UNIVERZITA PARDUBICE   
Fakulta elektrotechniky a informatiky

*NoSQL databáze dokumentová – MongoDB*

*Oleksandr Aronov, 3 ročník*

V …..dne….

Obsah

[Úvod 3](#_Toc190176618)

[1 Architektura 4](#_Toc190176619)

[1.1 Schéma a popis architektury 4](#_Toc190176620)

[1.2 Specifika konfigurace 4](#_Toc190176621)

[1.2.1 CAP teorém 4](#_Toc190176622)

[1.2.2 Cluster 4](#_Toc190176623)

[1.2.3 Uzly 4](#_Toc190176624)

[1.2.4 Sharding 4](#_Toc190176625)

[1.2.5 Replikace 4](#_Toc190176626)

[1.2.6 Perzistence dat 4](#_Toc190176627)

[1.2.7 Distribuce dat 4](#_Toc190176628)

[1.2.8 Zabezpečení 4](#_Toc190176629)

[2 Funkční řešení 5](#_Toc190176630)

[2.1 Struktura 5](#_Toc190176631)

[2.2 Instalace 5](#_Toc190176632)

[3 Případy užití a případové studie 6](#_Toc190176633)

[4 Výhody a nevýhody 7](#_Toc190176634)

[5 Další specifika 8](#_Toc190176635)

[6 Data 9](#_Toc190176636)

[7 Dotazy 10](#_Toc190176637)

[Závěr 11](#_Toc190176638)

[Zdroje 12](#_Toc190176639)

[Přílohy 13](#_Toc190176640)

Úvod

Tato semestrální práce se zabývá návrhem, implementací a provozem dokumentové NoSQL databáze MongoDB ve verzi 6.0.2. Cílem je detailní popis klastrové architektury, shardingu, replikace a bezpečnostních mechanismů v prostředí Docker‑Compose. Práce dále představuje automatizované skripty pro inicializaci clusteru, import dat a testování.

**Rozsah projektu**

* **Zahrnuje**: nasazení a konfiguraci MongoDB clusteru (config servers, shards, mongos routery), zabezpečení (SCRAM‑SHA, keyFile), Docker‑Compose orchestraci, import reálných datasetů a základní Python analýzu dat.
* **Nezahrnuje**: detailní monitoring (Prometheus/Grafana), pokročilé ACID transakce napříč shardovanými kolekcemi, replikaci mezi geograficky oddělenými datacentry.

# Architektura

Celá architektura je navržena pro zajištění vysoké dostupnosti, škálovatelnosti a zabezpečení.

## Schéma a popis architektury

Níže je uvedeno schéma clusteru:

(image)

Popis komponent:

* **Config Servers (rs-config-server)**: tři uzly v replica setu, uchovávají metadata a mapování shard key → shard.
* **Shardy (rs-shard-01, rs-shard-02, rs-shard-03)**: tři replica sety, každý se skládá ze tří uzlů pro horizontální dělení dat.
* **Mongos Routery (router01, router02)**: dva load‑balanced vstupní body pro klienty, zajišťují směrování dotazů.

**Odchylky od doporučení:**

* Doporučené je minimálně dva routery pro vysokou dostupnost → implementováno.
* Secundární uzly config serverů mají priority 0.5, aby primární vykonával zápis metadat s vyšším výkonem.

## Specifika konfigurace

### CAP teorém

Pro řešení se upřednostňuje **Availability** (A) a **Partition tolerance** (P), s eventual consistency (E) pro replikované dokumenty. Tento výběr vychází z požadavku na nepřetržitý provoz i při výpadku jednoho či více uzlů.

### Cluster

Použíto **jeden logický cluster**, který kombinuje:

* 3 config servery,
* 9 shard-nodů (3×3),
* 2 mongos routery.

V jednom clusteru je jednodušší správa a konzistence metadat, georeplikaci nevyužíváno.

### Uzly

Celkově je nasazeno 14 kontejnerů:

* 3× config server,
* 3×3 shard‑uzelů,
* 2× mongos router.

Každý shard má RF=3, aby byl zachován quorum-based výběr primárního uzlu.

### Sharding

Pro horizontální škálování používáno vlastní shard key s vysokou kardinality. Data se rovnoměrně rozkládají mezi tři shards, což podporuje lineární nárůst kapacity.

### Replikace

Replica factor 3 je nastaven v každém shard replica setu. Primární uzel má *priority: 1*, sekundární uzly *priority: 0.5*, čímž se optimalizuje výkon zápisů.

### Perzistence dat

Používáno výchozí storage engine **WiredTiger** s journalingem. Zápisy jsou nejprve aplikovány do journalu a poté do datových souborů na disku. Cache velikost je ponechána na výchozí hodnotě; checkpointy se ukládají každých 60 s.

### Distribuce dat

Při zápisu klient posílá dokument na libovolný mongos, který podle metadata z config serveru zjistí, do kterého shardu má dokument patřit, a oznámí to primárnímu uzlu daného shardu. Při čtení může mongos využít sekundáře, pokud je nastaven readPreference, čímž se snižuje zátěž primárních uzlů.

### Zabezpečení

Autentizace a autorizace probíhá pomocí keyFile a SCRAM‑SHA‑256. KeyFile je na všech uzlech uloženo s právy 600, čímž je zajištěno, že pouze MongoDB proces má přístup. Uživatelé se vytvářejí v databázi *admin* s rolí *root.*

# Funkční řešení

Cílem této kapitoly je popsat adresářovou strukturu projektu, detailní konfiguraci *docker-compose.yml* a postup instalace a inicializace celého clusteru.

## Struktura

Projekt je rozdělen do hlavních složek:

* data

Tato složka obsahuje datové soubory ve formátu CSV (např. TopAnime.csv, TopMovies.csv, TopNetflix.csv). Tyto soubory slouží jako zdroj dat, které budou následně importovány do databáze.

* dotazy

V této složce jsou umístěny soubory s dotazy (queries) na databázi. Tyto dotazy budou později využity k demonstrování funkčnosti a různých operací nad databází.

* funkcni\_reseni

Tato složka obsahuje všechno, co souvisí s nasazením a konfigurací řešení:

* + docker-compose.yml

Hlavní soubor pro orchestraci kontejnerů, který definuje jednotlivé služby – routery, config servery, shardy a další potřebné kontejnery (např. mongo-express pro vizuální správu databáze).

* + mongodb-build

Obsahuje konfigurační soubory potřebné pro sestavení vlastní image MongoDB.

* + - auth

Složka, ve které se nachází soubor s klíčem (např. mongoldb-keyfile) zajišťující bezpečnost při komunikaci mezi komponentami clusteru.

* + - Dockerfile

Konfigurační soubor definující postup sestavení image MongoDB, včetně potřebných úprav a závislostí.

* + scripts

Tato složka obsahuje řadu inicializačních a konfiguračních skriptů, které zajišťují automatizovanou konfiguraci celého clusteru. Mezi klíčové skripty patří:

* + - init-configserver.js – inicializace config serverů.
    - init-shard01.js, init-shard02.js, init-shard03.js – skripty pro konfiguraci jednotlivých shardů.
    - init-router.js – konfigurace routeru (mongos).
    - auth.js – skript pro nastavení autentizace mezi komponentami.
    - test-data.js – případný skript pro testování nebo import dodatečných dat.

### Docker-compose.yml

Ve docker-compose.yml jsou definovány tyto služby:

* **router01**, **router02**
  + Image: jin-mongo:6.0.2
  + Role: mongos router
  + Porty: 27117:27017, 27118:27017
  + Volumes: ./scripts:/scripts, perzistence dat
  + Popis: load‑balancing klientských požadavků
* **configsvr01**, **configsvr02**, **configsvr03**
  + Role: mongod --configsvr replSet rs-config-server
  + Porty: 27119:27017–27121:27017
  + Uchovávají metadata a konfiguraci clusteru
* **shard01-a/b/c**, **shard02-a/b/c**, **shard03-a/b/c**
  + Role: mongod --shardsvr replSet rs-shard-0X
  + Porty: 27122:27017–27130:27017
  + Zajišťují horizontální dělení dat a replikaci (RF=3)
* **mongo-express**
  + Image: mongo-express
  + Port: 8081:8081
  + Přístup: ME\_CONFIG\_MONGODB\_URL=mongodb://user:pass@router01:27017
  + Slouží pro vizuální správu databáze

Struktura volumes zajišťuje, že každý uzel má vlastní adresář pro data (/data/db) a konfigurační data (/data/configdb), což chrání data před ztrátou při restartu.

## Instalace

Instalace a spuštění celého řešení je zajištěno pomocí Docker Compose, což značně usnadňuje nasazení a automatizaci inicializačních procesů. Postup instalace je následující:

* Stažení a vytvoření kontejnerů:
  + Ujistěte se, že máte nainstalovaný Docker a Docker Compose.
  + V kořenovém adresáři projektu spusťte příkaz:

docker-compose up -d

* + Tento příkaz spustí všechny definované kontejnery (config servery, shardy, routery a pomocný kontejner pro mongo-express).
* Inicializace config serverů:
  + Po spuštění kontejnerů proveďte inicializaci config serveru spuštěním následujícího příkazu:

docker-compose exec configsvr01 bash "/scripts/init-configserver.js"

* + Tím se vytvoří a nakonfiguruje replikační soubor pro config servery.
* Inicializace shardů:
  + Pro každý ze tří shardů spusťte odpovídající inicializační skripty:

docker-compose exec shard01-a bash "/scripts/init-shard01.js"

docker-compose exec shard02-a bash "/scripts/init-shard02.js"

docker-compose exec shard03-a bash "/scripts/init-shard03.js"

* Konfigurace routeru:
  + Nakonfigurujte router spuštěním skriptu:

docker-compose exec router01 sh -c "mongosh < /scripts/init-router.js"

* + Tím se router správně připojí ke config serverům a připraví se na směrování dotazů.
* Nastavení autentizace:
  + Proveďte inicializaci autentizace jak na config serverech, tak na jednotlivých shardech:

docker-compose exec configsvr01 bash "/scripts/auth.js"

docker-compose exec shard01-a bash "/scripts/auth.js"

docker-compose exec shard02-a bash "/scripts/auth.js"

docker-compose exec shard03-a bash "/scripts/auth.js"

* Import dat:
  + Data z CSV souborů se importují do databáze pomocí nástroje mongoimport. Například pro import dat do kolekce TopAnime spusťte:

docker cp ../data/TopAnime.csv router-01:/home

docker exec -it router-01 bash -c "mongoimport -u user -p pass --authenticationDatabase admin -d MyDatabase -c TopAnime --file /home/TopAnime.csv --type csv --headerline"

* + Podobným způsobem importujte data z ostatních souborů (TopMovies.csv, TopNetflix.csv).
* Testování připojení:
  + Ověřte funkčnost celého řešení připojením ke clusteru:

docker-compose exec router01 mongosh --port 27017 -u "user" --authenticationDatabase admin

* + Po připojení lze provést testovací dotazy a ověřit, že je služba dostupná a správně distribuována.

Tento postup zajišťuje, že je celý cluster nakonfigurován a inicializován automaticky. Veškeré skripty a příkazy jsou integrovány do prostředí Docker Compose, což minimalizuje nutnost manuálních zásahů a umožňuje snadnou replikaci instalace v různých prostředích. Tímto je funkční řešení nasazeno, připraveno k importu dat a dalšímu testování dotazů.

# Případy užití a případové studie

## Obecné případy užití

MongoDB se nejčastěji uplatní v následujících scénářích:

* **Content Management** (CMS a publikační systémy) – pružné ukládání dokumentů a snadné rozšíření schématu.
* **Real‑time analytics** – agregace a analýza průběžně generovaných dat (telemetrie, IoT).
* **E‑commerce a katalogy produktů** – rychlé vyhledávání, filtrování a doporučování produktů.
* **Systémy s proměnlivou strukturou dat** – aplikace, kde se pole dokumentů často mění.
* **Geospatiální a časové dotazy** – ukládání a zpracování location-based a time-series dat.

## Případ použití

Pro semestrální projekt jsem si zvolil scénář zpracování žebříčků zábavy (TopAnime, TopMovies, TopNetflix). Mé řešení slouží pro:

* **Import** velkých CSV datasetů (>5 000 záznamů) do kolekcí MongoDB.
* **Rychlé CRUD operace** pro webové či mobilní rozhraní, které zobrazují nejpopulárnější tituly.
* **Agregační dotazy** (průměrná hodnocení, celkový počet hlasů) s nízkou latencí.

Zvolil jsem právě MongoDB z těchto důvodů:

* **Flexibilní dokumentový model** umožňuje snadno ukládat a rozšiřovat strukturu záznamů bez potřeby složitých migrací schématu.
* **Vestavěný agregation framework** poskytuje bohaté možnosti statistických a analytických dotazů přímo na databázové vrstvě.
* **Horizontální škálování** pomocí shardingu je jednoduché k nasazení a spravuje se automaticky mongos routery.
* **Primární sekundární replikace** s možností konfigurovat read preference pro škálování čtecích operací.
* **Bohatá podpora sekundárních indexů** (vč. geospatálních a textových), která zrychluje široké spektrum dotazů.
* **Silná komunita a ekosystém nástrojů** (mongo-express, Compass, MongoDB Atlas), což urychluje vývoj a deployment.

Ve srovnání s jinými databázemi:

* **Relační databáze** (např. PostgreSQL) by vyžadovaly pevné schéma a složité JOINy pro spojené žebříčky.
* **Key‑value store** (např. Redis) nedisponuje pokročilými dotazovacími možnostmi a agregacemi.
* **Wide‑column store** (např. Cassandra) není optimalizována pro ad‑hoc dotazy mimo primární klíč.
* **Dokumentové databáze** jako CouchDB mají omezenější možnosti shardingu a agregací.

Mé řešení je pro tento scénář ideální díky kombinaci dokumentového modelu, bohaté dotazovací vrstvy a jednoduché škálovatelnosti, kterou poskytuje MongoDB.

## Případové studie

### eBay: Globální Multi‑Data Center Architectures

Společnost eBay provozuje rozsáhlou infrastrukturu pro e‑commerce, která zahrnuje více než 3 000 instancí databází MongoDB, distribuovaných ve více geografických lokalitách po celém světě. Každá micro‑služba (např. katalog produktů, doporučovací systém a hledání) běží na samostatné instanci replica setu o 7 uzlech. Tyto uzly jsou členěny do tří datových center (DC) v Severní Americe, Evropě a Asii, přičemž v každém DC jsou minimálně dvě kopie dat a jeden arbiter. Pro distribuci shardů používá eBay **zone sharding**, kdy jsou rozsahy hodnot shard key přiřazeny konkrétním DC na základě regionálního provozu, čímž se minimalizuje latence a sdílení dat napříč kontinenty.

Nasazení zahrnuje:

* **7‑členný replica set**: primární a sekundární uzly s automatickým failoverem a priority settings pro zajištění konsistence.
* **Load balancing**: DNS round‑robin směruje klientské API požadavky na nejbližší dostupný mongos router.
* **Monitoring a alerty**: integrace s Prometheus a Grafana pro sledování metrik jako oplog lag, CPU, I/O.
* **Zálohování**: každodenní snapshooty do S3, inkrementální backup každou hodinu.

Díky této architektuře dosahuje eBay průměrné latence dotazů < 10 ms a SLA dostupnosti 99.99 %. V případě výpadku celého DC (např. plánovaná údržba nebo regionální blackout) se systém automaticky přepne na sekundární kopie, což zajišťuje kontinuitu služeb bez výpadků pro koncové uživatele. Zkušenosti týmu ukazují, že zone sharding výrazně zlepšuje lokalitu datových přenosů a snižuje náklady na meziregionální replikaci.

### Nokia Corteca Home Controller: Škálování na miliony zařízení

Platforma Nokia Corteca Home Controller zajišťuje správu domácích Wi‑Fi sítí pro koncové zákazníky v reálném čase. Před migrací využívala společnosti relační databáze, což vedlo k problémům s výkonem a náročností na schema migrations. Po přechodu na **MongoDB Atlas** se počet spravovaných zařízení zvýšil z původních 500 000 na více než 4,5 milionu s minimálním latencí.

Klíčové detaily nasazení:

* **Atlas Global Cluster**: tři zóny v regionálních oblastech USA, Evropy a Asie, každá se svým 3‑členným replica setem.
* **Time‑series collections**: použití vestavěného time‑series storage engine pro telemetry a statistiky, které automaticky agregují data do bucketů dle konfigurace (např. 1minutové intervaly).
* **RBAC a bezpečnost**: integrace s LDAP pro autentizaci, šifrování dat at rest pomocí AWS KMS, TLS pro data in transit.
* **Auto‑scaling**: nasazení automatického škálování sharding clusteru na základě CPU a I/O metrik z Performance Advisor.

Výsledky měření:

* **Snížení latence**: 95 % dotazů vykonáváno do 8 ms (oproti 50 ms na předchozím systému).
* **Snížení administrativních nákladů**: o 40 %, díky automatizovaným snapshotům a vestavěným backupům Atlas.
* **Dostupnost**: SLA 99.995 % v průběhu provozu.

### CERN CMS Data Aggregation System: Inteligentní cache pro vědecká data

Experiment CMS na CERN (Large Hadron Collider) generuje každoročně stovky PB dat z detektorů. Data Aggregation System (DAS) postavený nad MongoDB slouží jako centralizovaná cache a konsolidační vrstva pro metadata z různých zdrojů (RunSummary, DataQuality, LumiDB).

Konfigurace clusteru:

* **Hybridní shardovaný cluster**: 6 shardů pro různé typy dat, každý shard je replica set o 5 uzlech. Některé shardy používají **in‑memory storage engine** pro rychlé dotazy.
* **Capped collections**: pro časově řadové přenosy dat, kde je důležitá sekvenční velikost.
* **gridFS**: pro ukládání velkých souborů (např. výstupy analýz o velikosti stovek MB).
* **TTL indexy**: automatická expirace dat po konfiguraci (např. 30 dní pro dočasné telemetry).

Systém podporuje:

* **Vysokou propustnost**: průměrně 2 miliony dotazů denně, špičkově až 50 000 QPS.
* **Nízkou latenci**: < 5 ms pro klíčové RESTful API endpointy.
* **Spolehlivost**: díky replica setům a detekci chyb může cluster pokračovat v provozu i po výpadku 50 % uzlů v shardu.

Díky MongoDB DAS umožňuje vědcům rychle získat konzistentní metadata a výsledky měření bez potřeby přímého dotazování do primárních databází, což minimalizuje zátěž na primární systémy a urychluje výzkumné cykly.

# Výhody a nevýhody

V této kapitole hodnotím silné a slabé stránky MongoDB obecně a mého navrženého řešení.

## Výhody MongoDB

* **Flexibilní schéma** – dokumentový model umožňuje měnit strukturu dat bez složitých migrací.
* **Výkonný aggregation framework** – poskytuje širokou škálu operátorů pro transformaci a analýzu dat.
* **Horizontální škálovatelnost** – jednoduché přidání nových shardů bez downtime.
* **Vestavěná replikace** – zajištění vysoké dostupnosti a failover mechanismus.
* **Bohatá podpora indexů** – sekundární, geospatiální, textové a TTL indexy.
* **Široký ekosystém nástrojů** – Compass, mongo-express, backup/restore utility.

## Nevýhody MongoDB

* **Absence silných ACID transakcí napříč shardy** – transakce jsou omezené na single-shard nebo multi-document, ale mohou být pomalejší.
* **Vyšší režie storage engine** – WiredTiger používá kompresi a journal, což může zvýšit latenci zápisu.
* **Komplexita správy v multi-region architekturách** – nastavení zone sharding a balancování může být složité.
* **Omezená podpora JOINů** – join operace jsou možné pouze pomocí $lookup a mohou být náročné na výkon.
* **Memory-first engine** – při velkém zatížení může dojít k nedostatku paměti kvůli cache.

## Výhody mého řešení

* **Automatizace** – všechny kroky instalace a konfigurace jsou skripty integrovány do Docker Compose.
* **Jednoduché rozšíření** – přidání nových shardů nebo routerů vyžaduje minimální změny v konfiguraci.
* **Bezpečnost** – SCRAM‑SHA‑256 a keyFile s pevnými právy zajišťují omezený přístup.
* **Monitorovatelnost** – možnost snadné integrace s nástroji jako prometheus-exporter a Grafana.

## Nevýhody mého řešení

* **Chybějící centralizovaný monitoring** – neimplementoval jsem Prometheus/Grafana, což by zlepšilo observabilitu.
* **Ruční import dat** – skripty vyžadují manuální spuštění příkazů mongoimport, lze zautomatizovat dále.
* **Omezené testování odolnosti** – simulace výpadků uzlů nebyla plně pokryta testy failoveru.
* **Chybějící TLS mezi komponentami** – pro produkční nasazení je třeba přidat certifikáty pro šifrování.

# Další specifika

V tomto projektu jsem se držel výchozí recommended konfigurace MongoDB bez dalších customizací. Nepoužil jsem JSON Schema, TTL indexy, capped collections, multi‑document transakce, speciální indexy ani úpravy storage engine. Monitoring, audit logy i TLS šifrování mezi uzly zůstaly mimo rozsah této práce.

# Data

V této kapitole podrobně popisuji tři použité datasety, jejich formát, strukturu a metriky, předzpracování provedené před importem do MongoDB, základní analýzu dat pomocí Pythonu (Pandas, NumPy) a odkážím na přiložené skripty.

## Původ a formát datasetů

Datasety jsem stáhl z Kaggle:

* **TopAnime.csv** – "Top 15000 Ranked Anime Dataset". CSV s 22 sloupci a 15 000 záznamy. Kódování UTF‑8, hlavička v prvním řádku.

https://www.kaggle.com/datasets/quanthan/top-15000-ranked-anime-dataset-update-to-32025

* **TopMovies.csv** – "Movies Dataset TMDB Top Rated". CSV se 7 sloupci a 8 560 záznamy.

https://www.kaggle.com/datasets/muhammadtahir194/movies-dataset-tmdb-top-rated

* **TopNetflix.csv** – "Netflix Movies and TV Shows". CSV se 12 sloupci a 8 807 záznamy.

https://www.kaggle.com/datasets/shivamb/netflix-shows

## Struktura a chybějící hodnoty

**TopAnime.csv** (22 sloupců): např. anime\_id, name, english\_name (6 645 chybějících), score, popularity, members.

**TopMovies.csv** (7 sloupců): id, title, release\_date (2 chybějící), vote\_average, vote\_count, popularity, overview.

**TopNetflix.csv** (12 sloupců): show\_id, type, title, director (2 634 chybějících), cast (825 chybějících), country (831 chybějících), date\_added (10 chybějících), rating (4 chybějící), duration, release\_year, listed\_in, description.

## Předzpracování

Před importem do MongoDB jsem provedl tyto volitelné úpravy pro konzistenci a efektivní dotazy:

1. **Konverze typů**: pole release\_date, date\_added převedeno na datetime64[ns].
2. **Chybějící hodnoty**: textová pole doplněna prázdnými řetězci; v MongoDB lze zachovat null.
3. **Odebrání sloupců**: z TopAnime odstraněny synopsis, producers, studios (redukování velikosti dokumentů); z TopNetflix filtrovány pouze řádky s type == 'Movie'.
4. **Normalizace názvů**: převod názvů sloupců na malá písmena a podtržítka.

## Grafické výstupy a popis vybraných grafů

A graph of a number of blue bars

AI-generated content may be incorrect.Pro lepší pochopení datového obsahu a kvality dat jsem vytvořil řadu vizualizací, z nichž několik klíčových popisuji níže. Nejprve jsem porovnal velikost datasetů pomocí grafu *comparison\_records.png*. Tento sloupcový graf ukazuje, že dataset TopAnime obsahuje 15 000 záznamů, zatímco TopMovies a TopNetflix mají přibližně 8 558 a 8 807 záznamů. Tímto srovnáním je patrná výrazná rozdílná „bohatost“ zdrojů.

A graph with blue bars

AI-generated content may be incorrect.A graph with blue bars

AI-generated content may be incorrect.Kromě toho jsem detailně zkoumal rozložení chybějících hodnot v jednotlivých datasetech. Graf *TopAnime.csv\_missing.png* ukazuje rozložení chybějících hodnot v jednotlivých sloupcích TopAnime, přičemž nejvíce prázdných buněk se vyskytuje ve sloupcích english\_name, producers a studios. Podobně graf *TopNetflix.csv\_missing.png* vizualizuje chybějící hodnoty v TopNetflix, kde nejvíce neúplných dat je v polích director, cast a country, což naznačuje oblasti pro cílené čištění.

A graph with blue bars

AI-generated content may be incorrect.Pro hlubší vhled do struktury jednoho z datasetů ukazuji například graf TopAnime, kde *TopAnime.csv\_genres\_top10.png* prezentuje deset nejčastěji se vyskytujících žánrů: nejvíce titulů spadá do kategorií Action, Comedy a Drama. Následný graf TopAnime.csv\_missing.png zobrazuje rozložení chybějících hodnot v jednotlivých sloupcích TopAnime – zřetelně vyniká sloupec english\_name s 6 645 prázdnými položkami a dalšími polními chybějícími hodnotami ve sloupcích producers a studios.

A graph of different types of bars

AI-generated content may be incorrect.Pro ilustraci specifické kategorie jsem z datasetu TopAnime vybral rovněž graf *TopAnime.csv\_studios\_top10.png*, který ukazuje deset studií s nejvyšším počtem produkcí. Tento bar chart poukazuje na nejaktivnější studia, která vytvořila nejvíce anime titulů.

A graph of a vote

AI-generated content may be incorrect.V datasetu TopMovies jsem se zaměřil na distribuci hodnocení. Histogram *TopMovies.csv\_vote\_average\_hist.png* zobrazuje rozložení sloupce vote\_average, kde většina filmů dosahuje skóre mezi 6 a 8, což naznačuje konzistentní hodnocení oblíbenosti ve vybraném souboru.

A graph of blue bars with white text

AI-generated content may be incorrect.Dataset TopNetflix nabízí zajímavý pohled na geografii původu obsahu. Graf *TopNetflix.csv\_country\_top10.png* demonstruje deset zemí s největším počtem titulů, přičemž dominantním hráčem jsou Spojené státy americké, následované Indií a dalšími trhy. Současně jsem vytvořil graf TopNetflix.csv\_missing.png, který vizualizuje počet chybějících hodnot v jednotlivých sloupcích TopNetflix – největší díl připadá na sloupec director a dále na cast a country.

A graph with blue squares

AI-generated content may be incorrect.Konečně, graf *TopNetflix.csv\_type\_top10.png* ukazuje rozdělení typu obsahu (Movie vs TV Show) a potvrzuje, že naše filtrování zachytilo primárně filmy. Tyto vybrané vizualizace představují jen zlomek z mnoha vygenerovaných grafů, které dále zahrnují boxploty numerických sloupců, histogramy dalších polí a analýzu top kategorií pro textová pole.

## Skripty a upravené datasety Python skripty a upravená data Skripty a upravené datasety Python skripty a upravená data

Pro automatizaci úprav a generování grafů jsem vytvořil skript clean\_and\_analyze.py (umístěný v složce data/), který:

* Načte původní CSV.
* Provádí kroky z kapitoly 6.3.
* Uloží vyčištěné CSV do data/cleaned\_\*.csv.
* Vygeneruje PNG obrázky do složky plots/.

# Dotazy

Xxxxxx

Závěr

Xxxxxx

Zdroje

Xxxxxx

Přílohy

Xxxxxx