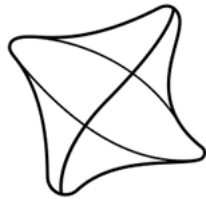


**ŽILINSKÁ UNIVERZITA V ŽILINE**  
**FAKULTA RIADENIA A INFORMATIKY**



**ŽILINSKÁ UNIVERZITA  
V ŽILINE**

**NÁVRH SEMESTRÁLNEJ PRÁCE**  
**VÝVOJ APLIKÁCIÍ PRE MOBILNÉ**  
**ZARIADENIA**

Študijný program:  
Informatika a riadenie

Vypracoval: Jakub Valek

Vyučujúci: doc. Ing. Patrik Hrkút, PhD.

Školský rok:  
2023/2024

Študijná skupina:  
5ZYR23

# 1. ANALÝZA APLIKÁCIE

## 1.1. NÁZOV APLIKÁCIE

Aplikácia má názov „Company Health Check“.

## 1.2 POPIS APLIKÁCIE

Aplikácia slúži pre ľudí, ktorí sa zaujímajú o investovanie do spoločností, ktoré sa verejne obchodujú, tzv. verejne obchodované spoločnosti. Po vyhľadani verejne obchodovanej spoločnosti aplikácia zobrazí aktuálne údaje, finančné ukazovatele a najpodstatnejšie informácie z dokumentov spoločnosti ako sú súvaha (balance sheet), výkaz ziskov a strát (income statement) alebo výkaz peňažných tokov (cash flow), v prehľadnej grafickej forme. Aplikácia je schopná vyhľadať tieto údaje k tisíckam najvyhľadávanejších verejne obchodovateľných spoločností, ako sú napr. Apple Inc., Tesla Inc., Microsoft Corporation, McDonalds's Corporation a pod.

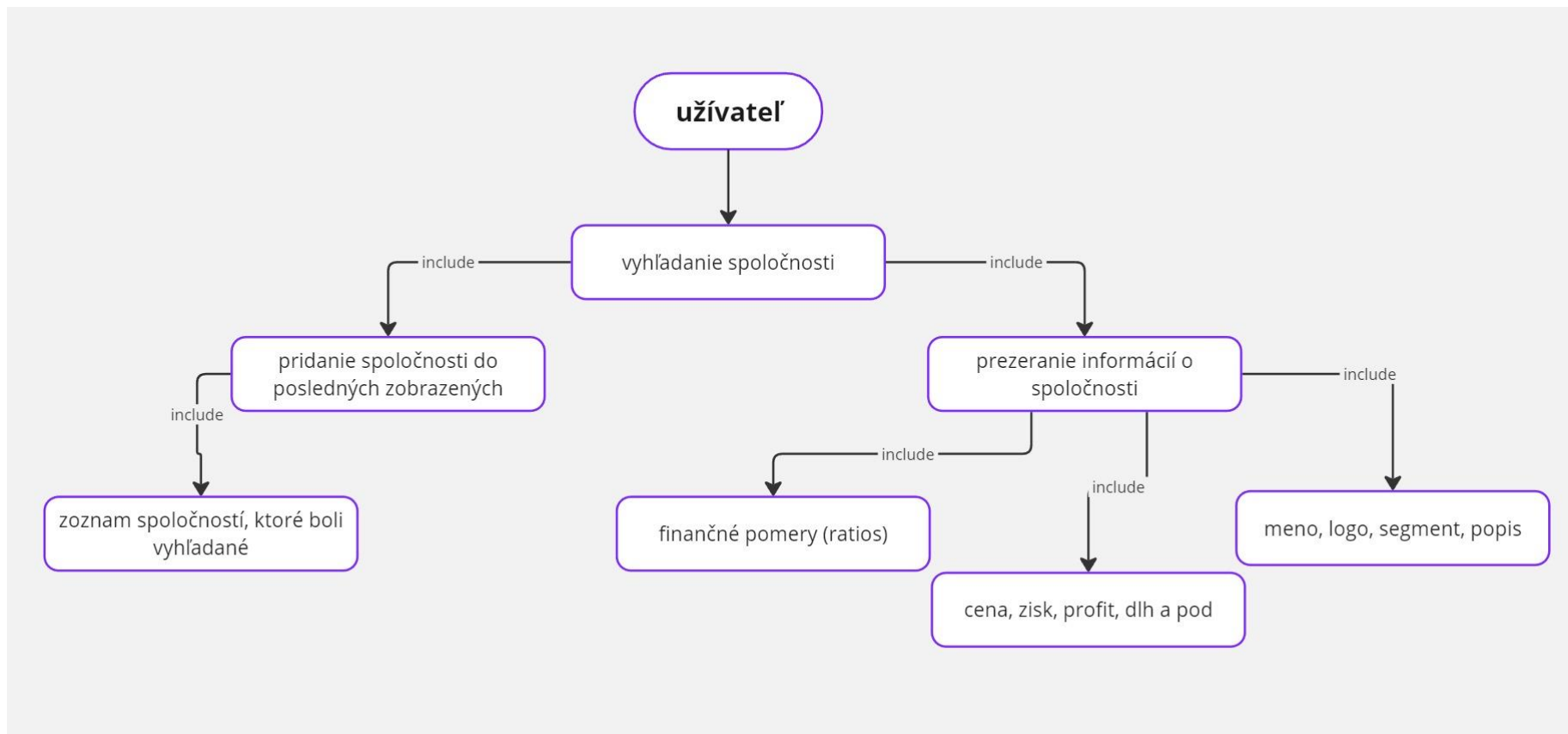
Zmysel aplikácie je tieto údaje prezentovať v čo najprehľadnejšej forme a vybrať iba najpodstatnejšie dáta, aby používateľ nemusel prezeráť veľké tabuľky s údajmi, v ktorých sa začínajúci investor vie stratiť.

## 1.3. FUNKCIONALITY APLIKÁCIE

Aplikácia má viacero funkcionalít medzi ktoré patria:

- Možnosť vyhľadania verejne obchodovanej spoločnosti po zadaní tzv. „tickeru“ spoločnosti. Ticker je symbol a jedinečná kombinácia písmen a čísel, ktoré predstavujú konkrétnu akciu alebo cenný papier kótovaný na burze. (1)
- Zobrazenie údajov k danej spoločnosti, ako sú:
  - názov a logo spoločnosti
  - cena jednej akcie spoločnosti
  - finančné pomery (ratios) ako sú: P/E ratio, cash flow/share a pod.
  - veľkosť dlhu a ukazovatele k tomuto dlhu
  - v prípade, že spoločnosť vypláca dividendu tak údaje o vyplácanej dividende ako je veľkosť dividendy, dividend yield, payout ratio a pod.
  - veľkosť voľných peňažných prostriedkov za uplynulé obdobie
- Aplikácia ukladá spoločnosti do databázy a zobrazuje ju ako posledné zobrazené spoločnosti

## 1.4. USE CASE APLIKÁCIE



Obrázok 1 - use case navrhovanej aplikácie

Zdroj: vlastné spracovanie

## 1.5. ARCHITEKTÚRA APLIKÁCIE

Aplikácia je naprogramovaná v programovacom jazyku Kotlin s využitím Jetpack Compose UI toolkit.

Pre získavanie informácií ohľadom vybranej spoločnosti sa využíva API zo stránky <https://site.financialmodelingprep.com/>. Sú to informácie ktoré „sú licencované a pochádzajú od kurátorských dátových partnerov tretích strán, ako je napríklad Komisia pre cenné papiere a burzu USA (SEC), Rada guvernérov Federálneho rezervného systému.“ (2).

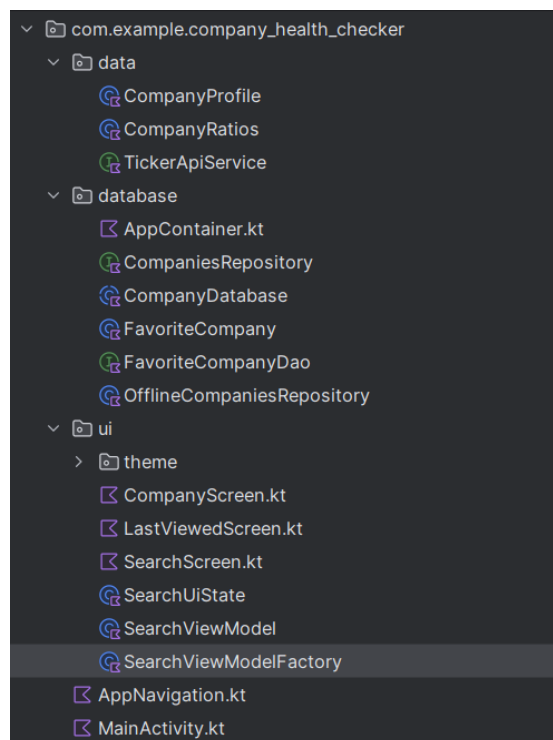
### 1.5.1. Obmedzenia spojené s dostupnými API

FMP stránka ponúka v basic verzii účtu 250 volaní / deň. (3) Pri spracovaní údajov ohľadom firmy sa používajú 2 endpointy z tejto stránky, čiže najvyšší počet hľadání za deň je 125 vyhľadávání.

## 1.6. IMPLEMENTÁCIA APLIKÁCIE

Aplikácia bola rozdelená na 3 základné balíčky:

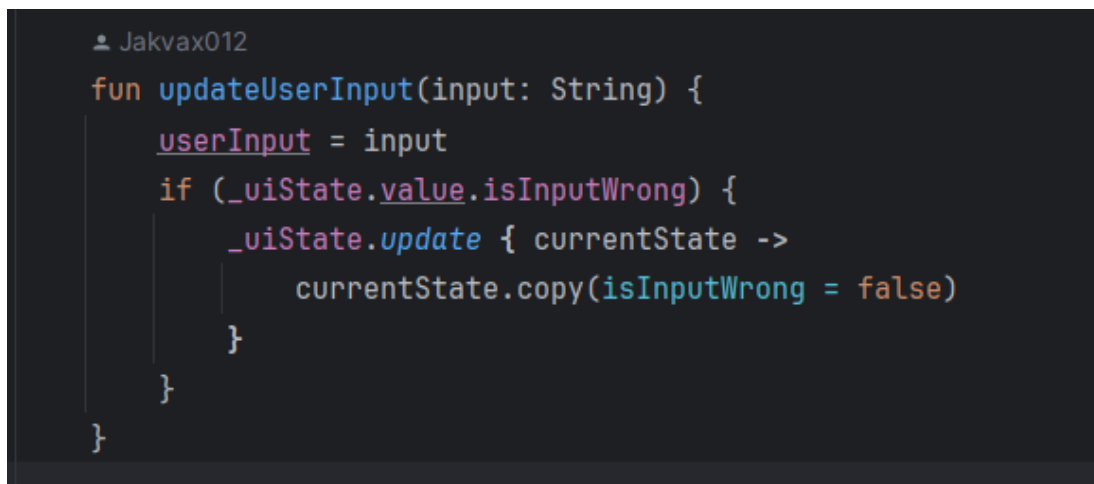
- ui – zabezpečuje vzhľad 3 hlavných obrazoviek spolu s ViewModel a uiState
- database – nachádzajú sa v nej triedy a súbory zabezpečujúce chod databázy
- data – tento balíček je zodpovedný za stiahnutie a načítanie dát z API



Obrázok 2 - rozdelenie balíčkov

### 1.6.1 Ako funguje vyhľadanie spoločnosti


SearchScreen obrazovka v sebe zahŕňa Composable funkcie na vykreslenie úvodnej obrazovky, na ktorej sa zadáva aj vstup v podobe tickera spoločnosti, ktorú chceme vyhľadať. Na SearchScreen je implementovaná SearchViewModel, spolu s SearchUiState. Pri zadávaní textu do textového poľa v SearchScreen, sa toto textové pole aktualizuje pri každom napísanom znaku pomocou ViewModelu.

A screenshot of a code editor showing a Kotlin function named updateUserInput. The function takes an input String and updates the \_uiState. It sets userInput to the input, then checks if \_uiState.value.isInputWrong. If true, it updates \_uiState with a new state where isInputWrong is false. The code is written in a dark-themed editor with syntax highlighting.

```
fun updateUserInput(input: String) {  
    userInput = input  
    if (_uiState.value.isInputWrong) {  
        _uiState.update { currentState ->  
            currentState.copy(isInputWrong = false)  
        }  
    }  
}
```

Obrázok 3 - updateUserInput

Po tom ako užívateľ dopísal ticker a odoslal ho sa v SearchScreen kontroluje, či je tento ticker správny alebo nie.

A screenshot of a code editor showing a Kotlin function that handles company data. It starts with a try block. Inside, it calls tickerApiService.getCompanyProfile(ticker) to get a profileResponse. If the response is not empty, it gets the first element of the response as companyProfile and updates \_uiState with a new state where isInputWrong is false and companyProfile is set to the retrieved profile. The code is written in a dark-themed editor with syntax highlighting.

```
// Company data - first endpoint  
try {  
    val profileResponse = tickerApiService.getCompanyProfile(ticker)  
    if (profileResponse.isNotEmpty()) {  
        val companyProfile = profileResponse.first()  
        _uiState.update { currentState ->  
            currentState.copy(  
                isInputWrong = false,  
                companyProfile = companyProfile,  
            )  
        }  
    }  
}
```

Obrázok 4 - kontrola Tickeru

Kontrola prebieha spôsobom, že sa odošle request na server FMP a v prípade, že server vráti prázdny JSON, zmení sa premenná v SearchUiState „isInputWrong“ na true a vráti používateľovi na obrazovke text, že ticker bol zadán nesprávne. V prípade, že JSON nie je prázdny, začne program sťahovať dáta o spoločnosti najprv z prvého endpointu, ktorý sa týka základných údajov spoločnosti a následne si vypýta druhý endpoint, ktorý sa týka finančných údajov spoločnosti.

```

@Javvax012
interface TickerApiService {
    @Javvax012
    @GET("profile/{ticker}")
    suspend fun getCompanyProfile(
        @Path("ticker") ticker: String,
        @Query("apikey") apiKey: String = "PrV6W7tSON7U60N1ipF46tH66Qccto6A"
    ): List<CompanyProfile>

    @Javvax012
    @GET("ratios-ttm/{ticker}")
    suspend fun getCompanyRatios(
        @Path("ticker") ticker: String,
        @Query("apikey") apiKey: String = "PrV6W7tSON7U60N1ipF46tH66Qccto6A"
    ): List<CompanyRatios>

    @Javvax012
    companion object {
        private const val BASE_URL = "https://financialmodelingprep.com/api/v3/"

        @Javvax012
        fun create(): TickerApiService {
            val retrofit = Retrofit.Builder()
                .baseUrl(BASE_URL)
                .addConverterFactory(GsonConverterFactory.create())
                .build()
            return retrofit.create(TickerApiService::class.java)
        }
    }
}

```

Obrázok 5 - Ukážka kódu s endpoints

Následne tieto údaje sa zozbierajú a uložia sa do dvoch premenných v SearchUiState, kde je možné s nimi ďalej pracovať.

### 1.6.2 Vypisovanie údajov o spoločnosti

Za obrazovku, na ktorej sa vypisujú údaje o spoločnosti je zodpovedná CompanyScreen, ktorá spolupracuje s SearchUiState.

```

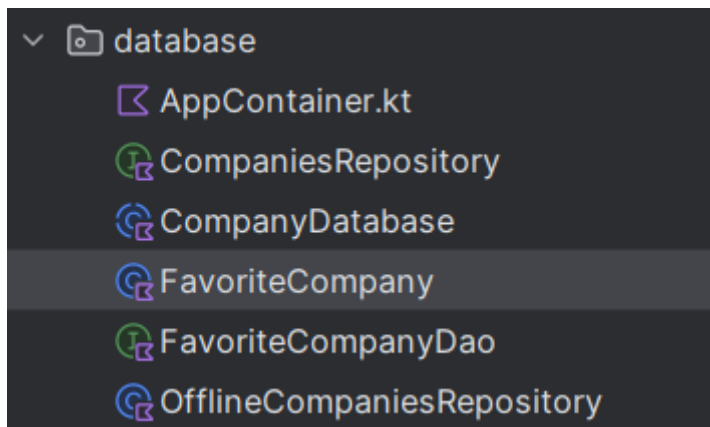
@Composable
fun CompanyData(searchUiState: SearchUiState) {
    searchUiState.companyRatios?.let { ratios ->
        Column(
            modifier = Modifier.padding(16.dp)
        ) {
            this: ColumnScope
            Text(text = stringResource(id = "Dividend information"), fontSize = 18.sp, fontWeight = FontWeight.Bold)
            Spacer(modifier = Modifier.height(8.dp))
            Row(
                horizontalArrangement = Arrangement.SpaceBetween,
                modifier = Modifier.fillMaxWidth()
            ) {
                this: RowScope
                CompanyInfoBox(stringResource(id = "Dividend yield"), value = "${"%".format(ratios.dividendYieldPercentageTTM)} %" ?: "N/A")
                CompanyInfoBox(stringResource(id = "Dividend rate"), value = "${"%".format(ratios.dividendPerShareTTM)} $" ?: "N/A")
            }
            Spacer(modifier = Modifier.height(8.dp))
        }
    }
}

```

Obrázok 6 - vypisovanie údajov

### 1.6.3 Databáza a ukladanie spoločností, ktoré boli naposledy zobrazené

Aplikácia využíva Room databázu na uchovávanie údajov o firme, ktorá bola zobrazená. V obrázku nižšie je možné vidieť, že v aplikácii sú implementované súbory/triedy ako: Entita, Dao, Databáza, Repository, OfflineRepository a AppContainer.



Obrázok 7 - database package

V ui package sa nachádza ešte SearchViewModelFactory, ktorý prepája SearchViewModel spolu s databázou a Repository.

Do databázy sa pridáva, každá spoločnosť, ktorá bola vyhľadaná (toto pridanie prebieha v SearchViewModel, keď je pozitívna odpoveď na endpoint z API) a ukladá tieto 3 dáta o spoločnosti: ticker, name a image a ako @PrimaryKey je považovaný ticker.

```

@Entity(tableName = "favorite_companies")
data class FavoriteCompany(
    @PrimaryKey
    val ticker: String,
    val name: String,
    val image: String
)
```

Obrázok 8 - údaje, ktoré sa ukladajú do databázy

### 1.6.4 LastViewedScreen

Táto obrazovka zabezpečuje vypísanie spoločností, ktoré sú v databáze do zoznamu. Po kliknutí na danú spoločnosť, je používateľ presmerovaný na stránku spoločnosti, keď sa použije uložený ticker z databázy a údaje o spoločnosti sa znovu načítajú cez API.

```

LazyColumn(modifier = Modifier.fillMaxSize()) { this: LazyListScope
    items(companies) { this: LazyItemScope company ->
        CompanyItem(company) {
            coroutineScope.launch { this: CoroutineScope
                searchViewModel.fetchCompanyData(company.ticker, fromViewedScreen: true)
                navController.navigate(Screen.CompanyScreen.name)
            }
        }
    }
}

```

Obrázok 9 - LazyColumn a zobrazenie spoločností

### 1.6.5 AppNavigation

V tomto súbore je vytvorená navigácia medzi 3 obrazovkami.

```

enum class Screen {
    SearchScreen,
    CompanyScreen,
    LastViewedScreen
}

@Composable
fun AppNavigation(navController: NavHostController, searchViewModel: SearchViewModel) {
    NavHost(navController = navController, startDestination = com.example.company_health_checker.Screen.SearchScreen.name) { this: NavHostController
        composable(com.example.company_health_checker.Screen.SearchScreen.name) { it: NavBackStackEntry
            SearchScreen(navController = navController, searchViewModel = searchViewModel)
        }
        composable(com.example.company_health_checker.Screen.CompanyScreen.name) { it: NavBackStackEntry
            CompanyScreen(searchViewModel = searchViewModel)
        }
        composable(com.example.company_health_checker.Screen.LastViewedScreen.name) { it: NavBackStackEntry
            LastViewedScreen(navController = navController, searchViewModel = searchViewModel)
        }
    }
}

```

Obrázok 10 - Kód s navigáciou medzi 3 obrazovkami

### 1.6.6 MainActivity

Inicializuje aplikáciu a repository.

```

class MainActivity : ComponentActivity() {
    private lateinit var appContainer: AppDataContainer

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        appContainer = AppDataContainer(context = this)
        val searchViewModel: SearchViewModel by viewModels {
            SearchViewModelFactory(application, appContainer.companiesRepository)
        }
        setContent {
            Company_Health_CheckerTheme {
                MainScreen(searchViewModel)
            }
        }
    }
}

@Composable
fun MainScreen(searchViewModel: SearchViewModel) {
    val navController = rememberNavController()
    AppNavigation(navController = navController, searchViewModel = searchViewModel)
}

```

Obrázok 11 - MainActivity



## **1.7 IMPLEMENTÁCIA APLIKÁCIE V BODOCH**

- 3 základné obrazovky
- AndroidX komponenty: Navigation, Room, ViewModel
- Externá knižnica/framework: Coil – zobrazovanie obrázka z URL adresy  
Retrofit – spracovanie API odpovedí
- Použitie sieťovej komunikácie pre spracovanie údajov
- Aplikácia rozdelená do balíčkov, oddelenie logiky a používateľského rozhrania, logické pomenovanie identifikátorov, zachytávanie výnimiek, dodržanie OOP

## **2. APLIKÁCIE PODOBNÉHO ZAMERANIA**

Táto kapitola sa zaoberá zmapovaním už podobne existujúcich aplikácií.

### **2.1. YAHOO FINANCE: STOCK NEWS**

Yahoo Finance: Stock News je jedna z najväčších aplikácií na sledovanie akcií, komodít, kryptomien a pod. Medzi hlavné funkcie tejto aplikácie patrí:

- Sledovanie akcií: Je možné sledovať akcie spoločností, a dostávať aktuálne ceny a personalizované správy o týchto spoločnostiach.
- Interaktívne grafy: Je možné porovnávať a hodnotiť akcie pomocou plnoobrazkových interaktívnych grafov.
- Správy: Aplikácia ponúka mnoho článkov na čítanie ohľadom ekonomiky, financií, alebo konkrétnej spoločnosti.
- Sledovanie ostatných finančných nástrojov: Je možné sledovať aj iné finančné nástroje, akými sú napr. meny, dlhopisy, indexy, ETF fondy, futures a pod.
- Historické dáta a zdravie spoločnosti: Aplikácia ponúka možnosť zobrazenia historických údajov z finančných dát jednotlivých spoločností.

Aplikácia Yahoo Finance: Stock News aj finančné údaje ako sú balance sheet, income statement a cash flow statement, podobne ako to ponúka vytvorená aplikácia. Narozdiel od Yahoo Finance, aplikácia prinesie tieto informácie vo veľmi graficky čistej a prehľadnej forme, na rozdiel od aplikácie Yahoo Finance, ktorá má tieto dáta v tabuľkách.

22:53 73%

## Financials

### Income Statement

All numbers in thousands

Breakdown	TTM
<b>29. 9. 2020</b>	
Total Revenue	385 706 000
Cost of Revenue	212 035 000
Gross Profit	173 671 000
✓ Operating Expenses	
Research Development	29 902 000
Selling General and Admini...	25 111 000
<b>Total Operating Expenses</b>	55 013 000
<b>Operating Income or Loss</b>	118 658 000
Interest Expense	-
Total Other Income/Expense...	-39 000
Income Before Tax	118 436 000
Income Tax Expense	17 523 000
Income from Continuing Ope...	100 913 000
<b>Net Income</b>	100 913 000
Net Income available to com...	100 913 000
Basic EPS	6.16

Obrázok 12 – Tabuľka income statement v aplikácii Yahoo Finance: Stock news  
Zdroj: (4)

## 2.2. SIMPLY WALL ST: STOCK ANALYSIS

Simply Wall St: Stock Analysis je aplikácia, ktorá pomáha s analýzou akcií a investičných portfólií. Tu sú niektoré z jej funkcií:

- Sledovanie akcií a portfólií: Aplikácia umožňuje jednoducho spravovať viacero sledovaných zoznamov a portfólií. Vizuálne zobrazuje investície, identifikuje ich silné a slabé stránky a sleduje výkonnosť.
- Podrobné analýzy akcií: Akčné správy poskytujú úplný obraz o spoločnosti s vizuálnymi informáciami a nezávislou analýzou. Je možné zistiť minulý výkon, riziká a odmeny, hodnotenie a porovnanie s inými spoločnosťami.

Aplikácia je veľmi rozsiahla a obsahuje aj informácie o zdraví spoločnosti, rôzne ukazovatele, tabuľky a podobne. Pre svoju veľkú rozsiahlosť a množstvo informácií sa stáva neprehľadnou a v prípade, že používateľ chce nájsť nejakú konkrétnu informáciu, vie sa v množstve informácií ľahko stratiť.



Obrázok 13 – záložka health v aplikácii Simply Wall St: Stock Analysis  
Zdroj: (5)

### **3. ZDROJE**

1. CFI - What is Ticker? [Online] [Dátum: 6. 4 2024.] Preložené z Anglického jazyka. <https://corporatefinanceinstitute.com/resources/wealth-management/what-is-ticker/>.
2. FMP - domovská stránka. [Online] [Dátum: 7. 4 2024.] Preložené z Anglického jazyka. <https://site.financialmodelingprep.com/>.
3. FMP - pricing. [Online] [Dátum: 7. 4 2024.] Preložené z Anglického jazyka. <https://site.financialmodelingprep.com/developer/docs/pricing>.
4. *Screenshot z Yahoo Finance: Stock news*. [Mobilná aplikácia] 2024.
5. *Screenshot z aplikácie Simply Wall St: Stock Analysis*. [Mobilná aplikácia] 2024.