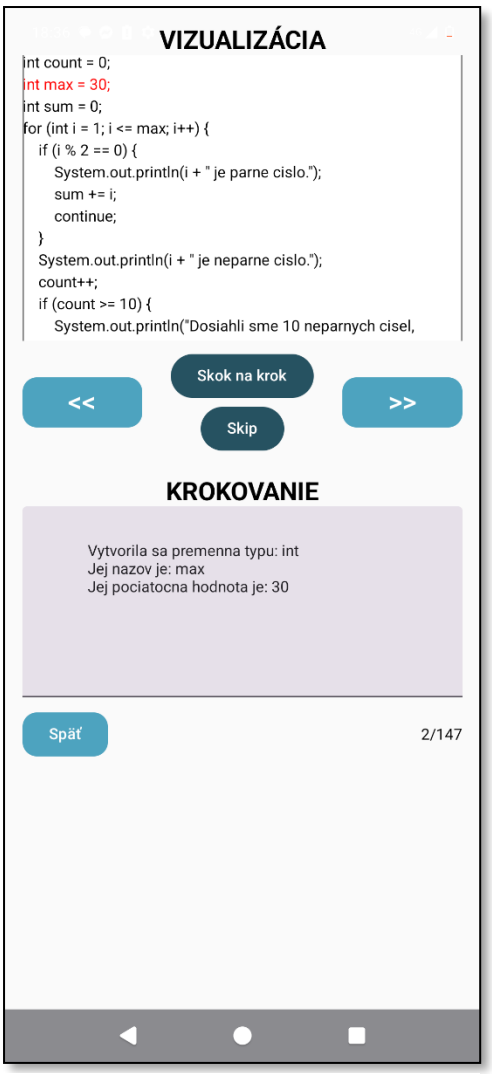

V tejto obrazovke sú zobrazené príslušné kódy nachádzajúce sa v databázovom systéme aplikácie. Na to aby bolo možné správne narábať s touto aplikácie, je potrebné zvoliť si príslušný kód pre krokovanie pomocou tlačidla *Vybrať*.

Je tu aj možnosť odstraňovania kódov z databázy, na to slúži tlačidlo: *Odstrániť Kód.* Tlačidlo: *spať* nás vráti na úvodnú obrazovku

Koreňový layout je Column.

*Obrázok 8 Obrazovka uložené kódy - FINAL*

**Obrazovka – vizualizácia**




*Obrázok 9 Obrazovka vizualizácia - FINAL*

Najdôležitejšia časť tejto aplikácie, obsahuje hlavnú myšlienku – komentovať kód ľudskou rečou krok za krokom.

Po zvolení kódu na krokovanie je možné tento kód vizualizovať. Kód je zobrazený v textovom poli VIZUALIZÁCIA a jednotlivé kroky sú zobrazované v textovom poli KROKOVANIE.

Na spodnej časti obrazovky sa nachádza tlačidlo späť na úvod a aktuálne poradie príkazu spolu s celkovým počtom príkazov (viď. napravo)

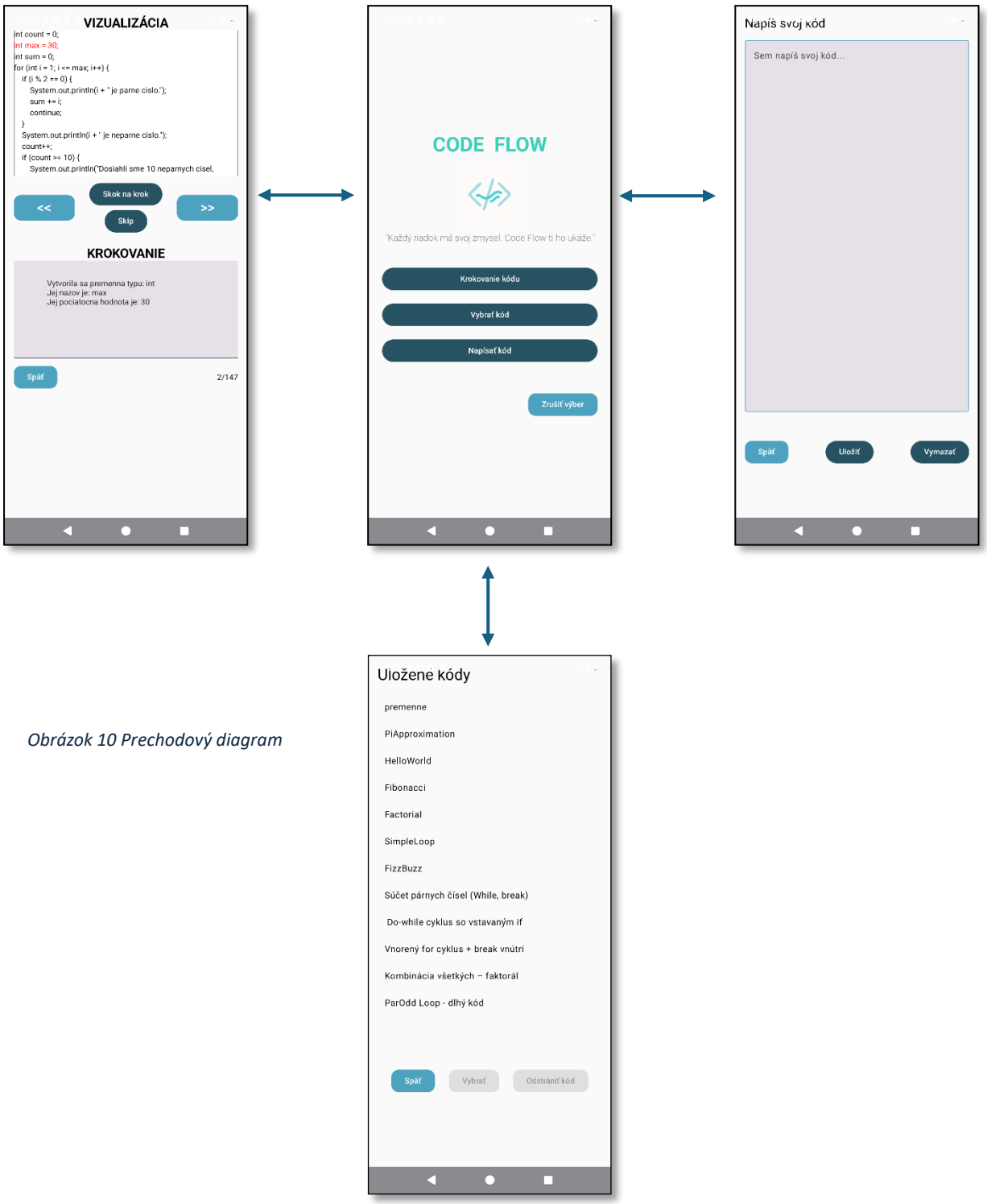
Aktuálne vykonávaný príkaz je označený červenou farbou. Čo sa týka vnorení, či už do cyklov alebo vetiev – je to vyriešené spôsobom, že každé ďalšie vnorenie do nejakého kódového bloku nadobudne novú farbu (viacnásobné vnorenia).

V kóde sa je možné krokovať šípkami doprava a doľava. Taktiež je možné skočiť na ľubovoľný krok programu – v rámci počtu krokov - pomocou tlačidla *Skok na krok*.

Tlačidlo *Skip* slúži na preskočenie aktuálneho cyklu – ak sa aplikácia v žiadnom nenachádza, tak jednoducho prejde na ďalší krok.

Koreňový layout je Column.

## Prechodový diagram




*Obrázok 10 Prechodový diagram*

**Stavový diagram**

*[Start]*
↓
*[Načítanie kódu z databázy]*
↓
*[Parsovanie kódu]*
↓
*[Pripravené na krokovanie]*
↓
*[Krokovanie vpred / vzad / skok / skip]*
↓
*[Koniec]*

**Popis implementácie**

**Parser**

Parser v aplikácii Code Flow bol navrhnutý tak, aby vedel spracovať jednoduchý Java-like kód, analyzovať jednotlivé príkazy a vizualizovať ich krokové vykonanie. Je postavený na princípe rekurzívneho spracovania blokov kódu, pričom vychádza z objektovo orientovaného návrhu s využitím dedičnosti.

1. *Rozdelenie vstupného kódu*

Na začiatku sa vstupný kód rozdelí podľa oddelovačov {, }, ;, ktoré predstavujú hranice spracovávaných reťazcov. Každý reťazec je potom spracovaný samostatne, pričom sa zachováva vnorená štruktúra (napr. podmienky v cykle atď.).

2. *Rekurzívna analýza kódu*

Spracovanie je riešené rekurzívne. Základom je trieda KodovyBlok, ktorá definuje spoločné správanie pre všetky bloky kódu (napr. for-cyklus, if-vetu, hlavný blok a pod.). Z nej dedí špecifická trieda HlavnyBlok, ktorá reprezentuje hlavný kontext kódu a obsahuje zoznam všetkých príkazov, ktoré je potrebné analyzovať a následne krokovať.

Pri spracovaní každého príkazu sa zisťuje, či ide o: premennú, výraz, podmienku, cyklus, break...

Ak parser narazí na príkaz, ktorý je sám o sebe blokom (napr. if, while...), vytvorí nový potomok triedy KodovyBlok (napr. IfBlok, WhileBlok) a rekurzívne do neho vstúpi – spracováva jeho príkazy rovnako ako v hlavnom bloku.

Týmto spôsobom parser umožňuje vnáranie sa do blokov ľubovoľnej hĺbky, čo je základná vlastnosť tohto riešenia.

3. *Vyčlenenie typu príkazu*

Každý načítaný riadok sa pred samotným spracovaním analyzuje – určí sa, akého typu príkaz je. Rozpoznávanie prebieha na základe kľúčových slov a syntaktických znakov.

**Vyhodnocovanie výrazov**

Vyhodnocovanie matematických výrazov v aplikácii je zabezpečené pomocou knižnice tretej strany JexlBuilder, ktorá pochádza z knižnice Apache Commons JEXL. Táto knižnica umožňuje jednoducho vypočítať výrazy zadané ako text (napr. a + b * 2) bez potreby ručného vytvárania parsera alebo výpočtového systému. Vďaka tomu dokáže aplikácia dynamicky vyhodnocovať aj zložitejšie výrazy počas krokovania kódu.

**Obrazovky**

Moja aplikácia Code Flow obsahuje štyri hlavné obrazovky, medzi ktorými sa používateľ môže plynule pohybovať vďaka Jetpack Navigation Compose komponentu.

*Úvodná obrazovka*

Táto obrazovka slúži ako vstupný bod aplikácie, kde používateľ môže prejsť na ďalšie časti.

*Obrazovka na napísanie kódu*

Umožňuje používateľovi zadať vlastný kód, ktorý chce analyzovať alebo vizualizovať.

*Obrazovka na výber kódu*

Tu si používateľ môže vybrať z preddefinovaných kódov uložených v databáze.

*Obrazovka pre vizualizáciu kódu*

Zobrazuje krok po kroku vykonávanie zvoleného kódu, vrátane zvýraznenia aktuálneho riadku a kódového bloku.

**Databáza**

Vo svojej aplikácii Code Flow som implementoval Room, oficiálnu databázovú knižnicu od Googlu, ktorá zjednodušuje prácu s lokálnou SQLite databázou v Android aplikáciách.

Na interakciu s databázou som vytvoril **DAO (Data Access Object)** rozhranie, ktoré obsahuje nasledujúce metódy:

```
@Query("SELECT * FROM kody ORDER BY datumUlozenia DESC")
suspend fun getVsetkyKody(): List<Kod>

@Insert(onConflict = OnConflictStrategy.REPLACE)
suspend fun vlozKod(kod: Kod)

@Query("DELETE FROM kody WHERE nazov = :nazov")
suspend fun odstranKod(nazov: String)

@Query("SELECT * FROM kody WHERE nazov = :nazov")
suspend fun dajMiKodPodlaNazvu(nazov: String): Kod?
```

*Obrázok 11 Databázové príkazy*

Tieto metódy umožňujú:

- Získať všetky uložené kódy zoradené podľa dátumu uloženia.
- Vložiť nový kód do databázy; ak už existuje záznam s rovnakým názvom, bude nahradený.
- Odstrániť kód z databázy na základe jeho názvu.
- Získať konkrétny kód podľa jeho názvu.

Použitím anotácií ako @Query a @Insert Room automaticky generuje potrebný kód na vykonanie týchto operácií, čím sa eliminuje potreba písania manuálnych SQL dotazov.

Po inicializácii databázy som zabezpečil jej predvyplnenie ukážkovými kódmi, ktoré slúžia ako príklady pre vizualizáciu

**View Model**

Vo svojej aplikácii Code Flow som vytvoril triedu KniznicaKodov, ktorá slúži ako centrálne miesto na prácu s databázou. Aby som zabezpečil, že sa nevytvára nová inštancia tejto triedy pri každom prechode medzi obrazovkami, implementoval som ju ako ViewModel. Týmto spôsobom sa KniznicaKodov zdieľa medzi všetkými obrazovkami aplikácie, čo zabraňuje opätovnému vytváraniu databázy a zvyšuje efektivitu aplikácie.

Rovnako som implementoval triedu KrokovanieKodov ako ViewModel, ktorý je priamo prepojený s composable obrazovkou zodpovedou za krokovanie kódu. Týmto spôsobom môže KrokovanieKodu ovplyvňovať stav a správanie tejto obrazovky, čo umožňuje dynamickú a interaktívnu vizualizáciu kódu.

**Zablokovanie otáčania obrazovky**

Zablokoval som otáčanie obrazovky v aplikácii Code Flow, pretože pri zmene orientácie by sa mohol zmeniť aj spôsob zobrazenia textu, ktorý sa krok po kroku analyzuje a vizualizuje. Tým by sa používateľovi mohol narušiť sled kódu, čo by mohlo viesť k nejasnostiam a zníženiu prehľadnosti aplikácie.

Toto rozhodnutie som prijal s cieľom zabezpečiť konzistentný a plynulý zážitok pri práci s aplikáciou, kde je dôležité, aby sa používateľ mohol sústrediť na postupné vykonávanie kódu bez rušivých zmien v rozložení obrazovky.

**Logo aplikácie**

Z dôvodu úplnosti riešenia som zvážil, že by bolo vhodné vytvoriť nejaké pekné logo pre moju aplikáciu – je to to isté logo, ktoré je zobrazené aj na úvodnej obrazovke. Nachádza sa v zložke resourec.

*Obrázok 12 Logo aplikácie*

# Zoznam zdrojov

**Code Editor - Compiler & IDE:** Code Editor - Compiler & IDE – Aplikace na Google Play

**Mimo:** Mimo: Naučte se programovat – Aplikace na Google Play

**SoloLearn:** Sololearn: Learn to code – Aplikace na Google Play