ŽILINSKÁ UNIVERZITA V ŽILINE
Fakulta riadenia
a informatiky

Semestrálna práca z predmetu

vývoj aplikácií pre mobilné zariadenia

<LIGHT CALCULATOR>

Vypracoval: Tomáš Hitka

Študijná skupina: 5ZYR21

Akademický rok: 2024/2025

V Žiline dňa 2.4.2025

Obsah

Úvod 2
Prehľad podobných aplikácií 3
Analýza navrhovanej aplikácie 4
Návrh architektúry aplikácie 5
Návrh vzhľadu obrazoviek 5
Popis implementácie 7
O aplikacii 8

Úvod

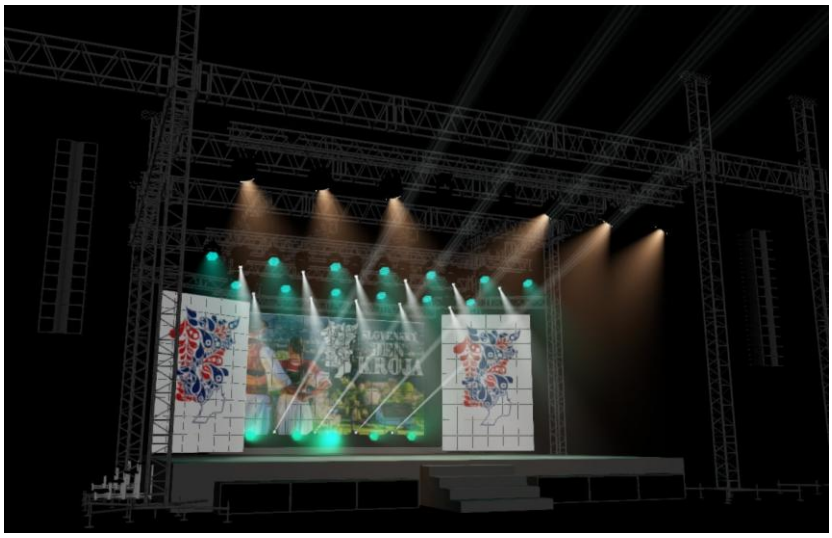
V aktuálnej dobe sa v oblasti osvetlenia kultúrnych podujatí často stretávam s problémom efektívneho rozdelenia svetiel, pokiaľ ide o zaťaženie elektrickej siete. Rovnako sa stretávam aj s adresovaním svetiel, aby ich bolo možné následne ovládať cez svetelný pult. Na každej akcii, kde sa takáto situácia vyskytne, sa rieši pomocou „papiera“ a „klasickej mobilnej kalkulačky“.

Toto ma motivovalo k vytvoreniu jednoduchej mobilnej aplikácie, ktorá by používateľovi umožnila zadať počet svetiel a aplikácia by všetky potrebné výpočty vykonala automaticky. Táto aplikácia by mohla výrazne zjednodušiť a hlavne skrátiť čas potrebný na výpočty týchto hodnôt.

Prehľad podobných aplikácií

Keďže ide o pomerne špecifickú aplikáciu, nepodarilo sa mi nájsť mobilnú aplikáciu s podobnou funkcionalitou. Existuje však desktopová aplikácia s názvom **WYSIWYG**, ktorá síce nie je primárne zameraná na túto problematiku, no dokáže vykonať výpočty, ktoré by mala zvládnuť aj moja navrhovaná mobilná aplikácia.

WYSIWYG je určený predovšetkým na **vizualizáciu celého pódia**, vrátane **hliníkovej konštrukcie (tzv. „supportu“) a samotného pódia (napr. samotných dosiek „Nivtecov“) a aj zvukového systému**. Zároveň poskytuje aj podrobné informácie o **rozmeroch** danej konštrukcie. Po umiestnení požadovaných svetiel na konštrukciu vo vizualizéri je aplikácia schopná vypočítať **záťaž, rozloženie a ďalšie technické parametre**.



Obrázok 1 - Vizualizácia Dňa Kroja v Banskej Bystrici 2024 v programe WYSIWYG

Obrázok 3 - Vizualizácia Dňa Kroja v Banskej Bystrici 2024 v programe WYSIWYG

Obrázok 2 - Vizualizácia Dňa Kroja v Banskej Bystrici 2024 v programe WYSIWYG

Analýza navrhovanej aplikácie

Používateľské role

V aplikácii sa nachádza jedna hlavná rola:

- **Osvetľovač (používateľ)** – osoba, ktorá navrhuje svetelnú zostavu pre podujatie a potrebuje výpočet technických parametrov (fázy, universy, adresy).

Prípady použitia

1. **Zobrazenie úvodnej obrazovky**
    - Po spustení aplikácie sa zobrazí úvodná obrazovka s možnosťou zobrazenia základných informácií o aplikácii ⓘ a vstupu do aplikácie.
2. **Výber typu operácie**
    - Používateľ si môže vybrať medzi:
        - Vytvorením nového podujatia
        - Načítaním existujúceho podujatia z uložených presetov
3. **Výber svetiel**
    - Používateľ si vyberá značku svetla, následne model, počet svetiel a požadovaný mód.
    - Každý výber sa pridáva do zoznamu svetiel.
    - Po potvrdení všetkých výberov aplikácia spracuje výpočet.
4. **Zobrazenie výpočtov**
    - Aplikácia zobrazí výsledky:
        - Potrebný počet fáz a universov
        - Rozdelenie svetiel medzi fázy
        - Adresy pre jednotlivé svetlá podľa zvoleného módu
5. **Zobrazenie nápovedy**
    - Na každej obrazovke je dostupná informačná ikona ⓘ, po kliknutí sa zobrazí stručná pomoc/návod pre aktuálnu obrazovku.

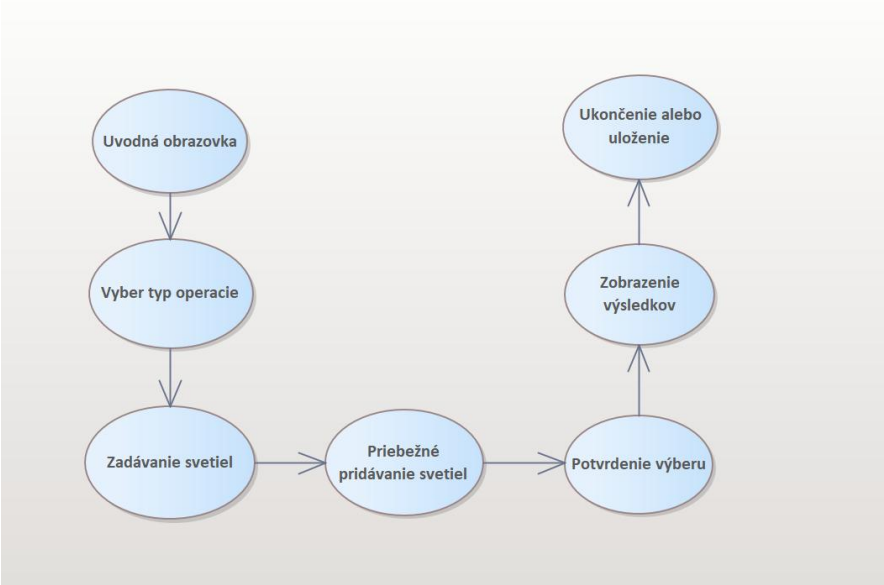
Mimo funkčné požiadavky

- **Rýchlosť spracovania** – výpočty musia byť vykonané okamžite po potvrdení výberov.
- **Offline dostupnosť** – aplikácia by mala fungovať aj bez internetového pripojenia.
- **Rozšíriteľnosť** – databázu svetiel (značky, modely, módy) by malo byť možné v budúcnosti jednoducho aktualizovať alebo rozšíriť.
- **Platforma** – aplikácia by mohla fungovať aj na iOS zariadeniach.

Návrh architektúry aplikácie

Dátový model

- **Svetlá**
  - Značka (napr. Robe, GLP, Clay Paky)
  - Model (napr. Pointe, MegaPointe, Impression x4L)
  - Mód (počet DMX kanálov)
  - Počet kusov
- **Podujatie**
  - Názov (voliteľný)
  - Zoznam svetiel použitých na danom podujatí
  - Výsledné výpočty: počet fáz, počet universov, DMX adresy, rozdelenie svetiel
- **Preset**
  - Identické dáta ako podujatie, slúži na opätovné načítanie



Obrázok 4 - Use case diagram aplikácie

Návrh vzhľadu obrazoviek

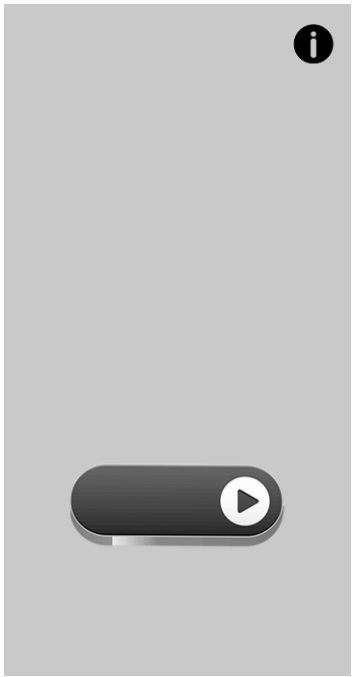
Po spustení aplikácie sa zobrazí úvodná obrazovka (obr. 2). V pravom hornom rohu sa bude nachádzať informačná ikona ⓘ, ktorá po rozkliknutí zobrazí základné informácie o aplikácii. V dolnej časti obrazovky bude umiestnené tlačidlo, pomocou ktorého sa používateľ presunie ďalej do aplikácie.

Na nasledujúcej obrazovke (obr. 3) budú k dispozícii dve tlačidlá, pomocou ktorých si používateľ vyberie, či chce vykonať výpočet pre nové podujatie, alebo načítať údaje z niektorého z predtým uložených podujatí (napr. keď už dané rozloženie svetiel použil). Na základe zvolenej možnosti bude

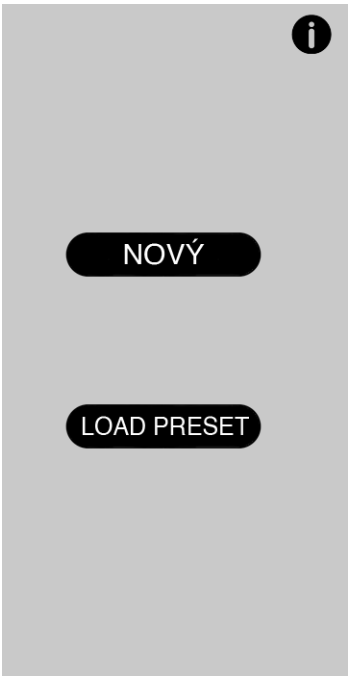
presmerovaný buď na obrazovku s výberom zo zoznamu uložených „presetov“, alebo na obrazovku na výber svetiel. Aj na tejto obrazovke bude v pravom hornom rohu informačná ikona s návodom, ako ďalej postupovať (ⓘ).

Na tretej obrazovke (obr. 4) si používateľ vyberie značku svetla. Po výbere značky sa zobrazia len svetlá príslušnej značky. Následne si zvolí konkrétny model, zadá počet kusov a vyberie požadovaný mód svetla (čím vyšší mód zvolí, tým viac možností bude k dispozícii). Pod výberovým formulárom sa budú nachádzať dve tlačidlá – jedno na potvrdenie výberu, čím sa zvolené svetlá pridajú do zoznamu v spodnej časti obrazovky vo formáte (**POCETx NAZOV_A_TYP_SVETLA)**, pričom sa formulár automaticky vynuluje pre zadanie ďalších svetiel. Druhé tlačidlo slúži na potvrdenie kompletného výberu svetiel pre danú akciu. Po jeho stlačení aplikácia vykoná potrebné výpočty. V pravom hornom rohu opäť nebude chýbať informačná ikona s návodom.

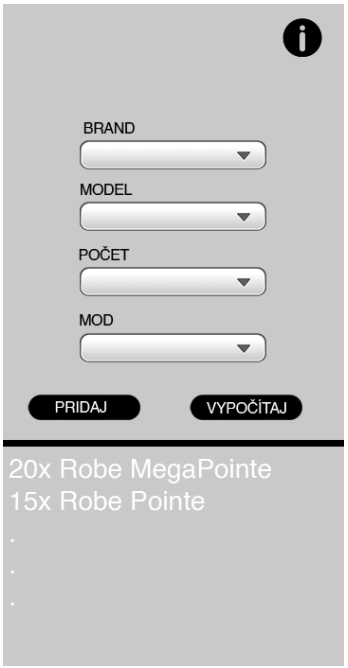
V poslednej časti sa zobrazia výsledky – počet potrebných elektrických fáz a DMX universov na základe zadaných požiadaviek. Aplikácia tiež vypočíta a zobrazí rovnomerné rozdelenie svetiel medzi jednotlivé fázy. Pre správne adresovanie svetiel budú vygenerované aj DMX adresy podľa zvoleného módu. Aj na tejto obrazovke bude dostupná informačná ikona s pomocnými inštrukciami. Túto obrazovku ešte uplne neviem ako nadizajnujem.



Obrázok 5 - Úvodná obrazovka



Obrázok 6 - Obrazovka na výber typu operácie



Obrázok 7 - Obrazovka na výber svetiel

Popis implementácie

Použitie ViewModel a State

V aplikácii je použitý architektonický vzor MVVM (Model-View-ViewModel), ktorý zabezpečuje oddelenie používateľského rozhrania od obchodnej logiky. Trieda CalculationViewModel uchováva stav zoznamu pridaných svetiel (items) pomocou MutableStateFlow, čo umožňuje automatické aktualizácie UI pri zmene dát. LightViewModel slúži na získavanie dát zo vzdialenej databázy (Firebase Firestore), ako sú značky, modely a DMX módy svetiel. V Compose komponentoch sa pozorujú tieto hodnoty pomocou collectAsState(), čo zabezpečuje ich reaktívne správanie.

Práca so súbormi – lokálne úložisko presetov

Presety (prednastavené zostavy svetiel) sa ukladajú a načítavajú lokálne pomocou PresetStorage. Dáta sa serializujú/deserializujú cez knižnicu GSON a ukladajú ako JSON súbory do adresára context.filesDir/presets. Každý preset má meno a zoznam svetiel (AddedLight), ktoré obsahujú všetky potrebné údaje pre výpočty.

Použité UI/UX prvky

Na tvorbu používateľského rozhrania bola použitá knižnica Jetpack Compose, ktorá umožňuje budovanie UI.

Použité komponenty:

- LazyColumn – zobrazenie zoznamu presetov
- AlertDialog – vstup názvu presetu pri ukladaní
- OutlinedTextField – zadávanie počtu svetiel
- ExposedDropdownMenuBox – výber značky, modelu a DMX módu
- Button – tlačidlá pre navigáciu a akcie (pridanie svetla, výpočet, uloženie)
- Text – popisné a výstupné texty
- Farby a pozadia – upravené pre dobrý kontrast (svetlosivá, čierna, tmavosivá)

Navigácia – Jetpack Navigation Compose

Navigácia medzi obrazovkami je realizovaná pomocou knižnice Navigation Compose. Všetky trasy sú definované v NavHost v MainActivity. Obrazovky ako NewCalculationScreen, LoadPresetScreen, CalculationResultScreen a HomeScreen sú prepojené cez NavController.

Špeciálny parameter isFromPreset sa prenáša cez navController.navigate("result?fromPreset=true") a zabezpečuje, že na výstupnej obrazovke sa nezobrazí tlačidlo na uloženie, ak sa otvorí z existujúceho presetu.

Architektúra a štruktúra projektu

Projekt je organizovaný do balíčkov:
- ui.screen – obsahuje všetky obrazovky (Compose komponenty)
- viewmodel – obsahuje CalculationViewModel a LightViewModel
- data – obsahuje dátové triedy AddedLight, Preset a PresetStorage

Použitý architektonický prístup MVVM zabezpečuje:

- Model: dátové triedy a PresetStorage
- ViewModel: uchovávanie a správa stavu, poskytovanie dát
- View: Jetpack Compose komponenty, ktoré pozorujú State a reagujú na zmeny

Použité technológie a knižnice

- Jetpack Compose – UI toolkit
- Navigation Compose – navigácia
- ViewModel + StateFlow – riadenie stavu
- GSON – serializácia JSON (pre ukladanie presetov)
- Firebase Firestore – čítanie údajov o svetlách
- Kotlin Coroutines – asynchrónne načítavanie dát
- Toast – notifikácie používateľovi pri uložení

O aplikacii

Aplikácia bola vyvinutá v jazyku Kotlin s využitím frameworku Jetpack Compose. Jej zložitosť spočíva v počte obrazoviek, prepojení s databázou a vo využití komponentov AndroidX. Všetky texty sú uložené v resource súboroch, čo zaručuje správu lokalizácií a jednoduchú údržbu obsahu.

Aplikácia obsahuje viacero obrazoviek ako napr. úvodná obrazovka, výber nového výpočtu, načítanie presetu, výber parametrov svetiel a výsledková obrazovka. Každá z nich má svoju vlastnú funkcionalitu, čím zodpovedá definícii samostatnej obrazovky. Na výpočet potrebného výkonu a adresovania DMX kanálov sú použité algoritmy, ktoré rozdeľujú svetlá do fáz a DMX vesmírov podľa maximálnych kapacít.

Z AndroidX komponentov aplikácia využíva Navigation Compose na správu prechodov medzi obrazovkami, ViewModel na uchovávanie stavov naprieč recompozíciami a životnými cyklami, ako aj LifeCycle pre správne reakcie na zmeny stavu aplikácie. Tým je zabezpečené oddelenie logiky a používateľského rozhrania, čo zodpovedá architektúre MVVM. ViewModely uchovávajú dáta a logiku, UI sa stará výlučne o zobrazenie.

Aplikácia využíva Firebase Firestore na dynamické načítanie technických údajov o svetlách, čím spĺňa požiadavku na sieťovú komunikáciu. Na lokálne uloženie presetov používateľa je implementovaná práca so súbormi cez `context.filesDir`, čím je zaistená perzistencia užívateľských nastavení. Presety sa ukladajú a načítavajú ako JSON súbory.

Z pohľadu dizajnu sú použité štandardné komponenty Jetpack Compose ako LazyColumn na zoznam presetov, AlertDialog na zadávanie názvov presetov a OutlinedTextField na zadávanie hodnôt. Farby a štýly sú upravené tak, aby bola zabezpečená čitateľnosť a kontrast aj pri použití svetlejšieho pozadia.

Z pohľadu GIT repozitára bola práca vyvíjaná priebežne, pričom sú zachované dobre pomenované commity s logickou štruktúrou zmien. Projekt je rozčlenený do balíčkov ako `ui.screen`, `data`, `viewmodel`, čím je zabezpečená prehľadnosť a udržiavateľnosť. V kóde nie sú duplikácie, sú dodržiavané zásady OOP – napríklad enkapsulácia a rozdelenie zodpovedností medzi triedy.

Celkovo aplikácia napĺňa technické aj dizajnové požiadavky zadania a je implementovaná spôsobom,

ktorý je zrozumiteľný, rozšíriteľný a dobre udržiavateľný.