Návrh systému – Blog

1. Popis aplikace a motivace

Popis aplikace

Blogová aplikace je webový systém umožňující uživatelům:

- Prohlížet a vyhledávat články na základě jejich titulů, obsahu a tagů (fulltextové vyhledávání).
- Registrovat se a přihlašovat.
- Publikovat články (autorské příspěvky), psát komentáře k článkům a lajkovat obsah.
- Dostávat real-time notifikace o změnách – například když někdo přidá komentář či lajkne článek, a tak je okamžitě informován, pokud aktuálně prohlíží daný článek.

Motivace

Motivací tohoto projektu je demonstrovat moderní, cloud-native architekturu využívající:

- **ASP.NET Blazor WebAssembly** pro bohaté uživatelské rozhraní.
- **ASP.NET Blazor Server** pro implementaci obchodní logiky a poskytování gRPC-Web API.
- **Elasticsearch** pro rychlé a flexibilní fulltextové vyhledávání.
- **SQL databázi (Azure SQL)** jako zdroj pravdy pro transakční data.
- **RabbitMQ** jako messaging systém pro asynchronní synchronizaci mezi hlavním úložištěm (SQL) a vyhledávacím indexem (Elasticsearch).
- **Redis Cache** pro zrychlení čtení dat, zejména tam, kde se informace nemění často.
- **SignalR** pro real-time notifikace, které informují uživatele o interakcích s články (komentáře, lajky).
- **Azure** jako cílovou cloudovou platformu, s využitím load balanceru, spravovaných databází a kontejnerizovaného nasazení.

Tento systém reflektuje moderní přístupy ke škálovatelným webovým aplikacím, kde se odděluje zápisová a čtecí logika, umožňuje efektivní fulltextové vyhledávání a zároveň nabízí dynamickou interakci mezi uživateli.

2. Funkční požadavky

1. **Publikování a správa článků:**
    - Uživatel (autor) může vytvářet, upravovat a odstraňovat své články.
    - Každý článek obsahuje titulek, obsah, seznam tagů, datum publikace a případně další metadata.
    - Články jsou ukládány do relační databáze (Azure SQL) a indexovány do Elasticsearch pro fulltextové vyhledávání.
2. **Vyhledávání a filtrování článků:**
    - Uživatelé mohou vyhledávat články zadáním vyhledávacích výrazů.
    - Vyhledávání probíhá přes Elasticsearch, kde se hledá v titulcích, obsahu a tagách.
    - Možnost filtrování podle kategorií, dat publikace nebo dalších kritérií.
3. **Uživatelská registrace a autentizace:**
    - Uživatel se může registrovat a následně přihlašovat.
    - Přihlášený uživatel získá možnost publikovat články, komentovat a lajkovat.
    - Autentizační mechanismy mohou využívat .NET Identity, OAuth2 nebo JWT.
4. **Interakce s články:**
    - Přihlášení uživatelé mohou psát komentáře k článkům.
    - Uživatelé mohou lajkovat články.
    - Každá interakce (komentář, lajkování) je uložena v databázi a případně změny jsou propagovány dále.
5. **Real-time notifikace a aktualizace:**
    - Server push notifikace využívající SignalR informují uživatele o změnách v článku, např. když někdo přidá komentář či lajkne post.
    - Uživatelé, kteří mají aktuálně otevřený článek, dostanou okamžitou aktualizaci, která změní zobrazení komentářů či počtu lajků.
6. **Synchronizace vyhledávacího indexu:**
    - Po každé změně článku (vytvoření, editace, smazání) se prostřednictvím RabbitMQ odešle událost.
    - Background worker na základě této události aktualizuje index v Elasticsearch.

3. Nefunkční požadavky

1. **Škálovatelnost a dostupnost:**
    - Systém musí být horizontálně škálovatelný.
    - Nasazení na Azure s load balancerem zajistí vysokou dostupnost.
    - Mikroservisní přístup umožňuje nezávislé škálování jednotlivých částí (frontend, backend, messaging).
2. **Výkon:**
    - Rychlá odezva díky použití Redis Cache pro nečastěně měněná data (např. obsah článků, metadata).
    - Elasticsearch zajišťuje okamžité fulltextové vyhledávání.
3. **Bezpečnost:**
    - Veškerá komunikace probíhá přes HTTPS.
    - Implementovaná autentizace a autorizace – přístup ke specifickým funkcím mají jen přihlášení uživatelé.
    - Ochrana proti útokům (např. ochrana proti SQL injection, XSS atd.).
4. **Monitoring a Logging:**
    - Integrace s Azure Monitor a Application Insights pro sledování výkonnosti a chyb.
    - Logování událostí (zejména operace související s messagingem a aktualizací indexu).
5. **Konzistence a synchronizace:**
    - Synchronizace mezi SQL databází (zdroj pravdy) a Elasticsearch probíhá asynchronně pomocí RabbitMQ.
    - Při zápisu do databáze musí dojít ke správné invalidaci v Redis Cache.

4. Seznam uživatelů

1. **Neregistrovaný uživatel (návštěvník):**
    - Může prohlížet články a vyhledávat obsah.
    - Nemá možnost komentovat ani lajkovat.
2. **Registrovaný uživatel:**
    - Může se přihlásit, prohlížet obsah, psát komentáře a lajkovat články.
    - Může také číst v reálném čase aktualizace, pokud má otevřený článek.
3. **Autor/Redaktor:**
    - Kromě základních funkcí registrovaného uživatele může také vytvářet, upravovat a spravovat články.
    - Má přístup k redakčním nástrojům a ovládání své publikace.
4. **Administrátor:**
    - Má nejvyšší práva – spravuje uživatele, moderuje obsah (odstraňuje nevhodné komentáře nebo články), a monitoruje systém.

5. Případy užití

Níže jsou uvedeny klíčové scénáře interakce v systému:

5.1. Registrace a přihlášení

- **Aktér:** Nový nebo existující uživatel
- **Popis:** Uživatel se registruje do systému, následně se přihlašuje a získá přístup k funkcím publikování, komentování a lajkování.

5.2. Publikování článku

- **Aktér:** Autor/Redaktor
- **Popis:** Po přihlášení autor vytvoří nový článek zadáním titulku, obsahu a tagů. Článek se uloží do SQL databáze. Událost o změně se odešle do RabbitMQ, kde background worker aktualizuje index v Elasticsearch.

5.3. Vyhledávání článků

- **Aktér:** Neregistrovaný i registrovaný uživatel
- **Popis:** Uživatel zadá vyhledávací dotaz a systém pomocí Elasticsearch prohledá index a vrátí relevantní výsledky.

5.4. Čtení článku a zobrazení komentářů

- **Aktér:** Uživatel
- **Popis:** Uživatel klikne na článek a systém načte detail článku (používá se Redis Cache pro rychlý přístup) a zobrazí komentáře uložené v SQL.

5.5. Přidání komentáře a lajku

- **Aktér:** Registrovaný uživatel
- **Popis:** Uživatel přidá komentář nebo lajku k článku. Tato akce se zapíše do SQL a server následně prostřednictvím SignalR pushne notifikaci všem uživatelům, kteří právě daný článek sledují.

5.6. Real-time aktualizace obsahu

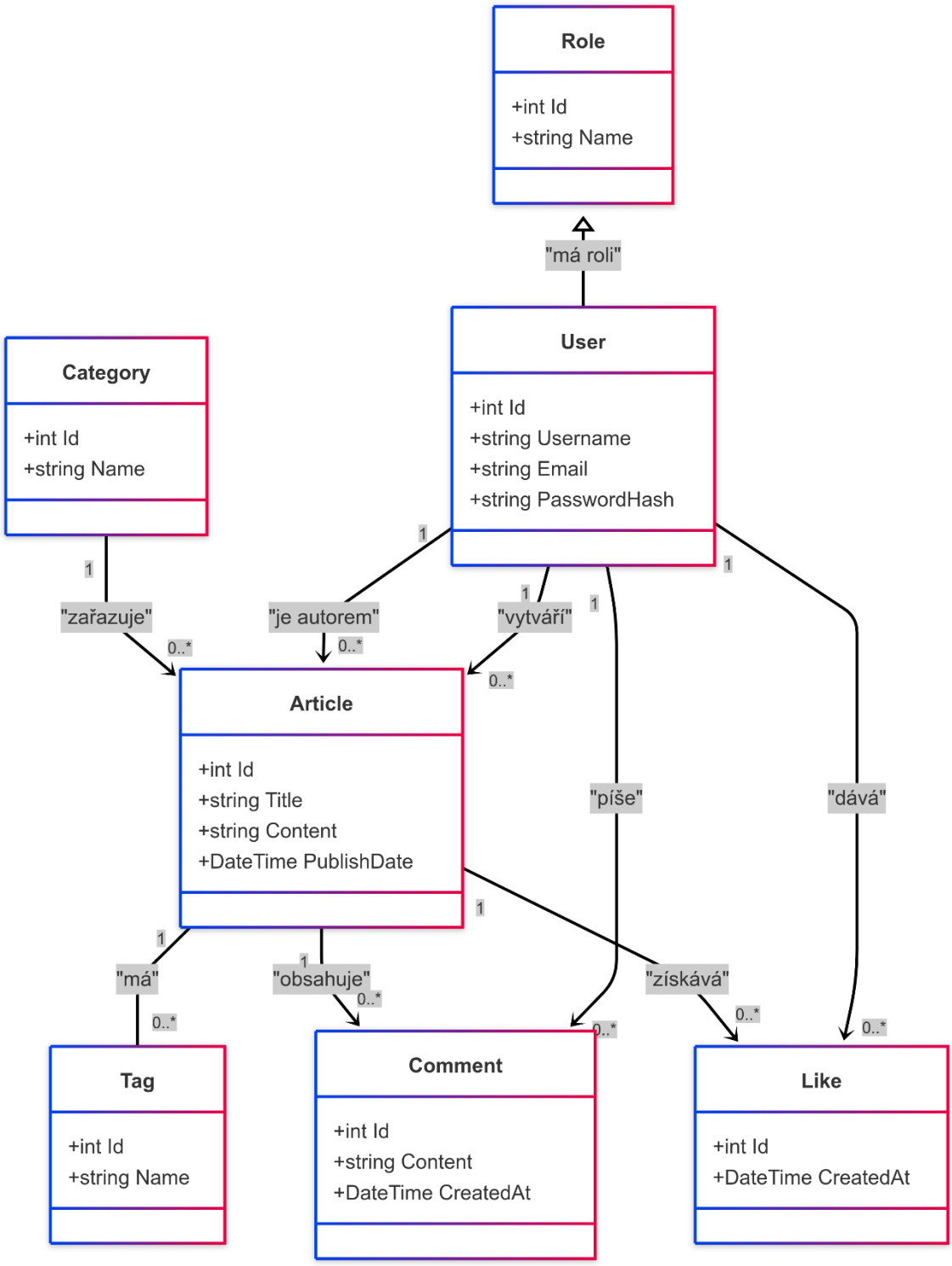
- **Aktér:** Uživatel, který má článek otevřený
- **Popis:** Při změně stavu (nový komentář, změna lajku) server aktivně pushne zprávu pomocí SignalR, klient následně provede obnovení relevantní části UI.

5.7. Synchronizace indexu

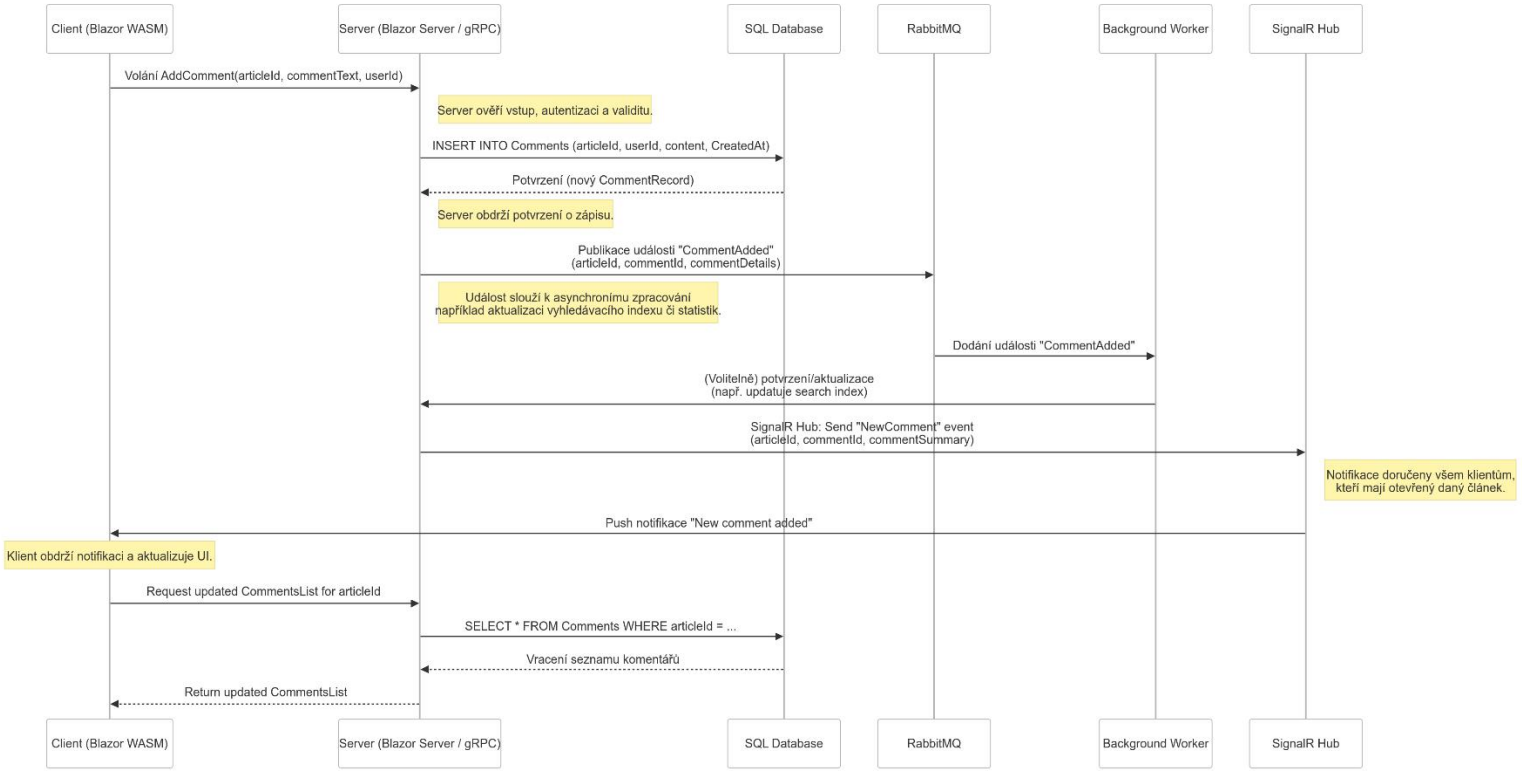
- **Aktér:** Background worker (systém)
- **Popis:** Při každé změně článku (vytvoření, úprava, smazání) se publikuje zpráva do RabbitMQ, kterou odbere worker, ten následně provede odpovídající operace na Elasticsearch indexu.

6. UML Diagramy

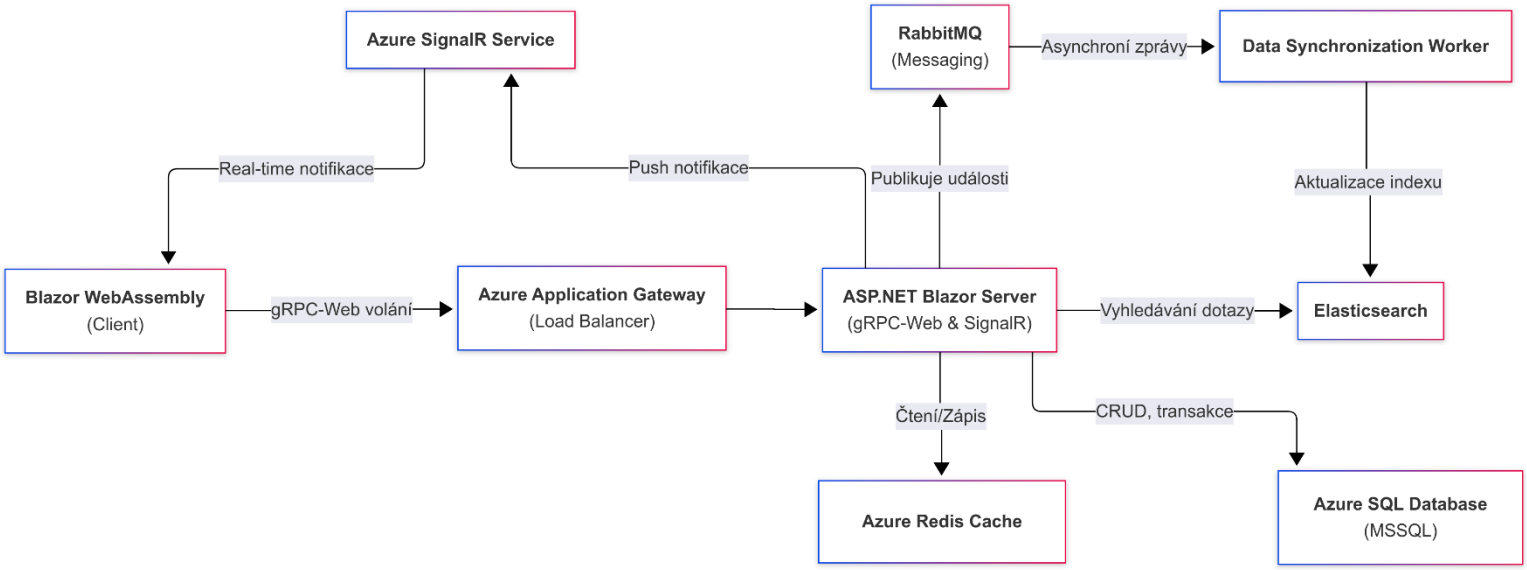
6.1. Class Diagram



6.2. Sequence Diagram – Přidání komentáře



6.3. Component Diagram



7. Výběr vhodných architektur a architektonického stylu

Navržený architektonický styl:

- **Mikroslužbový přístup (modulární architektura):**
  Všechny hlavní komponenty – frontend, backend, messaging, vyhledávací systém a caching – jsou oddělené a komunikují mezi sebou přes jasně definovaná rozhraní (např. gRPC-Web, RabbitMQ, SignalR).
- **Cache-aside pattern:**
  Redis se využívá jako pasivní cache s explicitní invalidací při zápisu do SQL.
- **Event-driven synchronizace:**
  RabbitMQ zajišťuje asynchronní synchronizaci mezi SQL a Elasticsearch, což odděluje transakční operace od vyhledávací logiky.
- **Real-time notifikace:**
  SignalR zajišťuje okamžité push notifikace do klientské části, což umožňuje dynamické aktualizace UI bez nutnosti pollingových dotazů.

Výhodné technologie a jejich použití:

- **ASP.NET Blazor WebAssembly:** Moderní, interaktivní klientská aplikace.
- **ASP.NET Blazor Server s gRPC-Web:** Efektivní zprostředkování dat mezi klientem a serverem.
- **Azure SQL Database (MSSQL):** Spolehlivý zdroj pravdy pro transakční data.
- **Elasticsearch (NEST):** Robustní fulltextové vyhledávání a filtrování obsahu.
- **RabbitMQ:** Asynchronní messaging, který propojuje transakční systém s vyhledávacím indexem.
- **Redis Cache:** Rychlý přístup k téměř statickým datům.
- **SignalR (Azure SignalR Service):** Push notifikace pro real-time interakce.
- **Azure platforma:** Nasazení a správa všech služeb v cloudu s load balancingem, škálovatelností a vysokou dostupností.