Dokument specifikace požadavků (DSP)

Název projektu: Burzalupa
Verze dokumentu: 1.0
Datum: 2025-05-15
Zodpovědný autor: Tým Burzalupa

---

1. Úvod

1.1 Účel

Tento dokument specifikuje veškeré požadavky na vývoj, implementaci a provoz softwarového řešení **Burzalupa**. Je určen především pro zadavatele projektu, technické architekty a vývojový tým.

Hlavní cíle dokumentu:

- Popsat, co má software Burzalupa dělat (funkční požadavky)
- Určit, za jakých podmínek má systém fungovat (nefunkční požadavky)
- Zajistit srozumitelnost mezi zadavatelem a realizačním týmem
- Sloužit jako podklad pro akceptační testy
- Sloužit jako vstupní bod pro budoucí rozšiřování systému

Dokument je živý a může být postupně doplňován či revidován podle změn v zadání, dostupných technologií nebo obchodních priorit.

1.2 Rozsah produktu

Burzalupa je webová aplikace zaměřená na:

- Sběr, analýzu a filtrování dat o akciových titulech z externích zdrojů (např. Tiingo API)
- Prezentaci těchto dat uživateli v přehledném webovém rozhraní
- Umožnění uživatelské personalizace (např. ukládání seznamu oblíbených tickerů)
- Zobrazení živých systémových logů (např. pomocí WebSocket kanálu)

Specifické funkcionality:

- Automatická analýza titulů, které 3 dny po sobě nebo 2× z 5 dnů vykázaly pokles
- Možnost plánovaného stahování dat (časovač)
- Konfigurační rozhraní prostřednictvím .env proměnných

Uživatelské scénáře:

- Uživatel si nastaví preferované tickery a každý den ráno (např. v 6:00) dostane aktualizovaný přehled
- Uživatel se může připojit k WebSocket a sledovat logovací zprávy v reálném čase

Systém není určen pro:

- Automatizované obchodování
- Přímé zobrazování cen v reálném čase (není realtime feed)

1.3 Definice a zkratky

Zkratka	Význam
DSP	Dokument specifikace požadavků
API	Aplikační programovací rozhraní
REST	Architektura pro přístup k datům přes HTTP
JSON	Strukturovaný formát pro výměnu dat (JavaScript Object Notation)
Ticker	Zkratka pro akciový titul, např. "AAPL" pro Apple
WebSocket	Obousměrný komunikační protokol pro přenos zpráv v reálném čase
CRON	Plánovací nástroj pro periodické spouštění procesů
NestJS	Node.js framework používaný pro backend vývoj

Pro účely dokumentace se používá český jazyk. Všechny technické výstupy (např. hlavičky požadavků, JSON struktury) mohou být v angličtině.

2. Architektura systému

Architektura systému **Burzalupa** je založena na vícevrstvé architektuře (multi-tier) s důrazem na modularitu, škálovatelnost a čisté oddělení zodpovědností. Systém lze rozdělit do tří hlavních vrstev:

2.1 Logická struktura systému

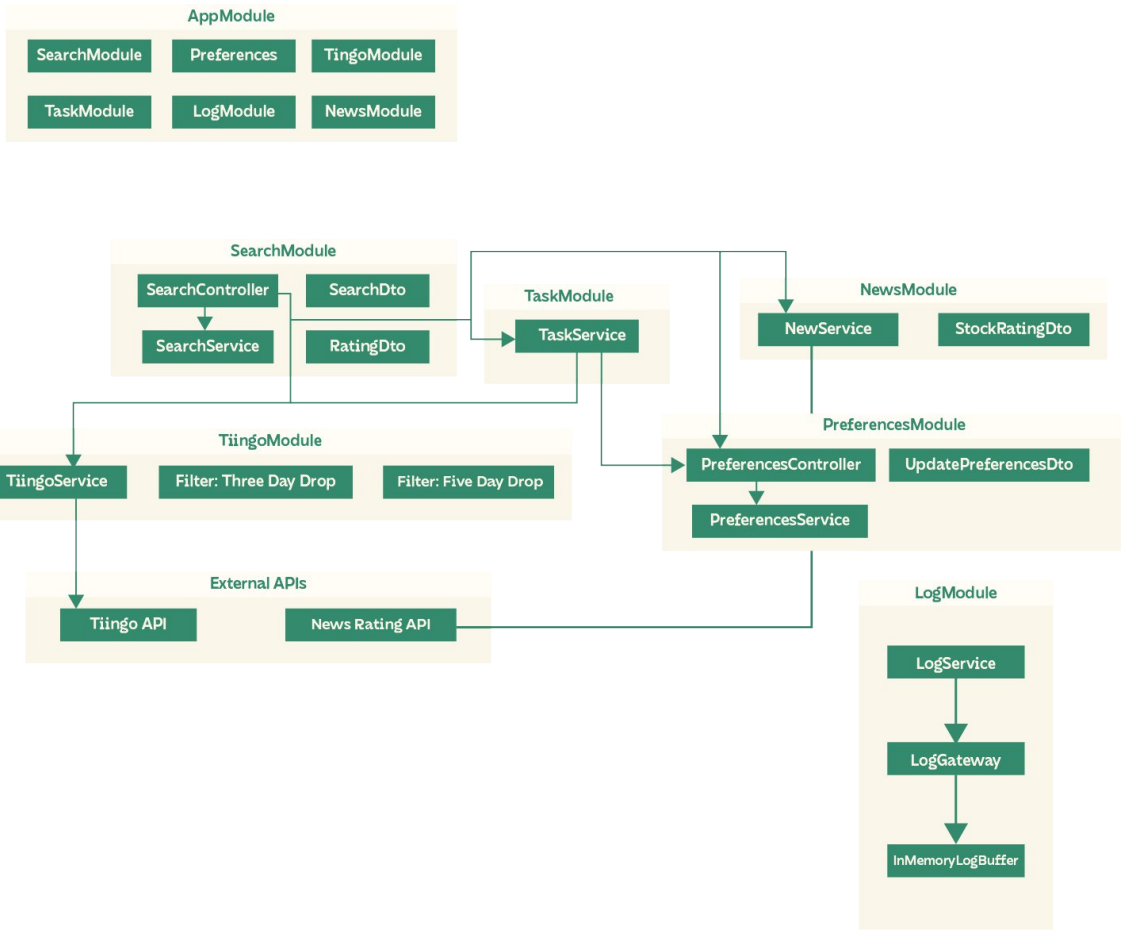
1. Prezentační vrstva (Frontend)

- Realizována pomocí HTML + JavaScript
- Tři hlavní stránky
    - / – hlavní stránka s vyhledáváním a přidáváním tickerů do oblíbených

- `/favorites` – zobrazování a správa preferovaných tickerů
- `/log` – živý výpis systémových logů
    - Připojuje se na WebSocket a REST API

2. Aplikační vrstva (Backend)

- Framework: **NestJS** (modulární Node.js framework)
- Základní moduly:
    - `search` – vyhledávání a filtrování akcií
    - `preferences` – správa preferovaných tickerů
    - `log` – WebSocket logovací server
    - `news` – zpracování dat z hodnotících API
    - `tiingo` – napojení na historická a aktuální data
    - `task` – plánovač (CRON úlohy)



3. Datová vrstva

- Data nejsou trvale ukládána do databáze, ale:
    - Preferované tickery jsou ukládány do `preferences.json`
    - Historie cen do `stock_prices_history.json`
    - Logy jsou bufferovány v paměti a streamovány klientovi

2.2 Komunikační toky

REST API (HTTP)

- Slouží pro získávání a filtrování dat, např.:
    - `GET /search/filter/3d`
    - `GET /preferences`
    - `PATCH /preferences`

WebSocket API

- Jednosměrný kanál `/log`
- Každá nová logovací zpráva se streamuje klientovi (event `log`)

CRON Plánovač

- Spouští aktualizace dat každých 6 hodin (0:00, 6:00, 12:00, 18:00)
- Pokud existují preferované tickery, stáhnou se nové ceny

2.3 Architektonické schéma

2.4 Vlastnosti architektury

- **Modularita:** Každá funkce je řešena vlastním modulem
- **Flexibilita:** Možnost -ířit o databázové úložiště
- **Testovatelnost:** Díky NestJS DI kontejneru lze moduly snadno testovat
- **Oddělení odpovědností:** Logika a prezentace jsou jasně rozděleny

Tato architektura umožňuje snadnou údržbu, přehlednost a adaptaci na změny v budoucnu.

Systém se skládá z těchto hlavních komponent:

- **Frontend**: HTML+JavaScript (3 stránky: /, /log a /favorites)

- **Backend**: NestJS aplikace s moduly `log`, `search`, `preferences`, `news`, `tiingo`, `task`
- **Externí API**:
    - Tiingo – historická a aktuální burzovní data
    - News – datový zdroj hodnocení akcií

Komunikační rozhraní:

- REST API pro většinu datových operací
- WebSocket pro živé logy
- Interní plánovač CRON pro periodické aktualizace

3. Specifické požadavky

3.1 Funkční požadavky

Požadavek 1.1 – Automatické stažení dat

```
// Spouštěno každých 6 hodin
if (preferences.hasFavorites()) {
  const tickers = preferences.get();
  const result = await tiingoService.updateStockPricesHistory(tickers);
  logger.info("Aktualizována historie cen");
}
```

Požadavek 1.2 – Filtrace titulů

Filtrované funkce implementují algoritmy:

- **3d pokles**: pokud cena klesala tři po sobě jdoucí dny
- **5d pokles**: pokud pokles nastal alespoň 3krát z posledních 5 dní

Ukázka algoritmu (reálný kód):

```
const recent = daily.slice(-3).reverse();
return recent[0].tngoLast < recent[1].tngoLast && recent[1].tngoLast <
recent[2].tngoLast;
```

Požadavek 1.3 – Správa preferencí

- Data se ukládají do `files/preferences.json`
- Operace:
    - `GET /preferences` — načtení oblíbených titulů
    - `PATCH /preferences` — aktualizace seznamu

Ukázka JSON:

```
{
  "favoriteTickers": ["AAPL", "TSLA"]
}
```

Požadavek 1.4 – WebSocket logy

- Po připojení klienta systém odešle posledních 100 logů:

```
const messages = await this.buffer.getBufferedMessages();
for (const msg of messages) client.emit("log", msg);
```

3.2 Nefunkční požadavky

Kategorie	Požadavek
Výkon	REST API vrací odpověď do 2 sekund
Dostupnost	Systém je funkční 99 % času v průběhu měsíce
Robustnost	Po selhání se opakuje požadavek každých 5 min (max 30)
Údržba	Konfigurace pomocí `.env` a jednoduchého přestavení
Bezpečnost	Jen validní JSON vstupy, data validována před zpracováním

---

4. Formuláře a pseudokód

Funkce: Filtrace 3denní pokles

Popis: Vrátí tituly, které 3 dny za sebou klesaly.

Vstupy: Historie cen akcií (array objektů s `tngoLast`)
Výstupy: Seznam tickerů splňujících podmínku
Pseudokód:

```
function goingDownThreeDays(data) {
  const daily = sampleOnePerDay(data);
  if (daily.length < 3) return false;
  const recent = daily.slice(-3).reverse();
  return recent[0].tngoLast < recent[1].tngoLast && recent[1].tngoLast
< recent[2].tngoLast;
}
```

---

5. Use Case

UC01 – Zobrazení oblíbených akcií s poklesem

Aktér: Uživatel
Scénář:

1. Uživatel otevře stránku "Favorites"
2. Zvolí filtr „3d“
3. Frontend volá `GET /search/filter/3d`
4. Backend získá tickery a filtruje podle historie
5. Vrátí tickery, které jsou v preferencích a zároveň splňují podmínku poklesu

Alternativní tok:

- Pokud nejsou žádné oblíbené tituly, frontend zobrazí hlášku „No favorites“

---

6. Rozhraní systému

REST API: `/search/ratings`

- **Metoda:** `GET`
- **Parametry:** `tickers=AAPL,MSFT`
- **Výstup:** JSON pole:

```
[
  { "name": "AAPL", "rating": 3, "sell": 0, "date": 1714521600 },
  { "name": "MSFT", "rating": 4, "sell": 1, "date": 1714521600 }
]
```

WebSocket: `/log`

- **Event:** `log`
- **Data:** `string | object` — jednotlivé logovací zprávy

---

7. Omezení implementace

- Programovací jazyk: TypeScript (NestJS framework)
- Testovací nástroje: Vitest
- Scheduler: Cron + systémová služba
- Logy: Pino + WebSocket transport