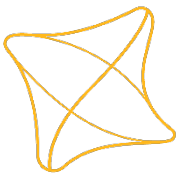


ŽILINSKÁ UNIVERZITA V ŽILINE
Fakulta riadenia
a informatiky

Semestrálna práca z predmetu

Vývoj aplikácií pre mobilné zariadenia

Wordle

Tomáš Danko
5ZYR21

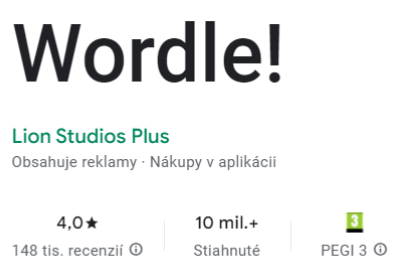
Obsah

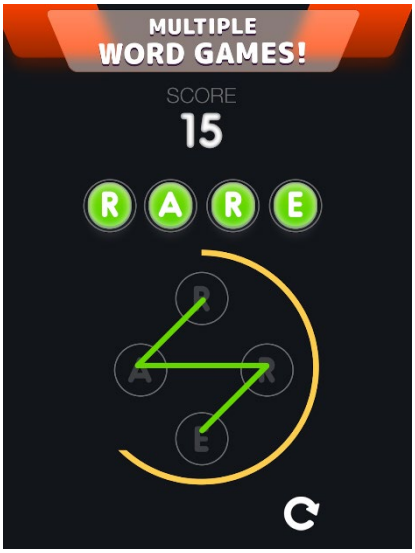
1. Popis a analýza riešeného problému

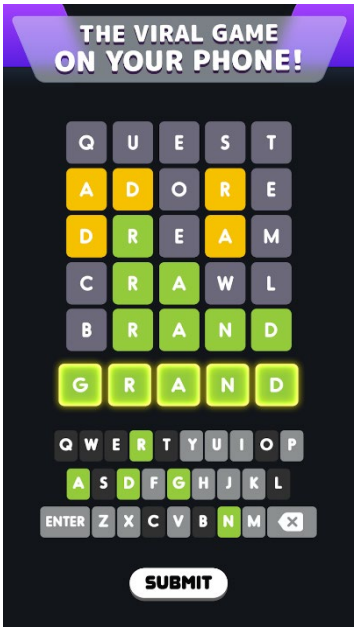
1.1 Špecifikácia zadania, definovanie problému

Jedná sa o aplikáciu, ktorá je sústredená na hru menom Wordle. Aplikácia bude obsahovať niekoľko herných módov a herných náročností. Užívateľovi bude dostupná sekcia „How to play“, teda návod na to, ako hru hrať a vysvetlenie všetkých herných módov a obtiažností. Užívateľ si bude môcť meniť prostredie hry pomocou sekcie „Settings“, kde si vie vybrať štýl a jazyk aplikácie. Takisto bude dostávať upozornenia, pokiaľ ich nevypne. Bude mať možnosť si resetovať všetok svoj progres. Cieľom hry je uhádnuť skryté anglické slovo. Nápovedy bude užívateľ dostávať vo forme vyfarbených písmen, resp. čísel (pri hernom móde „Number“).

1.2 Podobné aplikácie, porovnanie





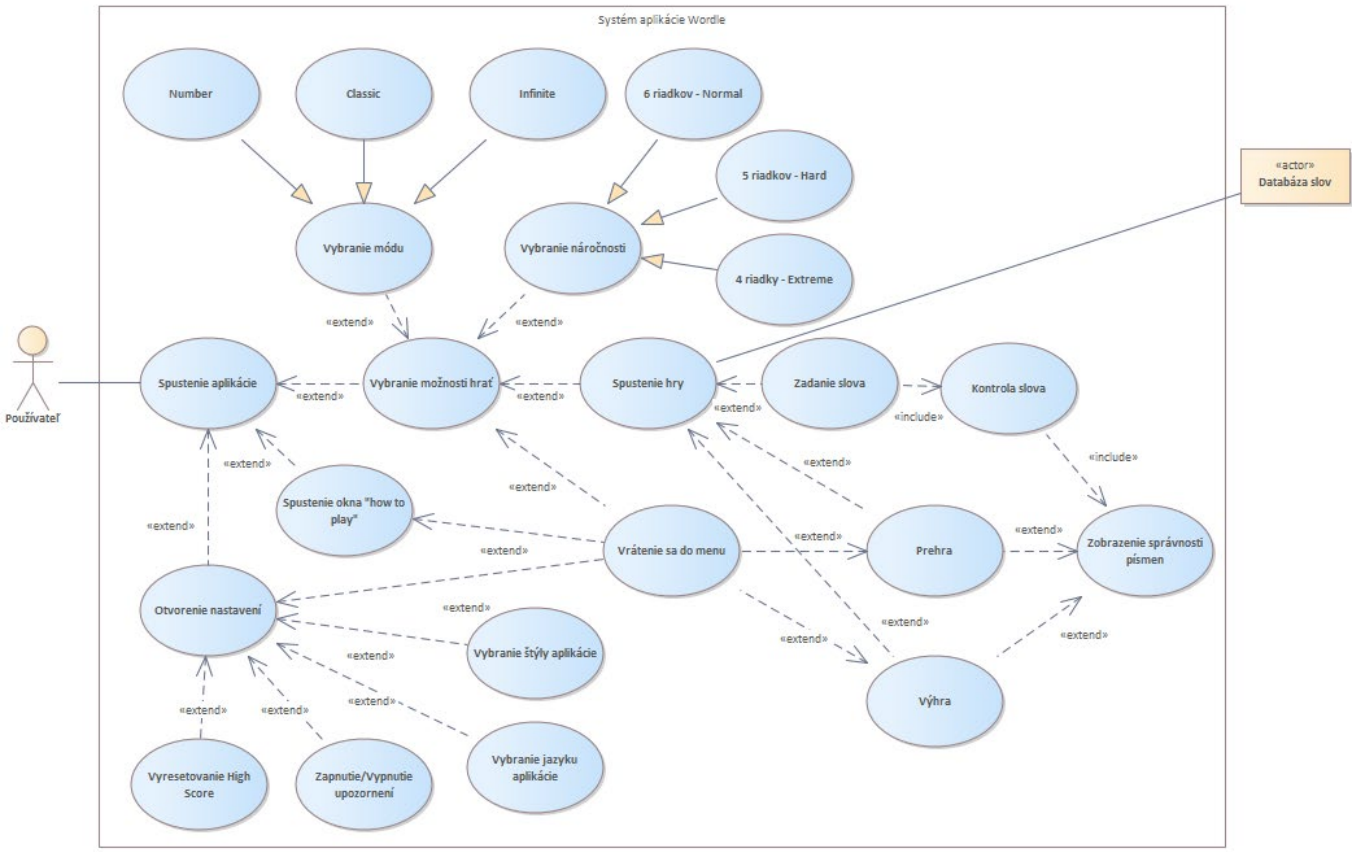


V porovnaní s aplikáciou „Wordle!“, moja aplikácia má výber z viacerých herných módov a obtiažností. Aplikácia „Wordle!“ ale obsahuje viacero slovných hier. Obsahuje množstvo reklám a odomykateľných nápovied za poplatky.

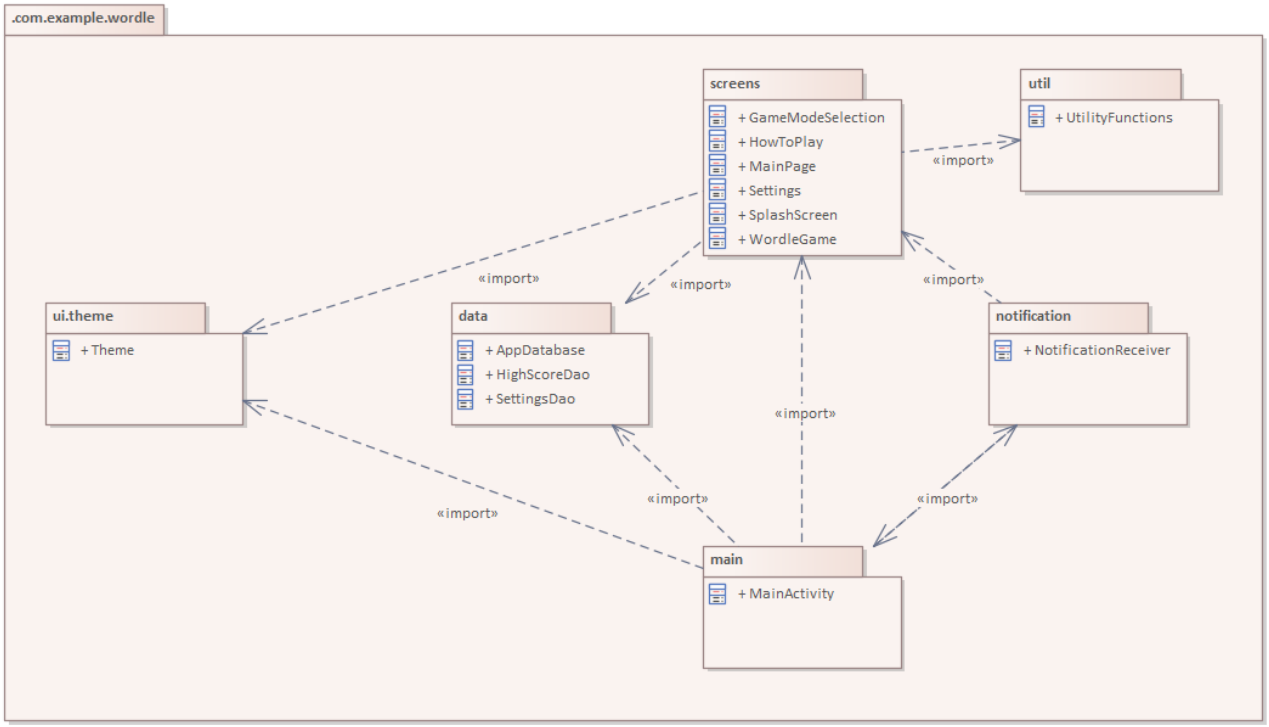
Aplikácia „NYT Games: Word Games & Sudoku“ obsahuje veľký výber slovných a logických hier, vrátane hry „Wordle“, avšak za väčšinu z nich je nutné zaplatiť. Po pár kôl hrania, je nutné počkať celý deň alebo si zaplatiť poplatok, aby mohol hráč hrať ďalej. V mojej aplikácií žiadne obmedzenia nie sú. Dizajn „NYT Games“ je viac farebný a príliš neprehľadný.

2. Návrh riešenia problému

2.1 Diagram prípadov použitia



2.2 Diagram balíčkov



3. Popis implementácie a komponentov

MainActivity pri vytvorení zoberie všetky dáta z databázy vytvorenej cez Room pomocou LifeCycle a Coroutines. Podľa uložených nastavení sa všetky nastavenia nastavia a zakomponuje sa NavHost. NavHost následne vytvorí SplashScreen. Po SplashScreen-e sa zobrazí MainPage. Z MainPage je možné ísť do Settings /HowToPlay/ GameModeSelection vďaka NavHost. V GameModeSelection je použitý RadioButton, ktorým sa dá vybrať obtiažnosť hry a v Settingoch je použitý Switch na zapínanie notifikácií a tmavého módu. V samotnej hre (pri výhre alebo prehre) a v nastaveniach (pri resetovaní HighScore) je použitý Dialog. Pri vypnutí aplikácie sa skontroluje, či v nastaveniach sú zapnuté notifikácie – ak áno, tak sa nastaví notifikácia (o deň) pomocou Context-u a BroadcastReceiver-u.

3.1 Databáza a jej načítanie pri zapnutí aplikácie

```
@Entity(tableName = "settings")
data class Settings(
    @PrimaryKey(autoGenerate = true) var id: Int = 0,
    var isDarkTheme: Boolean = true,
    var selectedLanguage: String = "English",
    var areNotificationsEnabled: Boolean = true
)
RaijinG
@Dao
interface SettingsDao {
    RaijinG
    @Insert(onConflict = OnConflictStrategy.REPLACE)
    fun insertSettings(settings: Settings)

    RaijinG
    @Query("SELECT * FROM settings LIMIT 1")
    fun getSettings(): Settings?
}
```

```
@Entity(tableName = "high_score")
data class HighScore(
    @PrimaryKey(autoGenerate = true) var id: Int = 0,
    val score: Int
)
RaijinG *
@Dao
interface HighScoreDao {
    RaijinG
    @Query("SELECT * FROM high_score ORDER BY score DESC LIMIT 1")
    fun getHighScore(): HighScore?
    RaijinG
    @Insert
    fun insertHighScore(highScore: HighScore)
}
```

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)

    fun updateLocale(language: String) {
        val locale = when (language) {
            "Spanish" -> Locale( language: "es")
            "French" -> Locale( language: "fr")
            "German" -> Locale( language: "de")
            else -> Locale( language: "en")
        }
        Locale.setDefault(locale)
        val config = resources.configuration
        config.setLocale(locale)
        resources.updateConfiguration(config, resources.displayMetrics)
    }

    db = Room.databaseBuilder(
        applicationContext,
        AppDatabase::class.java, name: "wordle-database-v2"
    ).build()

    lifecycleScope.launch { this: CoroutineScope
        currentHighScore = retrieveHighScore()
        val settings = retrieveSettings()
        isDarkTheme = settings.isDarkTheme
        selectedLanguage = settings.selectedLanguage
        updateLocale(selectedLanguage)
        areNotificationsEnabled = settings.areNotificationsEnabled
    }
```

3.2 NavHost

```
NavHost(navController, startDestination = "splashScreen") {
    composable("splashScreen") {
        SplashScreen(navController)
    }
    composable("mainPage") {
        MainPage(
            onPlayClicked = {
                navController.navigate("gameModeSelection")
            },
            onHowToPlayClicked = {
                navController.navigate("howToPlay")
            },
            onSettingsClicked = {
                navController.navigate("settings")
            }
        )
    }
    composable("gameModeSelection") {
        GameModeSelection(
            onGameModeSelected = { gameMode, difficulty ->
                val words = readWordsFromFile(resources)
                navController.navigate("wordDisplay/${words.random()}?gameMode=$gameMode&difficulty=$difficulty") {
                    popUpTo("gameModeSelection") { saveState = true }
                    launchSingleTop = true
                    restoreState = true
                }
            }
        )
    }
    composable("wordDisplay/{word}?gameMode={gameMode}&difficulty={difficulty}&score={score}") { backStackEntry ->
        val word = backStackEntry.arguments?.getString("word") ?: ""
        val gameMode = backStackEntry.arguments?.getString("gameMode") ?: "CLASSIC"
        val difficulty = backStackEntry.arguments?.getString("difficulty") ?: "NORMAL"
        val initialScore = backStackEntry.arguments?.getString("score")?.toIntOrNull() ?: 0
        val highScore = currentHighScore
        WordDisplay(word = word,
            gameMode = gameMode,
            difficulty = difficulty,
            navController = navController,
            initialScore = initialScore,
            highScore = highScore,
            saveHighScore = { score -> saveHighScore(score) }
        )
    }
    composable("howToPlay") {
        HowToPlayDisplay()
    }
```

```
composable("settings") {
    SettingsDisplay(
        isDarkTheme = isDarkTheme,
        onThemeChange = { darkTheme ->
            isDarkTheme = darkTheme
            saveSettings(darkTheme = darkTheme)
        },
        selectedLanguage = selectedLanguage,
        onLanguageChange = { language ->
            selectedLanguage = language
            saveSettings(selectedLanguage = language)
            updateLocale(language)
        },
        areNotificationsEnabled = areNotificationsEnabled,
        onNotificationsToggle = { enabled ->
            areNotificationsEnabled = enabled
            saveSettings(notificationsEnabled = enabled)
        },
        onResetGameData = { showResetDialog = true }
    )
}
```

3.3 RadioButton

```
RadioButton(
    selected = selectedDifficulty == difficulty,
    onClick = { selectedDifficulty = difficulty },
    colors = RadioButtonDefaults.colors(selectedColor = MaterialTheme.colorScheme.primary,
        unselectedColor = MaterialTheme.colorScheme.tertiary),
    modifier = Modifier.padding(horizontal = 70.dp)
)
```

3.4 Switch

```
Switch(
    checked = isDarkTheme,
    onCheckedChange = onThemeChange,
    colors = SwitchDefaults.colors(checkedThumbColor = MaterialTheme.colorScheme.primary,
        uncheckedThumbColor = MaterialTheme.colorScheme.secondary,
        checkedTrackColor = MaterialTheme.colorScheme.onPrimary)
)
```

3.5 Dialog

```
@Composable
fun CustomDialog(
    title: String,
    message: String,
    confirmButtonText: String,
    onConfirm: () -> Unit,
    dismissButtonText: String? = null,
    onDismiss: (() -> Unit)? = null,
    onDismissRequest: () -> Unit
) {
    Dialog(onDismissRequest = onDismissRequest) {
        Surface(
            shape = RoundedCornerShape(8.dp),
            color = MaterialTheme.colorScheme.surface,
        ) {
            Column(
                modifier = Modifier.padding(16.dp),
                horizontalAlignment = Alignment.CenterHorizontally
            ) { this: ColumnScope
                Text(text = title, fontWeight = FontWeight.Bold, fontSize = 20.sp)
                Spacer(modifier = Modifier.height(16.dp))
                Text(text = message)
                Spacer(modifier = Modifier.height(24.dp))
                Row(
                    modifier = Modifier.fillMaxWidth(),
                    horizontalArrangement = Arrangement.SpaceEvenly
                ) { this: RowScope
                    dismissButtonText?.let { it: String
                        Button(
                            onClick = onDismiss ?: onDismissRequest,
                            modifier = Modifier
                                .padding(vertical = 10.dp),
                            shape = RoundedCornerShape(5.dp),
                            colors = ButtonDefaults.buttonColors(
                                containerColor = MaterialTheme.colorScheme.secondary,
                                contentColor = MaterialTheme.colorScheme.tertiary
                            )
                        ) { this: RowScope
                            Text(text = it)
                        }
                    }
                    Button(
                        onClick = onConfirm,
                        modifier = Modifier
                            .padding(vertical = 10.dp),
                        shape = RoundedCornerShape(5.dp),
                        colors = ButtonDefaults.buttonColors(
                            containerColor = MaterialTheme.colorScheme.secondary,
                            contentColor = MaterialTheme.colorScheme.tertiary
                        )
                    ) { this: RowScope
                        Text(text = confirmButtonText)
                    }
                }
            }
        }
    }
}
```

3.6 BroadcastReceiver

```
    private fun scheduleNotification() {
        val alarmManager = getSystemService(Context.ALARM_SERVICE) as AlarmManager
        val intent = Intent( packageContext: this, NotificationReceiver::class.java)
        val pendingIntent = PendingIntent.getBroadcast( context: this,  requestCode: 0, intent, PendingIntent.FLAG_UPDATE_CURRENT)

        val triggerAtMillis = System.currentTimeMillis() + AlarmManager.INTERVAL_DAY

        alarmManager.set(
            AlarmManager.RTC_WAKEUP,
            triggerAtMillis,
            pendingIntent
        )
    }
}
```

```
class NotificationReceiver : BroadcastReceiver() {
    RaijinG
    override fun onReceive(context: Context, intent: Intent) {
        val notificationManager = context.getSystemService(Context.NOTIFICATION_SERVICE) as NotificationManager

        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
            val channel = NotificationChannel( id: "CHANNEL_ID",  name: "Wordle Notifications", NotificationManager.IMPORTANCE_DEFAULT)
            notificationManager.createNotificationChannel(channel)
        }

        val notificationIntent = Intent(context, MainActivity::class.java)
        val pendingIntent = PendingIntent.getActivity(context,  requestCode: 0, notificationIntent, PendingIntent.FLAG_UPDATE_CURRENT)

        val notification = NotificationCompat.Builder(context,  channelId: "CHANNEL_ID")
            .setContentTitle("Time to Play Wordle!")
            .setContentText("Its been a while since you last played Wordle. Come and...")
            .setSmallIcon(R.drawable.ic_launcher_foreground)
            .setContentIntent(pendingIntent)
            .build()

        notificationManager.notify( id: 1, notification)
    }
}
```

3.7 Zoznam použitých implementácií

- Navigation
- Room
- Coroutines
- LifeCycle

3.8 Zoznam použitých komponentov

- Broadcast Receiver
- RadioButton
- Switch

4. Zoznam použitých zdrojov

Broadcast Receiver

https://developer.android.com/develop/background-work/background-tasks/broadcasts

Zoznam slov

https://github.com/charlesreid1/five-letter-words/blob/master/sgb-words.txt