UNIVERZITA PARDUBICE   
Fakulta elektrotechniky a informatiky

*MongoDB*

*Oleksandr Mytro, 3. ročník*

V Pardubicích dne 26. dubna 2025

Obsah

[Úvod 4](#_Toc195810133)

[1 Architektura 5](#_Toc195810134)

[1.1 Schéma a popis architektury 5](#_Toc195810135)

[1.2 Specifika konfigurace 6](#_Toc195810136)

[1.2.1 CAP teorém 6](#_Toc195810137)

[1.2.2 Cluster 6](#_Toc195810138)

[1.2.3 Uzly 7](#_Toc195810139)

[1.2.4 Sharding 7](#_Toc195810140)

[1.2.5 Replikace 7](#_Toc195810141)

[1.2.6 Perzistence dat 8](#_Toc195810142)

[1.2.7 Distribuce dat 9](#_Toc195810143)

[1.2.8 Zabezpečení 11](#_Toc195810144)

[2 Funkční řešení 12](#_Toc195810145)

[2.1 Struktura 12](#_Toc195810146)

[2.2 Instalace 13](#_Toc195810147)

[3 Případy užití a případové studie 14](#_Toc195810148)

[3.1 Pro jaké účely je MongoDB vhodná 14](#_Toc195810149)

[3.2 Účel naší implementace a volba MongoDB 14](#_Toc195810150)

[3.3 Tři případové studie pro MongoDB 15](#_Toc195810151)

[3.3.1 Forbes – Modernizace CMS v mediální společnosti 15](#_Toc195810152)

[3.3.2 eBay – Škálování dat pro globální e-commerce 16](#_Toc195810153)

[3.3.3 Bosch Digital – IoT platforma pro velká data v průmyslu 19](#_Toc195810154)

[4 Výhody a nevýhody 23](#_Toc195810155)

[4.1 Obecné výhody a nevýhody MongoDB 23](#_Toc195810156)

[4.2 Výhody a nevýhody našeho řešení 23](#_Toc195810157)

[5 Další specifika 24](#_Toc195810158)

[6 Data 24](#_Toc195810159)

[6.1 Zdroje a popis dat 24](#_Toc195810160)

[6.1.1 Amazon – produktová data z Amazonu 24](#_Toc195810161)

[6.1.2 Google Play Store – data o mobilních aplikacích 25](#_Toc195810162)

[6.1.3 Video Game Sales – prodeje videoher 25](#_Toc195810163)

[6.2 Čištění a předzpracování dat 26](#_Toc195810164)

[6.3 Import dat do MongoDB 28](#_Toc195810165)

[6.4 Průzkumná analýza a základní statistiky 28](#_Toc195810166)

[6.5 Vizualizace dat a zjištění 30](#_Toc195810167)

[7 Dotazy 36](#_Toc195810168)

[Závěr 37](#_Toc195810169)

[Zdroje 39](#_Toc195810170)

[Přílohy 41](#_Toc195810171)

Úvod

Tato semestrální práce se zabývá návrhem a realizací distribuovaného databázového systému postaveného na MongoDB verze 6.0.2. Jejím cílem je ukázat principy fungování distribuované databáze v praxi – konkrétně efektivní ukládání a správu dat pomocí shardingu a replikace, jednoduché nasazení kontejnerizovaného řešení prostřednictvím Dockeru a automatizaci konfigurace.

Čtenář se v práci dozví podrobný popis architektury MongoDB klastru (shardy, konfigurační servery, routovací servery), postupy nastavení a spuštění jednotlivých komponent, příklady skriptů pro automatizaci nasazení a správy uživatelských rolí a autentizace. Dále se seznámí s importem dat (CSV, JSON) a s konkrétními praktickými ukázkami práce s třemi různými datasetů: produktů z Amazonu, aplikací z Google Play Store a prodejů videoher (VG Sales). Každý příklad ilustruje běžné operace CRUD, agregace i základní optimalizační techniky (indexování, shardování).

Do rozsahu této verze projektu nepatří výkonnostní testy ani srovnání MongoDB s relačními nebo jinými nerelačními databázemi (např. SQL databázemi). Tyto oblasti představují přirozené směry pro další rozšíření a prohloubení výzkumu v budoucnu.

# Architektura

Tato kapitola podrobně popisuje postup nasazení MongoDB jako NoSQL databáze v distribuovaném prostředí. Popsán je způsob nasazení databáze, její konfigurace v rámci klastru a praktické využití.

## Schéma a popis architektury

Изображение выглядит как текст, диаграмма, снимок экрана, План

Контент, сгенерированный ИИ, может содержать ошибки.

Navržená architektura MongoDB klastru se skládá ze tří hlavních komponent:

* **Shard servery**: Tři shardy, z nichž každý tvoří replikovaná sada (Replica Set). Každý shard obsahuje několik uzlů zajišťujících vysokou dostupnost dat a odolnost vůči selhání.
* **Konfigurační servery (mongod)**: Speciální sada tří serverů, které uchovávají informace o konfiguraci celého klastru. Tato konfigurace přispívá k odolnosti a zajištění dostupnosti dat.
* **Router (mongos)**: Jeden router fungující jako vstupní bod pro aplikace a uživatele. Směruje dotazy na příslušné shardy na základě konfigurace uložené na konfiguračních serverech.

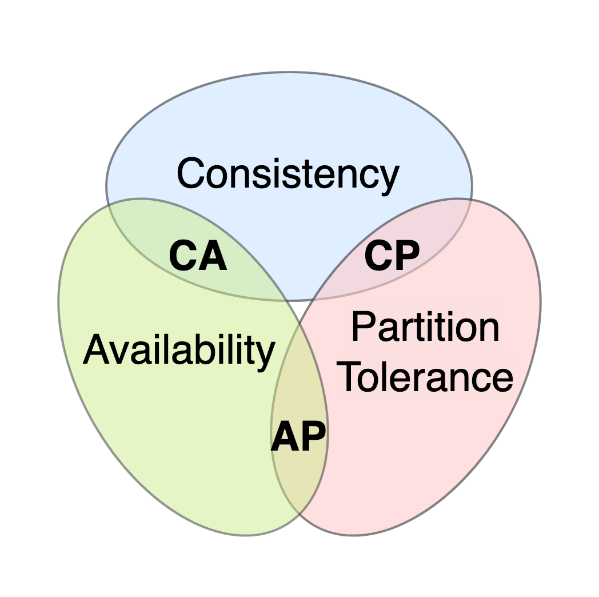
Zvolená architektura umožňuje efektivní distribuci zátěže mezi jednotlivé části databáze a poskytuje vysokou dostupnost a odolnost vůči výpadkům. Horizontální škálování prostřednictvím shardingu dovoluje přidávání dalších shardů při rostoucím objemu dat.

Použitá architektura vychází z doporučených standardů pro MongoDB klastry. Menší odchylky se mohou objevit v nastavení počtu uzlů, jejich velikosti nebo dalších parametrech, přizpůsobených dostupným zdrojům a požadavkům projektu.

## Specifika konfigurace

Tato část kapitoly rovněž detailně popisuje specifika konfigurace MongoDB klastru, včetně využití jednotlivých uzlů, implementace shardingu, replikace a zajištění bezpečnosti dat. Výběr konfigurace reflektuje dostupnost výpočetních prostředků, očekávaný objem dat a požadavky na spolehlivost a dostupnost databázového systému.

### CAP teorém



MongoDB jako distribuovaná databáze splňuje dvě hlavní vlastnosti dle CAP teorému: **dostupnost (Availability)** a **konzistenci (Consistency)**. Konzistence zajišťuje, že po úspěšném zápisu budou všechna data konzistentní napříč všemi uzly. Dostupnost znamená, že databáze odpovídá na každý požadavek i při nedostupnosti některých částí systému.

MongoDB nezaručuje odolnost vůči síťovým rozdělením (Partition tolerance) na stejné úrovni jako jiné systémy, avšak díky **replikaci** a **failover mechanismu** (automatickému přepnutí na jiný uzel) zůstává většinou funkční i při výpadcích.

Konfigurace klastru byla zvolena s ohledem na tyto priority:

* spolehlivý zápis a čtení dat,
* dostupnost systému i při selhání části infrastruktury,
* automatické zvládnutí výpadků bez nutnosti zásahu ze strany uživatele

### Cluster

V rámci řešení je nasazen MongoDB sharding cluster, který zahrnuje všechny klíčové komponenty:

* tři shardovací sady replik (**shard01**, **shard02**, **shard03**), kde každá sada tvoří samostatnou **replikační skupinu**,
* tři konfigurační servery (**configsvr1**, **configsvr2**, **configsvr3**), které uchovávají metadata o rozdělení dat mezi shardy,
* jeden router (**router1**), který přesměrovává dotazy na příslušné shard servery.

Všechna prostředí jsou nasazena jako **Docker** **kontejnery**, což zajišťuje snadnou správu, izolaci služeb a možnost jednoduchého horizontálního škálování přidáním dalších shardů nebo uzlů bez přerušení provozu.

### Uzly

Celkem je spuštěno **devět** **uzlů** v shardech a **tři** **uzly** konfiguračních serverů:

* Každý shard obsahuje jednu replikační sadu (**Replica** **Set**), která zahrnuje jeden **primární** a **dva** **sekundární** uzly (např. **shard01a**, **shard01b**, **shard01c**).
* Konfigurační servery také tvoří replikační sadu se **třemi** uzly, což zajišťuje vysokou dostupnost metadat.
* Router (**mongos**) neukládá žádná data, pouze směruje klientské dotazy k příslušným shardům a konfiguračním serverům.

Replikace umožňuje vysokou dostupnost a rozložení zátěže při čtení dat ze sekundárních uzlů, což zvyšuje celkovou efektivitu clusteru.

### Sharding

MongoDB implementuje horizontální shardování dat, což umožňuje distribuovat data mezi více serverů. Každý shard obsahuje část dat určenou shardovacím klíčem. V tomto řešení jsou nasazeny tři shardy: **shard01**, **shard02**, **shard03**.

Konkrétně se používají následující typy shardovacích klíčů:

* **Hashované** **klíče** (product\_id, \_id): umožňují rovnoměrné rozdělení dat pro kolekce **amazon**, **googleplaystore**, **reviews**.
* **Range** **klíče** (Name, Platform): umožňují efektivní provádění analytických dotazů pro kolekci **vgsales**.

Při růstu objemu dat provede MongoDB automatické rozdělení a distribuci chunků prostřednictvím interního balanceru.

### Replikace

Každý shard tvoří replikační sadu se **třemi** uzly (1× **primární**, 2× **sekundární**), což poskytuje vysokou dostupnost a ochranu dat proti ztrátě. Při výpadku primárního uzlu MongoDB **automaticky** aktivuje jeden ze sekundárních uzlů jako **nový** **primární** (tzv. **automatický** **failover**). Tento mechanismus zajišťuje nepřetržitý provoz a minimalizuje riziko výpadku služeb.

### Perzistence dat

**Způsob řešení persistence dat:**

* **Shardování:** Rozdělení dat na několik shardů (3 shardovací sady replik), což zajišťuje škálovatelnost a efektivní distribuci velkého objemu dat mezi více serverů. Shardování je založeno na hashovaných klíčích (např. **product\_id** nebo **\_id**) a na klíčích s rozsahem hodnot (např. kombinace **Name** a **Platform**).
* **Replikace:** Každý shard a konfigurační server fungují jako replikované sady, které se skládají z primární instance a dvou sekundárních instancí. Replikace zajišťuje vysokou dostupnost, zálohu dat a automatické zotavení v případě selhání.

**Použití primární a sekundární paměti:**

* **Primární paměť (RAM):** MongoDB aktivně využívá primární paměť pro ukládání pracovních sad dat, což urychluje přístup a zpracování operací s daty. Data jsou načítána do RAM podle potřeby dotazů a indexů, což minimalizuje diskové I/O operace.
* **Sekundární paměť (Disk):** Data jsou trvale uložena ve formě datových souborů a žurnálů na diskových jednotkách (volumes definovaných v Docker Compose). Každá instance (shardy, config servery, router) má svůj vyhrazený volume, který zachovává data a konfigurace.

**Načítání a ukládání dat:**

* **Načítání dat:** Probíhá přes MongoDB Router (mongos), který směruje dotazy na příslušné shardy podle shardovacího klíče. Čtení může probíhat z **primární** instance nebo ze **sekundárních replik** díky nastavení **readPreference=secondaryPreferred**, což pomáhá distribuovat zatížení databáze.
* **Ukládání dat:** Data se ukládají primárně přes **primární** instance jednotlivých shardů, které **synchronně** **replikují** změny na **sekundární** instance. Skripty pro import dat (**mongoimport**) jsou využity k hromadnému nahrávání datasetů (např. **amazon.csv**, **googleplaystore.csv**, **vgsales.csv**) přímo do databáze.

**Popis procesu:**

* Datové soubory jsou importovány pomocí skriptu **import\_datasets.sh**, který provádí načtení CSV souborů přímo do MongoDB clusteru přes MongoDB Router (**router-01**).
* Data jsou následně uložena na shardy v závislosti na shardovacích klíčích definovaných v inicializačním skriptu (**init\_cluster.sh**).
* Sekundární instance jsou pravidelně synchronizovány s primárními instancemi pomocí MongoDB replikace, čímž se udržuje integrita a aktuálnost dat napříč všemi replikami.

Díky tomuto systému zajišťujeme vysokou dostupnost, rychlost a spolehlivost ukládání a načítání dat.

### Distribuce dat

V našem řešení distribuce dat probíhá pomocí shardování a replikace, což zajišťuje optimální využití zdrojů a vysokou dostupnost dat.

**Shardování**

Data jsou rozdělena mezi tři shardy (**shard01, shard02, shard03**), přičemž každý shard uchovává část dat určenou shardovacím klíčem:

* **Hashované klíče** (např. product\_id, \_id) jsou využívány pro kolekce amazon, googleplaystore a reviews, což zajišťuje rovnoměrné rozložení dat.
* **Range klíče** (Name, Platform) jsou využívány pro kolekci vgsales, což usnadňuje analytické dotazy.

Изображение выглядит как снимок экрана, текст, схема

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как снимок экрана, текст, схема

Контент, сгенерированный ИИ, может содержать ошибки.

**Replikace**

Každý shard je tvořen replikační sadou se třemi uzly (1 primární, 2 sekundární). Při výpadku primárního uzlu MongoDB automaticky provede failover a zvolí sekundární uzel za nový primární, což zaručuje kontinuitu provozu.

**Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.**

**Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.** **Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.**

**Proces distribuce dat pro zápis a čtení**

* **Zápis dat** probíhá primárně přes primární uzly jednotlivých shardů. Data se synchronně replikují na sekundární uzly. Skript **import\_datasets.sh** používá MongoDB router (**router-01**) pro hromadné načítání dat z CSV souborů.

echo -e "${YELLOW}Importing amazon.csv into Ecommerce.amazon...${NC}"

mongoimport --host router-01 --port 27017 \

--username st69631 --password password --authenticationDatabase admin \

--db Ecommerce --collection amazon --type csv --headerline --file /data/amazon.csv

status=$?

echo -e "${YELLOW}Importing googleplaystore.csv into Ecommerce.googleplaystore...${NC}"

mongoimport --host router-01 --port 27017 \

--username st69631 --password password --authenticationDatabase admin \

--db Ecommerce --collection googleplaystore --type csv --headerline --file /data/googleplaystore.csv || status=1

echo -e "${YELLOW}Importing vgsales.csv into Ecommerce.vgsales...${NC}"

mongoimport --host router-01 --port 27017 \

--username st69631 --password password --authenticationDatabase admin \

--db Ecommerce --collection vgsales --type csv --headerline --file /data/vgsales.csv || status=1

echo -e "${YELLOW}Importing user reviews into Ecommerce.reviews...${NC}"

mongoimport --host router-01 --port 27017 \

--username st69631 --password password --authenticationDatabase admin \

--db Ecommerce --collection reviews --type csv --headerline --file /data/googleplaystore\_user\_reviews.csv || status=1

* **Čtení dat** je realizováno prostřednictvím MongoDB routeru (mongos). Speciálně pro náš projekt je čtecí preference nastavena pomocí:

readPreference=secondaryPreferred

Toto nastavení je definováno v souboru **.env**, který je použit službou **cli** v **docker-compose.yml**. Klient se připojuje k MongoDB přes proměnnou **MONGODB\_URI**, která zahrnuje tuto preferenci:

MONGODB\_URI="mongodb://st69631:password@router-01:27017/Ecommerce?authSource=admin&readPreference=secondaryPreferred"

Díky tomu mohou být dotazy směrovány nejen na primární, ale i na **sekundární** repliky, čímž se snižuje zátěž na primárních uzlech a zvyšuje celkový výkon systému.

**Automatické rozdělení dat (Balancer)**

MongoDB balancer automaticky sleduje a reguluje distribuci chunků mezi shardy. Tato funkcionalita je implicitně spuštěna během inicializace clusteru skriptem **init\_cluster.sh**, který stanovuje počáteční shardování a konfiguraci balanceru.

**init\_cluster.sh 🡫**

// Shard amazon (hashed, 8 chunks)

db.createCollection("amazon")

sh.shardCollection("Ecommerce.amazon", { product\_id: "hashed" }, false, { numInitialChunks: 8 })

// Shard googleplaystore (hashed, 4 chunks)

db.createCollection("googleplaystore")

sh.shardCollection("Ecommerce.googleplaystore", { \_id: "hashed" }, false, { numInitialChunks: 4 })

// Shard vgsales (range key) and pre-split

db.createCollection("vgsales")

sh.shardCollection("Ecommerce.vgsales", { Name: 1, Platform: 1 }, false)

// Manual pre-split for vgsales at Name="M" and Name="T"

sh.splitAt("Ecommerce.vgsales", { Name: "M", Platform: MinKey() })

sh.splitAt("Ecommerce.vgsales", { Name: "T", Platform: MinKey() })

// Shard reviews (hashed, 8 chunks)

db.createCollection("reviews")

sh.shardCollection("Ecommerce.reviews", { \_id: "hashed" }, false, { numInitialChunks: 8 })

**Počet záznamů na jednotlivých uzlech**

Pro ověření distribuce dat můžete použít následující příkazy:

// Počet dokumentů v jednotlivých kolekcích

db.amazon.countDocuments()

db.googleplaystore.countDocuments()

db.vgsales.countDocuments()

db.reviews.countDocuments()

Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

### Zabezpečení

Zabezpečení databáze je řešeno pomocí **autentizace a autorizace**. Všechny uzly klastru mají zapnutou autentizaci a využívají **keyFile** pro interní komunikaci. Tento soubor slouží jako sdílené tajemství mezi shardy, config servery a routerem a zabraňuje připojení neoprávněných instancí.

Byla vytvořena uživatelská role s oprávněními pro správu a manipulaci s daty. Přístup k databázi je umožněn pouze po zadání platných přihlašovacích údajů. Všechny akce lze auditovat, čímž je zajištěna bezpečnost a kontrola přístupu.

# Funkční řešení

Tato kapitola popisuje proces zprovoznění funkčního řešení a jeho vnitřní strukturu.

## Struktura

Projekt je rozdělen do dvou hlavních částí: adresáře **Data/** a složky **Funkční řešení/**.

* **Data/** obsahuje soubory amazon.csv, googleplaystore.csv a vgsales.csv. Tyto soubory jsou při spuštění automaticky importovány do MongoDB jako testovací data.
* **Funkční řešení/** představuje jádro celého projektu. Obsahuje:
  + **docker-compose.yml** – definice všech kontejnerů pro shardy, konfigurační servery a router,
  + **keyfile/** – složka s Dockerfilem a klíčem mongodb-keyfile pro zabezpečenou komunikaci v rámci MongoDB klastru,
  + **scripts/** – skripty pro inicializaci klastru, nastavení autentizace uživatelů a import dat (např. import\_datasets.sh, init\_shardxx.js a další ověřovací nebo testovací soubory).

Tato struktura umožňuje jednoduché a přehledné zprovoznění systému pomocí jednoho příkazu s využitím Dockeru.

## Instalace

Pro spuštění řešení je nutné mít nainstalovaný **Docker** a **Docker Compose**. Celé řešení bylo navrženo tak, aby bylo možné jej zprovoznit jediným příkazem, bez potřeby ručního spouštění pomocných skriptů.

**Postup instalace a nasazení:**

1. **Klonujte repozitář do cílového adresáře:**

git clone <repo-url> && cd <repo-folder>

1. **Spusťte celý systém pomocí jediného příkazu:**

docker compose up -d

Tento příkaz:

* vytvoří a spustí všechny potřebné kontejnery (router, config servery, shardy),
* spustí skript **init\_cluster.sh** pro automatickou inicializaci clusteru a nastavení replikace,
* nastaví autentizaci a vytvoří administrátorského uživatele,
* pomocí **init-data** kontejneru spustí skript **import\_datasets.sh**, který automaticky načte vstupní data do databáze,
* po dokončení je systém plně připraven k použití, není potřeba žádný ruční zásah.

Docker Compose využívá **depends\_on** a příznaky **condition**: **service\_completed\_successfully**, čímž zajišťuje, že jednotlivé části systému startují ve správném pořadí a s garantovaným dokončením předchozích kroků.

1. **Práce s databází**

Chcete-li se připojit do MongoDB routeru s nastaveným **readPreference=secondaryPreferred**, spusťte:

docker compose run --rm cli

Tento příkaz spustí interaktivní **mongosh** shell s přednastaveným připojením přes router na sekundární repliky.

# Případy užití a případové studie

Tato kapitola prezentuje praktické příklady využití MongoDB, oblasti vhodného nasazení NoSQL databází a důvody volby tohoto konkrétního systému. Součástí jsou i tři případové studie reálného nasazení.

## Pro jaké účely je MongoDB vhodná

MongoDB jako NoSQL databáze založená na dokumentovém modelu (formát podobný JSON) se hodí pro následující scénáře:

* ukládání a zpracování polo-strukturovaných a nestrukturovaných dat,
* rychlé zápisy a čtení dat i ve velkém měřítku,
* horizontální škálování pomocí shardingu,
* flexibilní schéma bez nutnosti předdefinovaných struktur tabulek.

## Účel naší implementace a volba MongoDB

Datové sady zahrnují informace o produktech z Amazonu, aplikacích z Google Play Store a datech o prodejích videoher (VG Sales). Tyto datasety jsou strukturované a polo-strukturované, což je činí vhodnými pro ukládání v dokumentové databázi.

Důvody volby MongoDB:

* potřeba snadného importu rozsáhlých CSV souborů a práce s nimi v průběhu provozu,
* očekávaný růst dat, který vyžaduje horizontální škálovatelnost,
* vestavěná podpora pro replikaci a sharding pro dosažení vysoké dostupnosti,
* flexibilní dokumentový model, který umožňuje uložení různorodých struktur bez nutnosti úprav schématu.

Alternativní NoSQL databáze nebyly zvoleny z důvodu nevhodnosti pro daný případ:

* **Key-Value databáze** neumožňují komplexní dotazování,
* **grafové databáze** by byly zbytečně složité pro daný typ dat,
* **sloupcově orientované databáze** nejsou optimální pro práci s polo-strukturovanými daty.

MongoDB zde představuje efektivní kompromis mezi flexibilitou, výkonností a jednoduchostí použití.

## Tři případové studie pro MongoDB

### Forbes – Modernizace CMS v mediální společnosti

**Kontext**

**Forbes** je globální mediální společnost známá svým obchodním zpravodajstvím a žebříčky. Působí v oblasti médií již přes sto let a dnes její digitální obsah oslovuje kolem **140 milionů uživatelů měsíčně po celém světě**. Forbes provozuje jeden z nejnavštěvovanějších zpravodajských webů a zároveň rozvíjí elektronický obchod. V roce 2011 se rozhodl přejít na moderní publikační platformu a začal využívat tehdy novou NoSQL databázi MongoDB pro svůj redakční systém (CMS). Forbes tak odstartoval digitální transformaci, která mu pomohla udržet pověst inovátora v publikačním průmyslu.

**Důvody pro výběr MongoDB**

Původní CMS Forbesu běžel on-premise (na vlastním hardware) a postupem času přestal vyhovovat nárokům redaktorů i čtenářů. **Projevovaly se všechny typické problémy lokálního řešení** – náročná údržba serverů, potřeba ručních zásahů (například výměna disků v datacentru) a komplikované nasazování aktualizací​([mongodb.com](https://www.mongodb.com/solutions/customer-case-studies/forbes#:~:text=%E2%80%9CAll%20the%20problems%20that%20come,%E2%80%9D)). Systém nebyl dostatečně flexibilní pro rychlé zavádění nových funkcí a integrací. Forbes proto v roce 2019 dospěl k rozhodnutí provést **zásadní modernizaci platformy**. Hledali **spravované cloudové řešení**, které by zjednodušilo replikaci databáze, škálování a správu infrastruktury. Tyto požadavky nejlépe splňovala služba **MongoDB Atlas**, jež nabízí plně spravovanou multi-cloud databázi([mongodb.com](https://www.mongodb.com/solutions/customer-case-studies/forbes#:~:text=Forbes%20wanted%20a%20managed%20solution,with%20MongoDB%20Atlas%20in%202019))​. MongoDB vyniklo nad konkurencí svým **vývojářsky přívětivým přístupem** – přechod ze světa SQL byl relativně snadný a tým mohl využít stávající znalosti. Důležitým faktorem byla i **flexibilita datového modelu** MongoDB (dokumentově orientovaná databáze), který umožňuje rychle iterovat a přizpůsobovat se měnícím se požadavkům.

**Technické detaily implementace**

Forbes zvolil **MongoDB Atlas** (databáze jako služba) nasazený v rámci Google Cloud Platform pro svůj nový CMS. Přechod do cloudu byl velmi rychlý – podařilo se jej dokončit za pouhých šest měsíců. Klíčovou výhodou Atlasu byla **jednoduchá instalace a správa**: databázová infrastruktura se dala zprovoznit během minut a Atlas zajišťuje automatickou replikaci dat, zálohování i škálování podle potřeby. Forbes rovněž využil **odborné konzultace MongoDB Professional Services** a program **MongoDB University** k rychlému zaškolení svých vývojářů, čímž hladce zvládli přechod na nový databázový systém. **Aplikační vrstva (API) i databáze byly zcela orchestrace v cloudu**, což značně usnadnilo nasazení a provoz. Původní fulltextový vyhledávač Apache Solr se rozhodli postupně nahradit funkcionalitou **Atlas Search**, tedy vestavěným vyhledáváním přímo v MongoDB, aby se zjednodušila architektura a zrychlilo vyhledávání([mongodb.com](https://www.mongodb.com/solutions/customer-case-studies/forbes#:~:text=of%20apps%20and%20tools%2C%20and,from%20Solr%20to%20Atlas%20Search))​. Díky **automatickému škálování úložiště** a výkonu Atlasu se Forbes nemusí starat o kapacitní plánování – systém se přizpůsobuje aktuální zátěži, což je obrovská výhoda pro vývojáře i administrátory.

**Výsledky nasazení MongoDB**

Přechod Forbesu na MongoDB Atlas přinesl **dramatické zlepšení výkonu i efektivity vývoje**. Průměrné časy buildů (sestavení a nasazení nového kódu) se **zkrátily z 25 minut na pouhých 9 minut**, což představuje zrychlení o cca **58 %**. Nasazování nových verzí softwaru je nyní **4× rychlejší** než dříve. Modernizace také snížila **celkové náklady na vlastnictví (TCO) o 25 %** díky omezení provozu vlastních serverů. Flexibilita MongoDB umožnila Forbesu rychle zavádět nové funkce – například personalizovaná doporučení článků pomocí AI – a lépe analyzovat chování čtenářů v reálném čase. Tyto inovace vedly k **27–28% nárůstu počtu předplatitelů** (zejména díky novým newsletterům). Kromě číselných ukazatelů pocítili zlepšení i koncoví uživatelé: web Forbes nyní načítá stránky rychleji a nabízí pohotovější zážitek. Stabilní výkon během enormního nárůstu návštěvnosti (například v době pandemie COVID-19) potvrdil, že cloudové řešení dokáže **automaticky škálovat** a udržet dostupnost i při rekordním provozu. Jak shrnul CTO Forbesu, Vadim Supitskiy, přesun na Atlas a Google Cloud umožnil týmu **“inovovat a prosperovat i v těch nejtěžších podmínkách”**.

**Zajímavosti a výzvy při integraci**

Jednou z výzev byla **změna myšlení vývojářů** při přechodu z relační databáze na dokumentovou. Díky školení však tým rychle zvládl využít flexibilní model MongoDB bez pevného schématu. Forbes také zjistil, že Atlas nabízí nové možnosti, např. integrované vyhledávání (Atlas Search), které plánují plně využít místo externího Solru([mongodb.com](https://www.mongodb.com/solutions/customer-case-studies/forbes#:~:text=of%20apps%20and%20tools%2C%20and,from%20Solr%20to%20Atlas%20Search))​. **Migrace stávajících dat** do MongoDB Atlas proběhla postupně a bez větších problémů – klíčové bylo zajistit kompatibilitu a optimalizaci dotazů, k čemuž pomohli experti z MongoDB. Během integrace se ukázalo, že **přenesení dříve on-premise databáze do cloudu** výrazně zjednodušilo údržbu: odpadla starost o hardware, zálohy i aktualizace, což uvolnilo ruce vývojářům k inovacím. Forbes nyní pokračuje ve vývoji nové funkcionality – kromě zmíněného vyhledávání plánuje dále rozvíjet svou platformu pro předplatné a personalizaci obsahu s podporou MongoDB. Případ Forbesu tak ukazuje, že i **tradiční mediální dům může díky MongoDB a cloudu projít úspěšnou digitální transformací**, zvýšit výkon systému a rychleji přinášet novinky svým uživatelům.

### eBay – Škálování dat pro globální e-commerce

**Kontext**

**eBay** je jedním z největších internetových tržišť na světě, působí v odvětví e-commerce (online maloobchod). S více než **170 miliony aktivních kupujících a 1 miliardou aktivních aukcí v 190 zemích** patří eBay mezi top 10 globálních maloobchodních značek([mongodb.com](https://www.mongodb.com/blog/post/ebay-building-mission-critical-multi-data-center-applications-with-mongodb#:~:text=As%20a%20top%2010%20global,facing)). Pro takovou platformu je kritická **nepřetržitá dostupnost** a rychlá odezva systému – i krátký výpadek by mohl ovlivnit obrovské množství transakcí. eBay proto investuje do robustní infrastruktury a mimo jiné již řadu let využívá MongoDB jako jednu ze svých klíčových databázových technologií. MongoDB zde funguje jako **součást centrální datové platformy**, která pohání vícero aplikací přímo na webu eBay.

**Důvody pro výběr MongoDB**

eBay kombinuje různé databázové systémy (relační i NoSQL) podle konkrétních potřeb aplikací. Pro některé úlohy ale naráželi na limity tradičních SQL databází z hlediska výkonu a škálovatelnosti. Například funkce **našeptávání vyhledávání** (vyhledávací našeptávač) vyžaduje, aby dotaz do databáze a vrácení návrhů proběhlo do **60–70 milisekund**, jinak nebude uživatelský dojem uspokojivý. Taková odezva je s relační databází obtížně dosažitelná při obrovském objemu dat eBay. Společnost proto experimentovala s dokumentovou databází MongoDB, aby zjistila, zda dokáže potřebný výkon zajistit. Důležitým faktorem výběru MongoDB byla i **flexibilita datového modelu** – eBay potřeboval ukládat různorodá data, např. metadata ke všem produktovým fotografiím, kde pevné schéma relační DB překáželo. V rámci projektu Zoom ukládá eBay **veškerá média a metadata k produktům do MongoDB**, právě kvůli pružnému modelování dat a lepší výkonové charakteristice. MongoDB navíc nabízí **vestavěnou replikaci a vysokou dostupnost**: jako distribuovaná databáze zvládá provoz ve více datacentrech, což bylo pro eBay klíčové k dosažení odolnosti vůči výpadkům. S postupným zráním NoSQL technologií eBay zjistil, že MongoDB již podporuje i pokročilé funkce (např. konzistentní zálohy v čase) a je dostatečně spolehlivé, aby mohlo sloužit i pro kritická data (tzv. system of record)([mongodb.com](https://www.mongodb.com/blog/post/ebay-building-mission-critical-multi-data-center-applications-with-mongodb#:~:text=database%20landscape%20has%20matured,Record%20use%20cases%20at%20eBay))​. Kombinace **škálovatelnosti (sharding)**, **výkonu** a **schopnosti zpracovat neustále se měnící datové formáty** učinila z MongoDB logickou volbu pro více klíčových služeb eBay.

**Technické detaily implementace**

eBay provozuje **tisíce MongoDB instancí** v různých clusterech – celkově přes **3 000 databázových uzlů NoSQL** spravujících dohromady **několik petabajtů dat**. Architektura je navržena s důrazem na **odolnost a škálování**. Jedním z osvědčených vzorů je nasazení MongoDB v podobě **replikační sady o 7 uzlech** rozprostřených mezi více datacentrům. Tato konfigurace zajišťuje, že i při výpadku více uzlů (či celého datacentra) zůstane databáze dostupná a automaticky zvolí nové primární replikační uzly. Pro extrémní nároky na výkon eBay využívá také **škálování do paměti** – některé datové sady uchovává v RAM, čímž dosahuje mimořádně rychlých odpovědí. Například seznamy návrhů vyhledávání byly v MongoDB navrženy tak, že každý možný dotaz (slovní prefix) je uložen jako dokument doplněný o metadata (např. kategorie produktů). Na tyto dokumenty byly vytvořeny **indexy podle prefixů i kategorií**, což umožňuje velice rychlé vyhledání relevantních návrhů. Tato sbírka pro našeptávač běží na **jediné replikační sadě bez shardingu**, přičemž datový set je uložen v paměti pro maximální rychlost. Díky tomu MongoDB dokáže vracet návrhy během **1,4 ms** – tedy hluboko pod požadovaným limitem 70 ms. Kde je potřeba horizontálně škálovat, eBay využívá **sharding (rozdělení dat mezi více uzlů)**. V minulosti eBay dokonce vyvinul vlastní aplikační vrstvu pro sharding nad některými NoSQL databázemi, aby zmírnil výkonnostní dopady rebalancování dat po výpadku uzlu. U MongoDB však vestavěný auto-sharding spolu s replikací pokrývá většinu potřeb, což zjednodušuje architekturu. eBay rovněž dbá na **zálohování a testování obnovy** – díky funkcionalitám MongoDB může dělat konzistentní zálohy a v případě potřeby rychle obnovit data do určitého bodu v čase​([mongodb.com](https://www.mongodb.com/blog/post/ebay-building-mission-critical-multi-data-center-applications-with-mongodb#:~:text=database%20landscape%20has%20matured,Record%20use%20cases%20at%20eBay)).

**Výsledky nasazení MongoDB**

MongoDB v eBay umožnil realizovat funkcionality, které by jinak byly obtížné nebo nemožné. **Vyhledávací našeptávač** nyní funguje s bleskovou odezvou – návrhy se uživateli zobrazují prakticky okamžitě, do **2 milisekund** od zadání dotazu. Tím se zlepšila uživatelská zkušenost a pravděpodobnost, že uživatel rychle najde, co hledá. Díky **automatickému shardingu** může eBay snadno škálovat své databáze s růstem počtu uživatelů a dat, aniž by bylo třeba složitě redesignovat aplikace – MongoDB se stará o rovnoměrné rozložení dat a zátěže. To se ukázalo jako zásadní například u online her od EA (Electronic Arts), kde MongoDB v prostředí s miliony hráčů prokázalo, že dokáže jednoduše přidávat další uzly při nárůstu popularity hry​ (eBay takovou možnost využívá pro své rostoucí množství dat z transakcí). Nasazením MongoDB eBay také dosáhl **vyšší dostupnosti**: i při poruše hardware nedochází k výpadku služby, replikační sady zajistí pokračování provozu. Interní testy prokázaly, že MongoDB splňuje přísné požadavky na konzistenci a obnovu dat, takže mu eBay začal svěřovat i více kritických datových úloh([mongo.com](https://www.mongodb.com/blog/post/ebay-building-mission-critical-multi-data-center-applications-with-mongodb))​. Důležitým přínosem je rovněž **rychlejší vývoj** – vývojáři mohou využívat flexibilní schéma a rychleji měnit datové modely podle potřeb nových funkcí, bez zdlouhavých migrací jako v případě SQL databází. Celkově MongoDB pomohl eBay udržet krok s obrovským růstem platformy a očekáváním zákazníků na **rychlé a spolehlivé služby** v globálním měřítku.

**Zajímavosti a výzvy během integrace**

Integrace MongoDB v eBay přišla postupně – nejprve v rámci izolovaného projektu (uložiště obrázkových metadat), následně pro vyhledávací našeptávač a další služby. Jednou z výzev bylo nastavení optimální **architektury pro vysokou dostupnost**. eBay musel pečlivě navrhnout geograficky distribuované repliky, aby dosáhl skutečné odolnosti proti výpadkům regionů. Také bylo nutné **přizpůsobit monitorování a správu** – eBay vyvinul sadu Resilience Design Patterns, osvědčených postupů, jak udržet databázové služby MongoDB co nejdéle v provozu (maximalizovat MTTF) a v případě problému je co nejrychleji obnovit (minimalizovat MTTR). Další zajímavostí je, že eBay porovnával MongoDB s jinými NoSQL databázemi. Některé z nich (s **peer-to-peer architekturou bez master uzlu**) sice nabízejí snadné škálování, ale po výpadku uzlu vyžadují náročné rebalance dat, což dočasně snižuje výkon. MongoDB díky modelu s primárním/sekundárními uzly tyto situace lépe zvládá a eBay tak nemusel řešit tak rozsáhlé výpadky výkonu při obnově. Během integrace se tým eBay také učil **nejlepším praktikám pro modelování dat v MongoDB** – například zjistili, že pro některé dotazy je výhodnější denormalizovat data do jednoho dokumentu (jako u našeptávače), místo dělat více dotazů. Vývojáři eBay se postupně vzdělali (často i pomocí bezplatných online kurzů MongoDB University podobně jako jinde) a dokázali přejít na “MongoDB způsob myšlení”, včetně změny přístupu od SQL relací k dokumentově orientovaným dotazům. Případ eBay tak demonstruje, jak lze MongoDB úspěšně nasadit v **mission-critical aplikacích globálního rozsahu**, zajistit špičkový výkon (v řádu milisekund) a škálovat na stovky milionů uživatelů.

### Bosch Digital – IoT platforma pro velká data v průmyslu

**Kontext**

**Bosch** je přední světová technologická a výrobní společnost působící ve více odvětvích – od automobilového průmyslu a průmyslové automatizace po spotřební zboží (domácí spotřebiče, elektronářadí) a Internet věcí (IoT). V rámci koncernu Bosch působí divize **Bosch Digital**, která se zaměřuje na digitální inovace a práci s daty z IoT zařízení. Bosch Digital provozuje službu **Bosch IoT Insights**, jež sbírá a analyzuje obrovské objemy dat z připojených zařízení po celém světě. IoT data přicházejí z různých odvětví – například maloobchod, energetika, zemědělství, chytrá mobilita či průmyslová výroba. Pro představu: Bosch IoT Insights zpracovává data z výrobních nástrojů, které utahují **6 milionů šroubů a matic při montáži jednoho letadla**, přičemž každý údaj o momentu a úhlu utažení je kriticky důležitý. Dalším zdrojem dat jsou elektronické řídicí jednotky (ECU) v automobilech, které zaznamenávají chování řidiče a vozu – tato anonymizovaná telemetrie pomáhá vývojářům vylepšovat konstrukci vozidel. Bosch Digital tak stojí před úkolem **uchovávat a zpracovávat ohromné objemy různorodých dat v reálném čase**, a proměnit je v užitečné poznatky pro inženýry i zákazníky.

**Důvody pro výběr MongoDB**

Aby Bosch Digital dokázal z IoT dat vytěžit maximum, potřeboval databázovou platformu, která zvládne **velmi škálovatelné, výkonné a flexibilní** zpracování dat. Tradiční přístupy, jako ukládání dat do objektového úložiště, nebyly dostatečně operativní – **vyžadovaly zdlouhavé načítání dat před každou analýzou**, což stálo čas a peníze​([mongodb.com](https://www.mongodb.com/solutions/customer-case-studies/bosch#:~:text=time,We%20can%20perform%20live)). Bosch hledal řešení, které umožní data **okamžitě dotazovat** a analyzovat bez zbytečných mezikroků. Proto se rozhodli pro **MongoDB**, které dokáže spravovat obrovské objemy dat a **vytvářet sjednocený pohled na data z více zdrojů v reálném čase(**[mongodb.com](https://www.mongodb.com/solutions/customer-case-studies/bosch#:~:text=time,We%20can%20perform%20live)**)**​. Klíčovou výhodou MongoDB je **flexibilní dokumentový model (BSON/JSON)**, jenž umožňuje přirozeně ukládat strukturovaná i nestrukturovaná data z IoT senzorů. Data z každého typu zařízení lze mapovat do odpovídajícího dokumentu v databázi a **není nutné dopředu definovat pevné schéma**, které by omezovalo budoucí rozšiřování. Tato **dynamická schémata** podporují agilní vývoj – projekty v Bosch Digital si samy definují datové struktury, které potřebují, a platforma je schopna je obsloužit. Pokud se požadavky časem změní, datový model se jednoduše upraví, aniž by to rozbilo ostatní části systému. Bosch rovněž ocenil, že MongoDB nabízí **podporu různých typů dotazů a indexů** – od sekundárních indexů přes fulltextové vyhledávání až po geospatial dotazy – což je pro IoT data nezbytné (mohou potřebovat filtrovat data podle času, polohy, typu senzoru apod.). Neméně důležitá byla i otázka **škálování a dostupnosti**: MongoDB v enterprise verzi nebo Atlasu zajišťuje vysokou dostupnost (replikace s automatickým fail-over) a horizontální škálovatelnost na cluster více uzlů, a to i v multi-cloud prostředí. Pro Bosch, který spolupracuje s různými cloudovými poskytovateli (např. Microsoft Azure), byla **nezávislost na konkrétním cloudu** a možnost nasadit databázi kdekoli zásadní výhodou. Celkově tedy Bosch zvolil MongoDB kvůli **rychlosti zpracování velkých dat, flexibilitě pro heterogenní IoT data, snadnému vývoji bez komplexní správy schémat a robustní škálovatelnosti a dostupnosti**.

**Technické detaily implementace**

Bosch Digital nasadil MongoDB jako **centrální datové úložiště** pro platformu IoT Insights. Všechny příchozí datové toky ze senzorů a zařízení jsou ukládány v reálném čase do MongoDB, kde jsou **okamžitě k dispozici pro analýzu**. Databáze je navržena tak, aby pojmula **ohromné objemy časových řad dat** – k tomu Bosch hodlá využít i specializované **Time Series Collections** v MongoDB (kolekce optimalizované pro časové řady), jakmile budou plně dostupné, což přinese ještě efektivnější ukládání a dotazování nad časovými daty. Aktuálně tým Bosch spravuje indexy a strukturu dat manuálně; například pokud se některé dotazy začnou zpomalovat, **přidají vhodný index**, aby se výkon vrátil na požadovanou úroveň. V MongoDB využívají bohaté možnosti agregací – **Aggregation Framework a MapReduce** – které umožňují nad nasbíranými daty provádět složité analytické dotazy a výpočty přímo v databázi. To znamená, že mnoho analytických úloh lze realizovat bez potřeby exportovat data do externích systémů, což šetří čas. Dokumentová struktura (JSON) usnadňuje vývojářům práci – data z IoT zařízení často přicházejí v JSON formátu, takže je lze **bez velkých transformací uložit přímo do MongoDB**. Přidávání nových typů zařízení či senzorů je jednoduché: stačí začít ukládat jejich data, aniž by bylo nutné upravovat centrální schéma databáze. MongoDB tak **odstraňuje neflexibilitu tradičních SQL schémat** a vývojáři se mohou soustředit na logiku aplikací místo na migrační skripty. Pro zajištění vysoké dostupnosti a výkonu v globálním měřítku pravděpodobně Bosch využívá víceuzlové replikační sady či cluster MongoDB Atlas (případová studie zmiňuje enterprise-grade availability a zero downtime). Důraz je kladen i na **bezpečnost a compliance** – MongoDB umožňuje pokročilé zabezpečení dat (šifrování, přístupová práva), což je pro citlivá průmyslová data nutností. Bosch Digital také těsně spolupracuje s dodavatelem: **MongoDB Inc. naslouchá zpětné vazbě** a vylepšuje produkt v dalších verzích podle potřeb zákazníků, navíc má Bosch přímou podporu od MongoDB týmu, kdykoli je třeba vyřešit komplikace. Celá platforma je navržená tak, aby byla **cloudově neutrální** – MongoDB lze provozovat jak on-premise, tak v různých cloudech, a Bosch plánuje využít **Azure** pro ukládání zákaznických dat, což MongoDB bez problémů umožňuje díky podpoře multi-cloud nasazení.

**Výsledky nasazení MongoDB**

Díky MongoDB dokáže Bosch Digital poskytovat svým zákazníkům a inženýrům **aktuální a akceschopné poznatky z velkých dat** rychle a efektivně. Platforma IoT Insights nyní **snadno dává smysl obrovským objemům dat** – ať už jde o údaje ze strojů v továrně, nebo z vozidel na silnicích – a to napříč různými use-casy. **Flexibilita MongoDB výrazně zkrátila dobu nutnou k nasazení nových projektů**: jednotlivé týmy nemusí čekat na návrh a vytvoření nových databázových schémat centrálním oddělením, prostě definují data, která chtějí ukládat, a MongoDB je dokáže obsloužit. To přináší **agilitu** – pokud se v průběhu vývoje zjistí, že je potřeba ukládat další atribut senzoru, lze ho přidat do dokumentu bez nutnosti migrace celé databáze. **Zákazníci Bosch** díky tomu mohou **experimentovat s daty** – otestovat, jak určitá struktura či dotaz funguje, ještě než se definitivně rozhodnou pro konkrétní datový model. MongoDB tak podporuje kulturu fail-fast a učení se z iterací, což je v oblasti IoT a big data velmi cenné. Vzhledem k rostoucímu významu **machine learningu a AI** hraje MongoDB důležitou roli i v Bosch: umožňuje uchovávat a rychle dotazovat historická data, na kterých lze trénovat modely a dělat prediktivní analýzy. Bosch hlásí, že s MongoDB mohou **“inovovat mnohem rychleji s menším úsilím”**, protože mohou stavět na předešlých zkušenostech a nemusejí vše začínat od nuly. Platforma postavená na MongoDB také dosahuje **podnikově požadované spolehlivosti a škálovatelnosti** – je schopná nepřetržitého provozu bez výpadků (zero downtime) i při velkém objemu dat, splňuje vysoké bezpečnostní standardy, a zároveň ekonomicky roste spolu s daty (náklady rostou lineárně s přidáváním kapacity, nejsou tam skokové investice do monolitických databází). To vše znamená, že Bosch může **odemykat nové zdroje příjmů** a optimalizovat své produkty a služby využitím dat, aniž by musel výrazně zvyšovat zátěž na své vývojové týmy či investovat do komplikované infrastruktury. Stručně řečeno, MongoDB pomohlo Bosch Digital proměnit IoT data ve skutečnou konkurenční výhodu – **případy využití velkých dat jsou prakticky neomezené a MongoDB umožňuje realizovat každý z nich**.

**Zajímavosti a výzvy během integrace**

Při zavádění MongoDB do IoT platformy Bosch musel tým čelit několika výzvám. Jednou z nich bylo **zvládnutí enormní rychlosti a objemu IoT dat** – senzory mohou posílat údaje v taktu minut či dokonce sekund, a systém je musel pojmout. MongoDB se osvědčilo svou schopností rychlých zápisů i čtení ve velkém paralelismu, a Bosch případně škáluje cluster přidáním uzlů, pokud objem dat roste nad rámec stávající kapacity. Zajímavostí je, že Bosch Digital pomáhá svým zákazníkům (interním i externím) s přechodem na nový způsob práce s daty: poskytují **podporu jak psát SQL-like dotazy v MongoDB** nebo jak efektivně vyhledávat. To naznačuje, že jednou z výzev bylo i **vzdělávání uživatelů, kteří byli zvyklí na SQL** – museli se naučit využívat agregace a indexy v MongoDB. Bosch tuto výzvu řeší proaktivně tím, že monitoruje dotazy a pokud jsou pomalé, navrhne úpravu datového modelu či vytvoření indexu. Dalším specifikem IoT řešení je práce s **časovými řadami** – než MongoDB vyvinulo specializovanou podporu, Bosch musel spravovat časová data manuálně (např. dělit je do kolekcí po částech času). Vyhlížené zavedení Time Series Collections v MongoDB však tento proces zjednoduší a poskytne vyšší výkon při dotazování napříč časem. Z hlediska architektury Bosch ocenil, že MongoDB je **multi-cloud** – jedna z výzev ve velkých korporacích je vyhnout se závislosti na jednom dodavateli cloudu. MongoDB lze provozovat on-premise i v cloudu různých poskytovatelů, což Bosch umožňuje **migraci části dat do Azure** a využití tamních analytických služeb, aniž by musel měnit databázovou platformu. Integrace MongoDB probíhala v úzké spolupráci s MongoDB Inc. – Bosch poskytoval zpětnou vazbu na typické scénáře IoT a MongoDB vylepšuje své produkty tak, aby těmto scénářům vyhověly (např. již zmíněné časové kolekce). Celkově lze říci, že Bosch Digital díky MongoDB **získal jednotnou datovou platformu pro IoT**, na které dokázal vybudovat robustní služby odolné vůči budoucím výzvám. Tato případová studie ukazuje, že i v průmyslovém gigantu s obrovským dědictvím dokázali využít moderní NoSQL databázi k **urychlení inovací a získání konkurenční výhody** v éře průmyslového internetu věcí.

# Výhody a nevýhody

Tato kapitola shrnuje obecné klady a zápory databázového systému MongoDB a hodnotí konkrétní silné a slabé stránky aktuálního řešení.

## Obecné výhody a nevýhody MongoDB

**Výhody:**

* **Flexibilita datového modelu:** Dokumentová struktura (BSON/JSON) umožňuje uložení různorodých dat bez nutnosti pevně definovaného schématu.
* **Horizontální škálování:** Pomocí shardingu lze snadno rozšiřovat kapacitu databáze přidáváním dalších shardů.
* **Vysoká dostupnost:** Replikace a automatický failover udržují databázi funkční i při selhání některého uzlu.
* **Rychlý vývoj a jednoduchý import dat:** Nástroje jako mongoimport umožňují efektivní práci s CSV/JSON soubory.
* **Vestavěný agregační framework:** Pokročilá analýza a transformace dat bez nutnosti externích nástrojů.

**Nevýhody:**

* **Možná nekonzistence při výpadcích:** MongoDB preferuje dostupnost, což může vést k dočasné nekonzistenci dat.
* **Složitější ladění výkonu:** Efektivita závisí na správném výběru shard klíče, indexování a topologie.
* **Citlivost na síťovou infrastrukturu:** Latence mezi uzly v distribuovaném prostředí může ovlivnit výkon systému.

## Výhody a nevýhody našeho řešení

**Výhody:**

* **Škálovatelnost:** Díky shardingu lze systém flexibilně rozšiřovat podle růstu dat.
* **Vysoká dostupnost:** Každý shard je replikován ve třech uzlech, což umožňuje provoz bez přerušení i při výpadku některé části systému.
* **Automatizace nasazení:** Použití docker-compose a připravených skriptů zajišťuje rychlé a bezchybné zprovoznění databázového prostředí.
* **Efektivní import dat:** CSV soubory jsou automaticky načítány při spuštění systému.
* **Flexibilní dotazování:** Dokumentový model podporuje rozšiřování struktury uložených záznamů bez zásahu do schématu.

**Nevýhody:**

* **Složitost infrastruktury:** Nastavení více uzlů, replikace a konfigurace klastru vyžadují zvýšené nároky na správu a prostředky.
* **Závislost na stabilní síti:** Distribuovaná architektura klade vyšší nároky na kvalitu síťového propojení mezi uzly.

Celkově řešení nabízí vysokou míru spolehlivosti a flexibility. Díky shardingu je systém připraven na budoucí rozšíření bez nutnosti zásadních změn v architektuře.

# Další specifika

Implementace MongoDB je provedena v souladu s doporučenými praktikami pro sharding a replikaci. Všechny klíčové komponenty (shardy, konfigurační servery, router) jsou nakonfigurovány standardním způsobem a integrují se s automatizačními skripty.

Systém neobsahuje žádné specifické nebo nestandardní úpravy, což umožňuje jednodušší správu a snadnější údržbu. Nasazení odpovídá doporučenému modelu MongoDB a pokrývá všechny důležité aspekty bez nutnosti alternativních nebo složitějších řešení.

# Data

## Zdroje a popis dat

V rámci projektu pracujeme se třemi různými datovými sadami (datasets), které jsme získali z veřejných zdrojů. Všechny tři datasety jsou ve formátu CSV (comma-separated values). Níže popisujeme každý dataset, jeho původ (zdroj a případný URL), strukturu (počet sloupců, přítomnost ID, datové typy) a obsahové zaměření.

### Amazon – produktová data z Amazonu

První dataset obsahuje informace o produktech z e-shopu Amazon. Data pocházejí z **Kaggle** – konkrétně z datové sady *“Amazon Sales Dataset”* vytvořené uživatelem Karkavelraja J., která byla sesbírána webovým scrapováním oficiálního webu Amazon v lednu 2023([zenodo.org](https://zenodo.org/records/10157504#:~:text=This%20dataset%20is%20having%20the,the%20Official%20Website%20of%20Amazon)). Dataset je dostupný ve formě CSV souboru amazon.csv (cca 4,7 MB)​([kaggle.com](https://www.kaggle.com/datasets/karkavelrajaj/amazon-sales-dataset)). Tento soubor obsahuje **1465 záznamů** (řádků), z nichž každý reprezentuje jeden produkt na Amazonu (včetně jedné ukázkové recenze). Data mají celkem **16 sloupců**​. Mezi hlavní atributy patří: unikátní identifikátor produktu **product\_id** (slouží jako ID), název produktu, kategorie, aktuální (snížená) cena a původní cena, procentuální sleva, průměrné hodnocení produktu (**rating**) a počet hodnocení (**rating\_count**), dále textový popis produktu a informace o jedné uživatelské recenzi (ID a jméno uživatele, ID recenze, titulek a obsah recenze) a také odkazy na obrázek produktu a na produktovou stránku​ ([zenodo.org](https://zenodo.org/records/10157504#:~:text=%2A%20product_id%20,voted%20for%20the%20Amazon%20rating)) . Všechny sloupce byly v originále načteny jako textové (řetězcové) hodnoty. Klíčové numerické údaje jako ceny, procenta slev či hodnocení jsou tedy potřeba převést na číselné datové typy (viz čištění dat níže). Dataset neobsahuje mnoho chybějících hodnot – **většina produktů má všechny údaje vyplněny**; pouze u ~10 záznamů chyběl název či cena (což jsme vyřešili při čištění dat odstraněním těchto neúplných záznamů). Celkově tato datová sada poskytuje komplexní pohled na produkty na Amazonu včetně jejich cen, slev a hodnocení od zákazníků.

### Google Play Store – data o mobilních aplikacích

Druhá datová sada obsahuje informace o mobilních aplikacích z obchodu Google Play. Dataset jsme získali taktéž z **Kaggle**, kde je dostupný jako *“Google Play Store Apps”*​([medium.com](https://medium.com/@vivdto/data-science-is-it-possible-to-predict-the-rating-of-google-play-store-apps-based-on-the-given-1a8dc369d062#:~:text=Data%20Source%20Details)). Soubor googleplaystore.csv má **13 sloupců** a **obsahuje 10 841 záznamů** (řádků), z nichž každý představuje jednu mobilní aplikaci dostupnou na Google Play Store​([datacareer.de](https://www.datacareer.de/blog/exploratory-data-analysis-in-python/#:~:text=A%20guide%20to%20Exploratory%20Data,App%2010841)). Mezi atributy patří název aplikace, **kategorie** (např. GAME, FAMILY, TOOLS,...), **průměrné hodnocení** aplikace (stupnice 1–5), počet uživatelských recenzí, velikost aplikace, počet instalací (downloads), typ aplikace (Free/Paid), cena (v USD pro placené aplikace, jinak 0), věkové hodnocení obsahu (*Content Rating*, např. Everyone, Teen), hlavní žánr/y aplikace, datum poslední aktualizace, aktuální verze a minimální vyžadovaná verze systému Android​([tamoghnasaha-22.medium.com](https://tamoghnasaha-22.medium.com/all-you-really-need-to-know-python-notebook-advanced-pandas-da1697aee647)). Datové typy jsou kombinací řetězců (název, kategorie, atd.) a číselných/textových hodnot, které však často obsahují jednotky nebo symboly (např. počet instalací je uveden jako text "1,000,000+"). Tento dataset nemá vyloženě explicitní ID sloupec – unikátní identifikátor by mohl být odvozen např. z názvu aplikace (případně kombinace názvu a dalších atributů), přičemž při importu do databáze jsme se spolehli na automaticky generované **\_id** v MongoDB. **Chybějící hodnoty**: ve zdrojových datech chybí hodnocení u části aplikací – rating má vyplněno pouze 9367 z 10841 záznamů​ (tj. asi 13,6 % aplikací nemá rating, patrně protože byly nové nebo nedostatečně hodnocené). Několik záznamů postrádalo také údaj o aktuální verzi či Android verzi (řádově jednotky případů). Celkově však většina polí (název, kategorie, počet instalací atd.) je vyplněna u všech záznamů. Tento dataset umožňuje analyzovat popularitu aplikací (počty instalací), jejich hodnocení a kategorizaci.

### Video Game Sales – prodeje videoher

Třetím datasetem jsou historická data o prodejích videoher po celém světě. Data pocházejí opět z **Kaggle**, kde jsou k dispozici jako *“Video Game Sales”* dataset. Jde o často citovanou sadu, která kombinuje údaje z herního průmyslu (pravděpodobně převzaté z VGChartz a dalších zdrojů). Dataset obsahuje **16 598 záznamů** (po odstranění 2 neúplných řádků) a **11 sloupců**​. Každý záznam představuje jednu konkrétní videohru (typicky titul na dané platformě). Atributy zahrnují: **Rank (pořadí)** – pořadí hry podle celosvětových prodejů (unikátní hodnota 1–16598, lze ji použít jako ID záznamu), **Name** – název hry, **Platform** – platforma/konzole (např. Wii, PC, X360, PS4,...), **Year** – rok vydání, **Genre** – žánr hry (Action, Sports, RPG, atd.), **Publisher** – vydavatelská společnost, a dále numerické atributy udávající odhad počtu prodaných kusů (v milionech) v různých regionech: **NA\_Sales** (Severní Amerika), **EU\_Sales** (Evropa), **JP\_Sales** (Japonsko), **Other\_Sales** (zbytek světa) a **Global\_Sales** (celosvětové prodeje, což je součet všech regionů)​. Všechna prodejní čísla jsou číselného datového typu (desetinné číslo, jednotka = milion prodaných kusů). Tento dataset neobsahuje žádné textové recenze ani hodnocení – jedná se čistě o numerická obchodní data. Po předčištění dat (odstranění dvou záznamů s chybějícími údaji) v něm **nezůstávají chybějící hodnoty** v žádném sloupci – pro každou hru máme kompletní informace o roce, žánru, prodejích atd. Tato data umožňují analyzovat historické trendy v prodejích her, oblíbené žánry a úspěšnost vydavatelů na různých trzích.

### Google Play Store – uživatelské recenze

*(dataset* ***googleplaystore\_user\_reviews****)*

Pro účely ukázky práce s vnořenými (embedded) dokumenty v MongoDB byl k hlavnímu datasetu **„Google Play Store Apps“** přidán pomocný dataset **googleplaystore\_user\_reviews**. Obsahuje předem normalizované textové recenze uživatelů k jednotlivým aplikacím (pole *review\_id, app\_name, translated\_review, sentiment\_polarity, sentiment\_subjectivity* aj.). Díky samostatné kolekci lze na recenze cíleně provádět pokročilé operace nad vnořenými strukturami, zejména:

* **agregace** a statistiky sentimentu (**$unwind → $group**, výpočet průměrné polarity, třídění recenzí podle délky textu),
* **filtrování** vnořených polí (**$elemMatch**, podmínky na *sentiment\_polarity* či tagy),
* **transformace** a obohacení dokumentů (**$project**, **$addFields**, vytváření derivovaných metadat, např. délka recenze nebo tematické štítky).

Oddělení recenzí do vlastní kolekce zároveň:

* **zjednodušuje škálování** – objem recenzí může růst nezávisle na tabulce aplikací,
* **zpřehledňuje datový model** – aplikace a jejich hodnocení jsou logicky odděleny,
* **demonstruje flexibilitu MongoDB** – snadné ukládání heterogenních dat bez pevného schématu a rychlé dotazování nad embedded poli.

Tato kolekce se importuje automaticky spolu s ostatními CSV soubory (**import\_datasets.sh**) a při spuštění clusteru je shardována hashovaným **klíčem \_id**, aby se zápis i čtení rovnoměrně rozkládaly mezi uzly.

## Čištění a předzpracování dat

Před nahráním do databáze bylo nutné všechny tři datasety očistit a převést do vhodné struktury pomocí Pythonu (knihovny *Pandas*, *NumPy* apod.). Níže uvádíme hlavní kroky **čištění dat pro každý dataset**:

* **Amazon (produkty):** Ceny byly v databázi Amazonu uváděny s měnou a dalšími symboly – např. indická rupie ₹ před cenou. Nejprve jsme tedy odstranili symbol měny (₹) a také případné oddělovače tisíců či jiné nepovolené znaky z cenových polí (**actual\_price**, **discounted\_price**)​. Následně jsme tyto sloupce převedli z textu na číselný typ (desetinné číslo – ceny v měně). Podobně jsme očistili procenta slev – odstranění znaku **%** a konverze na číselný typ (0–100). U hodnocení produktu (**rating**) a počtu hodnocení (**rating\_count**) jsme provedli převod na float, resp. integer. Dále jsme zkontrolovali chybějící hodnoty – několik málo záznamů nemělo vyplněný název produktu či cenu; tyto neúplné řádky jsme odstranili, jelikož představovaly zanedbatelný zlomek dat. Výsledkem je čistá tabulka, kde každá řádka obsahuje kompletní informace o daném produktu v jednotném formátu (např. číselné sloupce mohou být dále použity pro výpočty).
* **Google Play (aplikace):** Tento dataset vyžadoval odstranit či upravit několik specifických znaků v datech. Ze sloupce *Installs* (počet instalací) jsme odstranili **znaménko “+”** a **čárky** v číslech (např. **"1,000,000+"** -> **1000000**) a převedli hodnotu na integer, abychom získali skutečné číselné počty instalací. Ve sloupci *Price* jsme u placených aplikací odstranili symbol **“$”** (např. **"$4.99"** -> **4.99**) a poté převedli cenu na číselný typ (float); u neplacených aplikací zůstala cena 0. Sloupec *Reviews* (počet recenzí) byl načten jako text, ten jsme převedli na integer. Dále jsme sjednotili označení velikosti aplikace – položky **"Varies with device"** (liší se podle zařízení) jsme nahradili hodnotou *NaN* (neznámá hodnota), protože je nelze kvantifikovat. Při čištění jsme také narazili na **nekonzistentní řádek**, kde chyběla kategorie a posunuly se hodnoty v sloupcích; tento vadný záznam jsme odstranili, aby data byla správně zarovnána. Rovněž jsme eliminovali případné duplicitní záznamy (pokud se tatáž aplikace v datasetu objevila dvakrát). Nakonec jsme zkontrolovali chybějící údaje – sloupec *Rating* má cca 1474 chybějících hodnot (tyto záznamy jsme ponechali a s NaN hodnotami pracovali opatrně při výpočtech průměrů), u 1 záznamu chyběl *Content Rating* a u několika aplikací nebyla uvedena verze (ponecháno jako nevyplněné). Po těchto úpravách zůstalo v datasetu **10840** platných záznamů (po odstranění chybných či duplicitních řádků).
* **Videoherní prodeje:** Tato datová sada byla již v relativně dobré kvalitě. Identifikovali jsme **2 řádky s chybějícími údaji**, pravděpodobně chybějící rok či prodejní čísla – ty jsme odstranili (čímž se počet záznamů snížil z 16 600 na **16 598**. Následně jsme provedli drobné úpravy typů: rok vydání jsme převedli na celočíselný typ (integer) a prodejní čísla ponechali jako desetinná čísla (float). Hodnoty celosvětových prodejů jsme pro kontrolu přepočítali jako součet regionálních prodejů, abychom ověřili jejich konzistenci (souhlasily). V tomto datasetu nebylo potřeba odstraňovat žádné speciální znaky – čísla již byla uvedena přímo (v milionech kusů). Data tak byla připravena k analýze bez dalších zásahů.

## Import dat do MongoDB

Po vyčištění byla data nahrána do databázového systému **MongoDB**. Pro import jsme využili nástroj **mongoimport**, který umožňuje importovat CSV soubory přímo do MongoDB kolekcí. Celý proces byl zautomatizován ve skriptu **import\_datasets.sh**, jenž běžel uvnitř Docker kontejneru nazvaného **init-data**. Skript při startu kontejneru spustil sekvenci příkazů **mongoimport** pro každý dataset.

Všechna data byla uložena do jedné MongoDB databáze s názvem **Ecommerce**, každá datová sada v samostatné kolekci (např. ***amazon***, ***googleplatstore***, ***vgsales***). V MongoDB tak každý záznam z CSV odpovídá jednomu dokumentu ve své kolekci. Primární klíče nebylo nutné ručně vytvářet – MongoDB automaticky přidělilo každému dokumentu jedinečný **\_id**. Import pomocí Docker kontejneru zajistil, že data budou připravena k použití ihned po spuštění aplikace, bez nutnosti manuálního zásahu.

**Proč právě MongoDB?** Zvolili jsme MongoDB zejména kvůli její **flexibilitě** a snadné integraci s Dockerem. MongoDB je dokumentová NoSQL databáze, která nevyžaduje předem pevně definovaná schémata – to se hodí, protože naše tři datasety mají odlišnou strukturu a nelze je jednoduše vtěsnat do jedné relační tabulky. MongoDB umožňuje ukládat každou datovou sadu samostatně, s různými poli, a případně je rozšiřovat o nové atributy bez migrací schématu. Dále, objem dat (několik desítek tisíc záznamů) pro MongoDB nepředstavuje problém a dotazování nad nimi je rychlé. Využití MongoDB nám tedy poskytlo **flexibilní úložiště** schopné pojmout heterogenní data z e-commerce, mobilních aplikací i herního průmyslu na jednom místě.

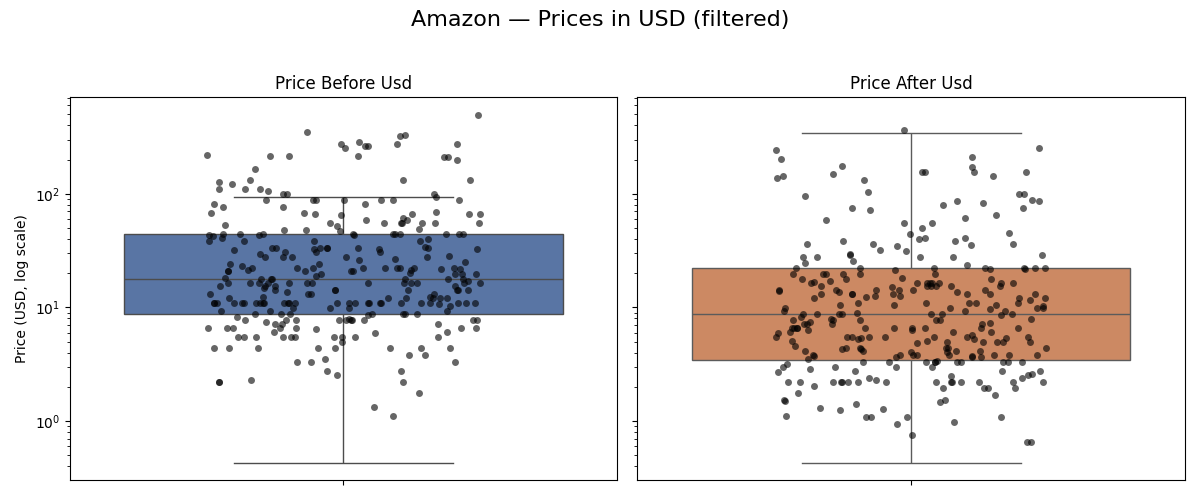
## Průzkumná analýza a základní statistiky

Po importu jsme nad daty provedli základní statistické analýzy a explorační datovou analýzu pomocí knihoven **Pandas** a **NumPy**. Cílem bylo získat přehled o obsahu datasetů – například počty záznamů, rozsahy hodnot, průměry, sumy, četnosti kategorií a podobně – a odhalit případné zajímavé vzory či odchylky. Níže uvádíme shrnutí klíčových poznatků pro každý dataset:

* **Amazon (produkty):** Celkem máme **1465 produktů** napříč ~160 různými kategoriemi (podkategorie Amazonu; některé jsou velmi specifické). Průměrná **aktuální cena** produktu po slevě je zhruba 35 USD, zatímco průměrná **původní cena** byla kolem 60 USD – to naznačuje, že typická sleva je okolo **40–45%**. Potvrdili jsme to i výpočtem průměrné hodnoty sloupce *discount\_percentage*, který vyšel ~42%. Hodnocení produktů jsou na Amazonu obecně vysoká: průměrné **uživatelské hodnocení** je cca **4,4 hvězdičky** z 5 a medián dokonce 4,5. Nejčastěji se v datech vyskytují produkty z kategorií jako elektronika a příslušenství – např. kategorie *Surge Protectors*, *Memory* nebo *Streaming Clients*. Spočítali jsme, že **nejvíce produktů** spadá do kategorie *Accessories* a *Beverages* (obě po ~50 produktech). Dále jsme zjistili, že korelace mezi cenou a hodnocením není výrazná – dražší produkty nemají nutně lepší hodnocení (viz vizualizace níže). Celkový počet hodnocení (**rating\_count**) se mezi produkty velmi liší: některé položky mají jen desítky hodnocení, jiné i několik tisíc (průměrný počet hodnocení je ~550). Souhrnně jsme ověřili, že datový soubor pokrývá širokou škálu produktů s různými cenovými úrovněmi a většina z nich je velmi dobře hodnocena zákazníky.
* **Google Play (aplikace):** Dataset obsahuje informace o **10 840 aplikacích** (po očištění). Tyto aplikace jsou rozděleny do **34 různých kategorií**; zjistili jsme, že **nejpočetnější kategorií** je *Family* (rodinné a dětské aplikace), která zahrnuje kolem **1900 aplikací**, dále *Game* (~1100 aplikací) a *Tools* (~800). Naopak nejméně zastoupené jsou kategorie jako *Dating* či *Events*. Pokud jde o hodnocení, průměrné **rating** aplikace je ~**4,17** (tj. většina aplikací má hodnocení okolo 4 až 4,5 hvězdičky). Přibližně 13,5 % aplikací nemá hodnocení uvedeno (NaN), což jsme zohlednili – při výpočtu průměru hodnocení jsme brali v úvahu pouze hodnocené aplikace. **Počet instalací** se mezi aplikacemi velmi liší – rozpětí je od jednotek (některé nové aplikace mají např. jen 5+ instalací) až po více než 1 miliardu. Vypočítali jsme, že medián počtu instalací spadá do kategorie 10 000+ až 50 000+ (tj. pro typickou aplikaci desítky tisíc stažení). **Drtivá většina aplikací je zdarma** – konkrétně cca 92.5% jsou Free a jen ~7.5% placené (to odpovídá poměru na Google Play). Mezi placenými aplikacemi je průměrná cena relativně nízká, kolem 2–3 USD (většina placených stojí do 5 USD, jen několik málo překračuje 20 USD). Zajímavým zjištěním je, že **hodnocení aplikací nekoreluje silně s počtem instalací** – i malé nové aplikace mohou mít vysoké hodnocení 5.0, zatímco některé masově stahované aplikace mají hodnocení “jen” kolem 4.0 (uživatelé jsou náročnější). Tento dataset nám tak poskytl přehled o ekosystému mobilních aplikací – víme, které kategorie dominují, jaké jsou typické počty stažení a že uživatelské skóre je u většiny aplikací kladné.
* **Videoherní prodeje:** V datasetu je zahrnuto **16 598 her** vydaných v letech 1980–2020. Průměrná hra z našeho vzorku prodala **~0,53 milionu kusů** (tj. ~530 tisíc kopií), přičemž medián je ještě nižší (~0,17 mil.) – to ukazuje, že distribuce prodejů je velmi **skewed** (většina her má relativně nízké prodeje, ale existuje pár obrovských hitů). Nejprodávanější hra v datech je *Wii Sports* (Nintendo, 2006) s **82,74 miliony** prodaných kusů globálně – výrazný outlier. Druhá v pořadí je *Grand Theft Auto V* (~55 milionů) a dále *Super Mario Bros.*, *Tetris* apod. Celkový součet globálních prodejů všech her v datasetu je zhruba **9 miliard kusů**. Z hlediska **žánrů** jsou nejpočetněji zastoupeny hry žánru *Action* (akční, cca 3316 her) a *Sports* (2346 her​, následují Shooter, Role-Playing, atd. – což naznačuje, jaké typy her vycházely nejčastěji. Co se týče prodejů podle regionu, **Severní Amerika** tvoří největší podíl na globálních prodejích (zhruba 50 % všech prodejů pochází z NA), druhým největším trhem je Evropa (~30 %), následuje Japonsko (~15 %) a zbytek světa (~5 %). Tento poměr jsme získali sečtením prodejů ve sloupcích NA\_Sales, EU\_Sales, JP\_Sales a Other\_Sales – například souhrn pro NA region je kolem 4320 milionů kusů, zatímco globální souhrn srovnatelných dat ~8926 mil. Naznačuje to, že severoamerický trh byl pro herní průmysl nejzásadnější. Dále jsme zkoumali **vydavatele** – dataset obsahuje přes 500 různých publisherů, ale prodeje jsou koncentrovány do několika velkých firem. Spočítali jsme, že **nejúspěšnějším vydavatelem** (součtem globálních prodejů) je **Nintendo** (více než 1,5 miliardy prodaných kopií her, které vydalo), následuje **Electronic Arts** (~1,1 mld) a potom Activision, Sony, Ubisoft atd. Deset největších vydavatelů pokrývá podstatnou část celkových prodejů (viz graf níže). Tyto statistiky potvrzují očekávané trendy – dominanci Nintenda a EA v herním průmyslu a silnou roli akčních a sportovních her.

## Vizualizace dat a zjištění

Pro lepší pochopení dat jsme vytvořili řadu grafů: krabicové grafy (*boxploty*) pro rozdělení hodnot, histogramy četností, bodové grafy pro zkoumání závislostí, sloupcové grafy pro porovnání kategorií a také výsečové grafy (*pie charts*) pro podíly. Níže uvádíme některé příklady těchto vizualizací spolu s interpretací:

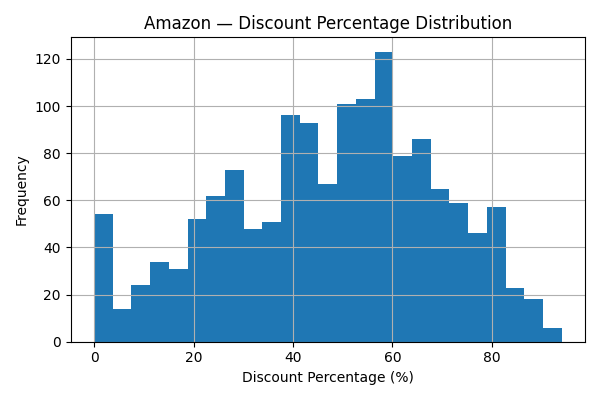


**Obr. 1: Rozložení cen produktů na Amazonu.**

Graf zobrazuje vedle sebe boxploty původní ceny (price\_before\_usd, vlevo) a ceny po slevě (price\_after\_usd, vpravo) na logaritmické ose v USD.

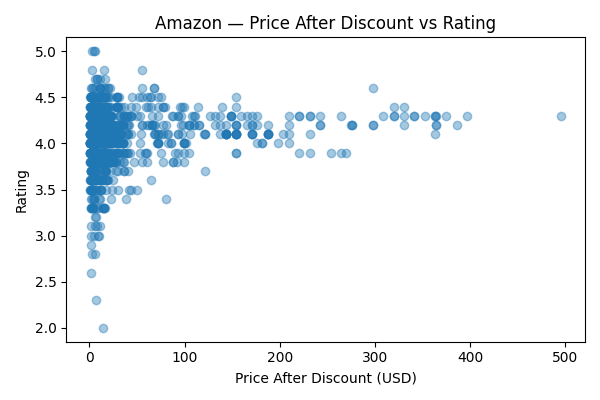
* **Medián** původní ceny je přibližně **18 USD**, medián zlevněné ceny kolem **9 USD**, což jasně ukazuje posun dolů.
* **IQR** původních cen se pohybuje zhruba od **9 USD do 47 USD**, zatímco u zlevněných cen od **4 USD do 22 USD**, tedy po slevě je cenové rozmezí o něco užší.
* **Outliery** (tečky) dosahují cen ve stovkách i tisících dolarů (např. prémiové položky kolem 500–1 000 USD); logaritmická osa zároveň umožňuje vidět i nejlevnější produkty v řádu jednotek dolarů.

Celkově tento graf potvrzuje, že dataset zahrnuje široké spektrum cen od levných položek pod 5 USD až po luxusní zboží rozsahem cen, a že slevy výrazně snižují střední hodnotu cenového rozdělení.

****

**Obr. 2: Histogram** **procentuálních slev**

Kromě boxplotu jsme pro Amazon data vykreslili také **histogram** procentuálních slev, který ukázal dvě nejčastější skupiny produktů – jedny se slevou minimální (0–5%) a jiné s výraznými slevami okolo 40–60% (většina položek spadá právě do rozmezí 40–60% slevy, což koresponduje s průměrem ~42%).

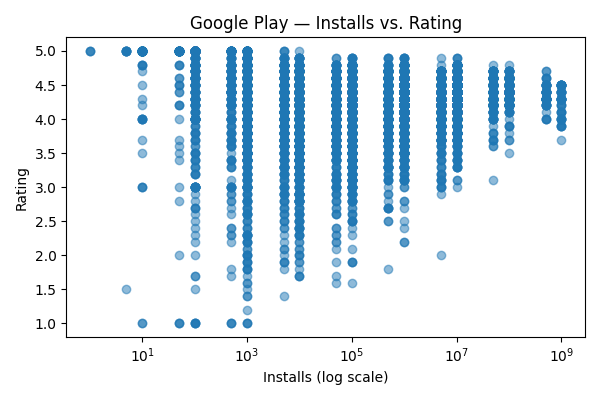
****

**Obr. 3: Bodový graf** **závislosti ceny a hodnocení**

Na tomto scatter plotu („Amazon — Price After Discount vs Rating“) vidíme rozptýlení bodů podle ceny po slevě (v USD) a uživatelského hodnocení (1–5 hvězd). Graf ukazuje:

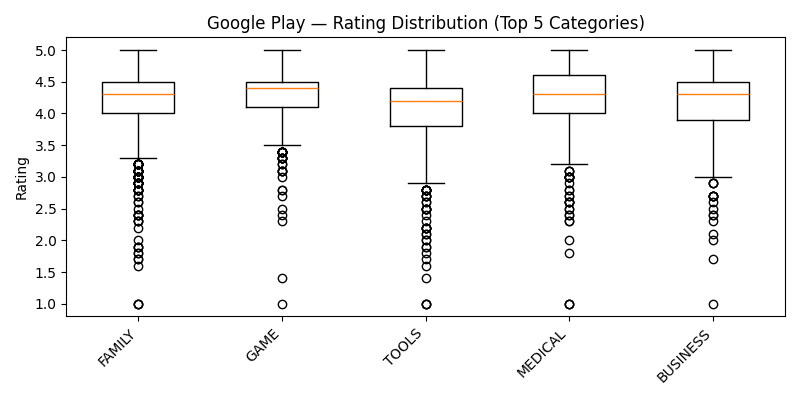
* Produkty s velmi nízkou cenou (pod 10 USD) i produkty s cenou v řádu stovek USD dosahují podobně vysokých hodnocení (cca 4–5 hvězd).
* Body jsou rozptýlené napříč celou šířkou osy cen, bez zjevného trendu vzestupu či poklesu hodnocení s rostoucí cenou.
* To potvrzuje nízkou korelaci mezi cenou po slevě a ratingem — cenová hladina nemá výrazný vliv na průměrné hodnocení.

Celkově scatter plot doplňuje boxplot, ukazuje, že ačkoliv slevy významně snižují cenu, kvalita (ve smyslu hodnocení) je vysoce konzistentní napříč celou škálou cen.



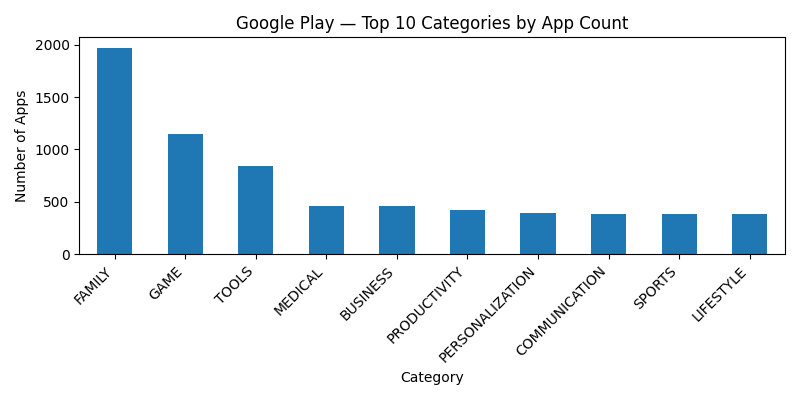
**Obr. 4: Vztah počtu instalací a hodnocení u aplikací (Google Play).**

Tento bodový graf znázorňuje každou aplikaci jako bod v souřadnicích: horizontálně logaritmus počtu instalací (10^1 = 10 instalací, 10^9 = 1 miliarda) a vertikálně její průměrné uživatelské hodnocení. Graf ukazuje zajímavý trend: valná většina aplikací má hodnocení mezi 4.0 a 5.0 (vysoké skóre) bez ohledu na počet stažení – body jsou nahoře napříč celou osou X. I aplikace s velmi malým počtem instalací (10–100) často dosahují hodnocení 4–5. Naopak aplikací s hodnocením pod 3.0 je málo a většinou mají také nižší počet instalací (levý spodní roh grafu). Aplikace s **extrémně vysokým počtem stažení** (pravý okraj, kolem $10^8$–$10^9$) mají hodnocení převážně v rozmezí 4.1–4.6 – zdá se tedy, že mega-populární aplikace si udržují solidní, ale ne nutně nejvyšší možná hodnocení. Tento graf ilustruje, že **počet instalací nepřímo souvisí s hodnocením**: špičkově hodnocené jsou i niche aplikace s málo uživateli, zatímco globální hity mohou mít hodnocení lehce nad 4.0 (což je stále velmi dobré).



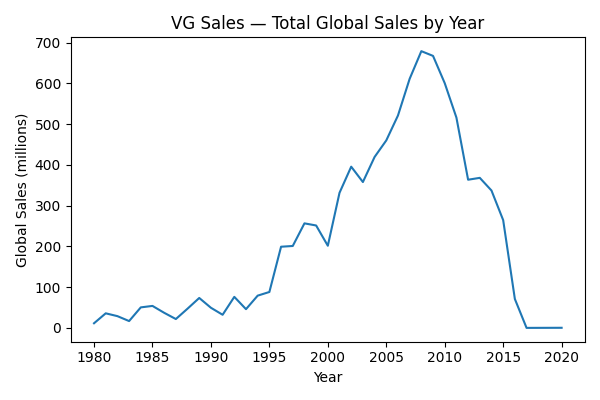
**Obr. 5: Boxplot hodnocení v top 5 kategoriích**

Zajímavý byl také **krabicový graf hodnocení v top 5 kategoriích** (Family, Game, Tools, Medical, Business): ukázal, že mediány hodnocení se mezi kategoriemi liší jen mírně – např. kategorie *Family* a *Game* měly medián ~4.3, zatímco *Business* lehce nižší (~4.0). To naznačuje, že uživatelské spokojenosti jsou vysoké napříč různými typy aplikací, jen u pracovních a nástrojových app může být hodnocení o něco přísnější.



**Obr. 6: Počet aplikací v jednotlivých kategoriích**

Sloupcový graf **počtu aplikací v jednotlivých kategoriích** pak zdůraznil dominanci kategorie *Family* (nejvyšší sloupec, ~18% všech aplikací) a *Game*, oproti minoritním kategoriím.



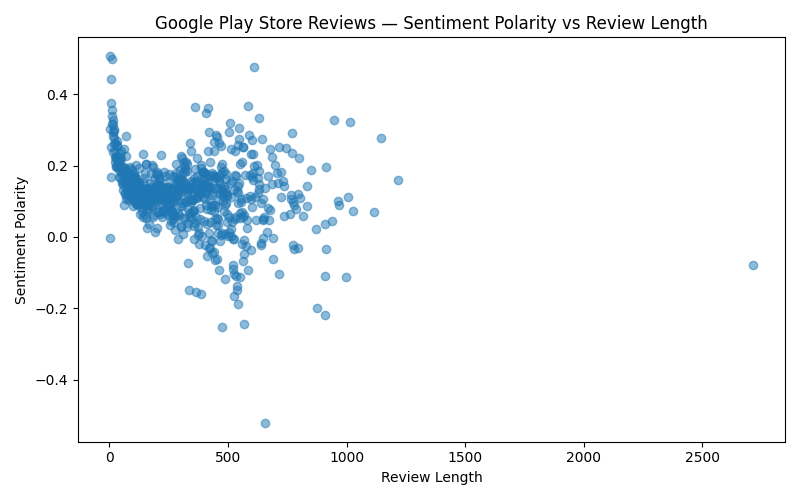
**Obr. 7: Historický vývoj globálních prodejů videoher podle roku vydání.**

Čára znázorňuje součet ročních globálních prodejů všech her vydaných v daném roce (v milionech kusů). Je vidět výrazný trend: prodeje videoher **pozvolna rostly od 80. let**, na přelomu 80/90. let se pohybovaly kolem 50–100 milionů ročně, v polovině 90. let začaly strměji stoupat a okolo roku 2008–2009 dosáhly vrcholu – **v roce 2009 celkové globální prodeje her činily téměř 680 milionů kusů**. Toto období odpovídá éře velmi úspěšných konzolí (Wii, Xbox 360, PS3) a hitů jako Wii Sports. Poté vidíme **pokles** – po roce 2010 prodeje klesají, v roce 2015 byly kolem 275 mil. a v roce 2017 dokonce jen ~10 mil. (dataset již po roce 2016 obsahuje minimum dat, protože pravděpodobně nejsou zahrnuty digitální distribuce na nové generace konzolí). Tento graf ilustruje nástup a ústup jednotlivých generací herního průmyslu.



**Obr. 8: Top 10 vydavatelů dle globálních prodejů**

Kromě tohoto časového grafu jsme připravili také **sloupcové grafy pro srovnání prodejů podle kategorií** – například graf **Top 10 vydavatelů dle globálních prodejů** ukázal, že Nintendo dominuje (nejvyšší sloupec ~1700 mil.), následované EA (~1000 mil.) a dalšími (viz výše). Vizualizace pro dataset prodejů her nám pomohly lépe pochopit historické **trendy** (nárůst a pokles prodejů), **profil nejúspěšnějších žánrů a firem** i **geografické rozdíly** v oblibě videoher.



**Obr. 9: Vztah délky recenze a sentimentu (Google Play Store)**

Bodový graf ukazuje, že nejvíce recenzí do **≈ 600 znaků** mívá lehce kladný sentiment (0 – 0,25). S delším textem se hodnocení rozptyluje a častěji klesá k **nule** či mírně do **záporu**, takže obsáhlejší komentáře bývají kritičtější. Extrémně **negativní** či **pozitivní** postoje se vyskytují jen **výjimečně**.

**Shrnutí:** Datová část projektu zahrnovala práci se třemi rozsáhlými a různorodými datasety. U každého z nich bylo nutné provést důkladné předzpracování – od čištění a konverzí dat přes uložení do vhodné databáze (MongoDB) až po výpočet souhrnných statistik. Následná explorační analýza a vizualizace odhalily zajímavé informace: na Amazonu výrazné slevy a vysoká hodnocení produktů, na Google Play dominanci určitých kategorií a obecně pozitivní uživatelské skóre, a v herním průmyslu vrchol prodejů kolem roku 2009 a převahu několika málo velkých hráčů na trhu. Tato zjištění tvoří základ pro další, hlubší analytické úlohy v rámci semestrálního projektu.

# Dotazy

V této kategorii najdete 30 netriviálních příkladů MongoDB dotazů rozdělených do pěti hlavních oblastí: Práce s daty (CRUD), Agregační funkce, Vyhledávání a indexování, Cluster a shardování a Schéma a validace.

**Kategorie 1 – Práce s daty (CRUD)**

Dotaz 6:

db.vgsales.bulkWrite([

{

updateMany: {

filter: { Platform: "PS4" },

update: { $set: { LastUpdated: new Date() } }

}

},

{

deleteMany: { filter: { Year: { $lt: 2000 } } }

},

{

insertOne: {

document: {

Rank: 1,

Name: "New Indie",

Platform: "PC",

Year: 2025,

Genre: "Indie",

Publisher: "DevLab",

NA\_Sales: 1.2,

EU\_Sales: 0.7,

JP\_Sales: 0.4,

Other\_Sales: 0.2,

Global\_Sales: 2.5,

LastUpdated: new Date()

}

}

}

]);

Tento dotaz demonstruje efektivní provádění více CRUD operací v rámci jednoho **bulkWrite** volání. Nejprve aktualizuje všechna PS4 data s aktuálním timestampem, poté odstraní staré záznamy před rokem 2000 a nakonec přidá zcela nový dokument.

**Kategorie 2 – Agregační funkce**

Dotaz 12:

db.vgsales.aggregate([

{ $match: { Year: { $ne: null } } },

{ $group: {

\_id: { $subtract: [ "$Year", { $mod: [ "$Year", 10 ] } ] },

totalSales: { $sum: "$Global\_Sales" }

} },

{ $project: {

decade: { $concat: [ { $toString: "$\_id" }, "s" ] },

totalSales: 1,

\_id: 0

} },

{ $sort: { decade: 1 } }

]);

Tento dotaz využívá kombinaci operátorů **$subtract** a **$mod** k výpočtu desetiletí pro každý rok a poté pomocí **$group** a **$sum** spočítá součet globálních prodejů za každé desetiletí.

**Kategorie 3 – Vyhledávání a indexování**

Dotaz 15:

db.amazon.createIndex(

{ category: 1, discount\_percentage: -1, price\_after\_discount: -1 },

{ name: "CategoryDiscountPriceIdx" }

);

db.amazon.find(

{

category: /^Home&Kitchen\|/,

discount\_percentage: { $gte: 10 }

},

{

\_id: 0,

product\_name: 1,

category: 1,

price\_before\_discount: 1,

price\_after\_discount: 1,

discount\_percentage: 1,

realDiscountValue: {

$subtract: ["$price\_before\_discount", "$price\_after\_discount"]

}

}

)

.sort({ discount\_percentage: -1, price\_after\_discount: 1 })

.hint("CategoryDiscountPriceIdx")

.limit(10);

Dotaz vybírá produkty z kategorie „Home&Kitchen“ s alespoň 10% slevou, spočítá skutečnou hodnotu slevy (realDiscountValue) a seřadí je podle velikosti slevy a ceny. .hint() zajistí efektivní použití složeného indexu.

**Kategorie 4 – Cluster a shardování**

Dotaz 24:

sh.addShardTag("rs-shard-01", "A-M");

sh.addShardTag("rs-shard-02", "N-S");

sh.addShardTag("rs-shard-03", "T-Z");

sh.addTagRange(

"Ecommerce.vgsales",

{ Name: MinKey(), Platform: MinKey() },

{ Name: "M", Platform: MaxKey() },

"A-M"

);

sh.addTagRange(

"Ecommerce.vgsales",

{ Name: "N", Platform: MinKey() },

{ Name: "S", Platform: MaxKey() },

"N-S"

);

sh.addTagRange(

"Ecommerce.vgsales",

{ Name: "T", Platform: MinKey() },

{ Name: MaxKey(), Platform: MaxKey() },

"T-Z"

);

db.getSiblingDB("config").tags.find({}).forEach(printjson);

Zone sharding umožňuje směrovat dotazy o datech na konkrétní shardy podle definovaných rozsahů: názvy her začínající A–M jdou na **rs-shard-01**, N–S na **rs-shard-02** a T–Z na **rs-shard-03**.

**Kategorie 5 – Nested (embedded) dokumenty**

Dotaz 26:

db.apps\_meta.aggregate([

{

$match: {

reviews: { $not: { $elemMatch: { Sentiment\_Polarity: { $lt: 0 } } } },

$expr: { $gt: [{ $size: "$reviews" }, 5] }

}

},

{

$project: {

\_id: 0,

appName: 1,

totalReviews: { $size: "$reviews" },

sentimentPolarities: {

$map: {

input: "$reviews",

as: "r",

in: "$$r.Sentiment\_Polarity"

}

}

}

}

]);

Dotaz vybírá aplikace, které mají více než 5 recenzí a žádná z nich není negativní (**Sentiment\_Polarity** < 0). Výstupem je název aplikace, celkový počet recenzí a seznam všech hodnot sentimentu (**Sentiment\_Polarity**) z vnořených dokumentů v poli reviews.

**Poznámka: Vzhledem k rozsáhlosti a komplexnosti všech 30 příkladů, zahrnujících různé scénáře CRUD, agregací, indexování, shardování i nested dokumenty, naleznete kompletní sadu zbývajících dotazů v souboru dotazy.txt.**

Závěr

V této semestrální práci byl úspěšně navržen a nasazen distribuovaný databázový klastr na bázi MongoDB 6.0.2, využívající sharding, replikaci a kontejnerizaci pomocí Dockeru. Implementace pokryla čtyři hlavní kroky: definici architektury, automatizované zprovoznění celého prostředí, import a čištění heterogenních datových sad (Amazon, Google Play, Video Game Sales) a základní průzkumnou analýzu s vizualizacemi.

Projekt prokázal vysokou flexibilitu dokumentového modelu, jednoduchou škálovatelnost a spolehlivou dostupnost dat. Zároveň by bylo vhodné doplnit hloubkové výkonové testy a srovnání s relačními systémy, rozšířit část s monitoringem a vyhodnocením reálné zátěže. Přesto řešení poskytuje solidní základ pro další rozvoj—možnosti rozšíření o detailní benchmarking, pokročilou bezpečnost či integraci s cloudovými službami otevírají cestu k profesionálnímu nasazení v reálném provozu.

Zdroje

* **Amazon Sales Dataset**  
  Karkavelraja J., *Amazon Sales Dataset*, Kaggle. Dataset produktů z e‑shopu Amazon (1465 záznamů, 16 sloupců).  
  <https://www.kaggle.com/datasets/karkavelrajaj/amazon-sales-dataset>
* **Zenodo – Amazon Data**  
  “This dataset is having the data scraped from the Official Website of Amazon”, Zenodo. Dodatečný popis původu a struktury Amazon dat.  
  <https://zenodo.org/records/10157504#:~:text=This%20dataset%20is%20having%20the,the%20Official%20Website%20of%20Amazon>
* **SAS Communities Library**  
  *Amazon Sales Dataset – EDA of Products Purchased at Original Price*, SAS Communities Library. Metodika čištění cenových polí a procentuálních slev.  
  <https://communities.sas.com/t5/SAS-Communities-Library/Amazon-Sales-Dataset-EDA-of-Products-Purchased-at-Original-Price/ta-p/949725#:~:text=We%20also%20remove%20the%20character,data%20amazon_data_clean_numeric%3B%20set%20work>
* **Google Play Store Apps**  
  *Google Play Store Apps*, Kaggle. Dataset mobilních aplikací (10841 záznamů, 13 sloupců).  
  <https://www.kaggle.com/datasets/lava18/google-play-store-apps>
* **Medium – vivdto**  
  Vividto, *Data Science: Is it possible to predict the rating of Google Play Store Apps based on the given features?*, Medium. Detailní popis zdroje a jeho atributů.  
  <https://medium.com/@vivdto/data-science-is-it-possible-to-predict-the-rating-of-google-play-store-apps-based-on-the-given-1a8dc369d062#:~:text=Data%20Source%20Details>
* **DataCareer.de**  
  *Exploratory Data Analysis in Python*, DataCareer blog. Průvodce EDA postupu nad Google Play datasetem.  
  <https://www.datacareer.de/blog/exploratory-data-analysis-in-python/#:~:text=A%20guide%20to%20Exploratory%20Data,App%2010841>
* **Medium – Tamogh Nasa**  
  Tamogh Nasa, *All You Really Need to Know: Python Notebook Advanced Pandas*, Medium. Návody na čištění a transformaci dat pomocí Pandas.  
  <https://tamoghnasaha-22.medium.com/all-you-really-need-to-know-python-notebook-advanced-pandas-da1697aee647>
* **Medium – CloudCraftz**  
  CloudCraftz, *Google Play Store Ratings & Reviews Analysis*, Medium. Další příklad EDA a vizualizace hodnocení aplikací.  
  <https://medium.com/cloudcraftz/googleplaystore-ratings-reviews-20d2943a11ee#:~:text=Google%20Play%20Store%20Data%20Analysis,object%202%20Rating%209367>
* **Video Game Sales**  
  *Video Game Sales*, Kaggle. Historická data o prodejích videoher (16 598 záznamů, 11 sloupců).  
  <https://www.kaggle.com/datasets/gregorut/videogamesales>

Přílohy

sem| folder\_structure.txt

| semestralni\_prace\_bsqbd.docx

|

+---Data

| | amazon.csv

| | analyse\_data.py

| | googleplaystore.csv

| | googleplaystore\_user\_reviews.csv

| | vgsales.csv

| |

| +---plots

| | myplot.png

| | plot\_2025-04-17 17-00-48\_10.png

| | plot\_2025-04-17 17-00-48\_11.png

| | plot\_2025-04-17 17-00-48\_4.png

| | plot\_2025-04-17 17-00-48\_5.png

| | plot\_2025-04-17 17-00-48\_6.png

| | plot\_2025-04-17 17-00-48\_7.png

| | plot\_2025-04-17 17-00-48\_8.png

| | plot\_2025-04-17 17-00-48\_9.png

| | plot\_2025-04-17 19-31-03\_0.png

| | plot\_2025-04-17 19-31-03\_1.png

| | plot\_2025-04-17 19-31-03\_2.png

| | plot\_2025-04-17 19-31-03\_3.png

| |

| \---raw

| amazon.csv

| googleplaystore.csv

| googleplaystore\_user\_reviews.csv

| vgsales.csv

|

+---Dotazy

| dotazyMongoDB.txt

|

\---Funkcni reseni

| .env

| docker-compose.yml

|

+---keyfile

| Dockerfile

| mongodb-keyfile

|

\---scripts

auth.js||

import\_datasets.sh

init-configserver.js

init-router.js

init-shard01.js

init-shard02.js

init-shard03.js

init\_cluster.sh

mongosh.js

wait-for-it.sh

**1. Složka Data**

| **Soubor / podsložka** | **Stručná charakteristika** |
| --- | --- |
| amazon.csv, googleplaystore.csv, googleplaystore\_user\_reviews.csv, vgsales.csv | Očištěné CSV soubory (uložené v kořenovém adresáři) obsahující data po zpracování v analyse\_data.py. |
| analyse\_data.py | Python skript, který načítá, čistí a provádí první analytické kroky nad těmito datasety. |
| plots/ | |  | | --- | |  |  |  | | --- | | Automaticky generované grafy ve formátu PNG (plot\_\*.png, myplot.png apod.). | |
| raw/ | Původní, neupravené CSV soubory; |

**2. Složka Datazy**

| **Soubor** | **Stručná charakteristika** |
| --- | --- |
| dotazyMongoDB.txt | Jeden soubor obsahující všechny požadované dotazy. U každého dotazu je nejprve uvedeno zadání v přirozeném jazyce (stručné vysvětlení), následované ekvivalentním řešením v MongoDB Query Language (MQL). |

**3. Složka Funkční\_reseni**

| **Soubor / podsložka** | **Stručná charakteristika** |
| --- | --- |
| docker-compose.yml | Konfigurace kontejnerů pro spuštění shardovaného MongoDB clusteru a podpůrných služeb. |
| .env | Konfigurační proměnné (hosty, porty, přístupová hesla). Soubor je přiložen pouze pro potřeby hodnocení. |
| keyfile/ → mongodb-keyfile | Tajný klíč pro interní autentizaci nodů v clusteru. |
| scripts/ | Bash & JavaScript skripty nutné k inicializaci clusteru, vytvoření uživatelů, importu datasetů a ověření funkčnosti: |