# Zadání (2. kolo) – Node.js + TypeScript mini-aplikace „FX Rates“

Cílem je ověřit práci s HTTP API, periodickým fetchováním dat, jednoduchou architekturou služby a ošetřením chyb.

## Cíl

Vytvořte backendovou aplikaci v Node.js a TypeScriptu, která bude periodicky stahovat kurzovní lístek, ukládat data do in-memory cache a poskytne několik HTTP endpointů.

## Funkční požadavky

### 1) Periodické stahování kurzů

- Aplikace musí periodicky (např. každých N minut) stahovat kurzovní lístek z veřejného zdroje dle vašeho výběru.
- Získaná data ukládejte do in-memory cache (např. Map, objekt, případně knihovna dle volby).
- Cache musí vždy obsahovat poslední úspěšně stažená data.
- Pokud stahování selže, aplikace musí dál běžet – rozumně to zalogujte a ponechte poslední známá data v cache.

### 2) HTTP endpointy

Aplikace musí poskytovat následující routy:

### A) Nezabezpečený healthcheck

- GET /healthcheck
- Response: status 200, Content-Type text/plain, body např. OK (libovolný jednoduchý text).
- Tento endpoint je veřejný (bez autentizace).

### B) Dvě zabezpečené GET routy (JSON)

Obě routy musí být GET, vracet application/json a musí být zabezpečené. Jak si routy pojmenujete a jaké zvolíte zabezpečení je na vás (např. API key, Basic Auth, Bearer token/JWT apod.). Popište to v README.

### B1) Výpis všech kurzů

- Endpoint název je na vás.
- Musí vracet kompletní výpis kurzů ve striktně daném formátu:

```
[
  { "USD": 23.45 },
  { "EUR": 24.12 }
]
```

- Response je pole objektů.
- Každý objekt obsahuje právě jeden klíč.
- Klíč je kód měny (ISO 4217).
- Hodnota je aktuální kurz (number) vůči základní měně zvoleného kurzovního lístku.
- Pořadí položek v poli není podstatné.

### B2) Kurz pro konkrétní měnu

- Endpoint název je na vás.
- V parametru musí být kód měny (path param nebo query param – vaše volba).
- Musí vracet JSON ve striktně daném formátu:

```
{ "USD": 23.45 }
```

- Pokud měna neexistuje nebo není v cache, vraťte vhodný status (např. 404) a JSON s chybovou zprávou (formát je na vás, ale musí být konzistentní).

## Nefunkční požadavky

- TypeScript je povinný.
- Aplikace musí jít spustit lokálně.
- Stručné a srozumitelné logování.
- Preferujeme čistou strukturu projektu (oddělení HTTP vrstvy / logiky / fetchu / cache).

## Odevzdání

- Repo (GitHub/GitLab) nebo ZIP.
- README, které obsahuje: jak projekt spustit, jak nastavit/zadat autentizaci pro zabezpečené routy, seznam endpointů s příklady request/response, zdroj kurzovního lístku a periodu stahování, a krátké shrnutí architektury.

## Bonus

- Unit testy (alespoň pro parser / service / cache).
- Validace vstupů.
- Graceful shutdown / ošetření chyb.
- Informace „kdy naposledy proběhl úspěšný fetch“ (např. v paměti nebo v logu; klidně i jako součást admin odpovědi).

## Hodnocení

- Čitelnost a kvalita kódu + struktura.
- Ošetření chyb a edge-case (nedostupný zdroj, prázdná cache, neexistující měna).
- Konzistentní API a vhodné status kódy.
- Rozumné zabezpečení a jasné vysvětlení v README.

Datum vygenerování zadání: 30.01.2026