VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

INTEGRACE TECHNOLOGIE ELASTICSEARCH DO ERP SYSTÉMU K2

INTEGRATION OF ELASTICSEARCH TECHNOLOGY INTO THE ERP SYSTEM K2

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE **Bc. MICHAL SZYMIK**
AUTHOR

VEDOUCÍ PRÁCE **Ing. VLADIMÍR BARTÍK, Ph.D.**
SUPERVISOR

BRNO 2024

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Zadání diplomové práce

157534

Ústav:	Ústav informačních systémů (UIFS)
Student:	**Szymik Michal, Bc.**
Program:	Informační technologie a umělá inteligence
Specializace:	Strojové učení
Název:	**Integrace technologie Elasticsearch do ERP systému K2**
Kategorie:	Web
Akademický rok:	2023/24

Zadání:

1. Seznamte se s ERP systémem společnosti K2 Atmitec s.r.o., konkrétně s architekturou systému, se způsobem zpracování, uchovávání a získávání dat a aktuálním stavem filtrování a vyhledávání v nich.
2. Seznamte se s technologií Elasticsearch, zhodnoťte její využití v ERP systémech, výhody a nevýhody oproti jiným existujícím nástrojům. Popište v čem může být tato technologie přínosem pro ERP systém K2.
3. Prostudujte existující integrace technologie Elasticsearch.
4. Navrhněte způsob integrace Elasticsearch do ERP systému K2 za účelem efektivnějšího filtrování a vyhledávání v datech s využitím stávajících prostředků, které systém nabízí. Zaměřte se na konfiguraci Elasticsearch serveru, definice indexů a vyhledávací dotazy.
5. Implementujte navržené řešení s využitím dostupných technologií.
6. Proveďte testování vytvořeného řešení ověřením jeho použitelnosti a porovnáním s původními způsoby filtrování a vyhledávání v datech. Zhodnoťte dosažené výsledky.

Literatura:

- Gormley, C., Tong, Z.: Elasticsearch: The Definitive Guide. O'Reilly Media, Inc., 2015. ISBN: 978-144-93-5854-9.

Při obhajobě semestrální části projektu je požadováno:
Body 1-3, částečně bod 4.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce:	**Bartík Vladimír, Ing., Ph.D.**
Vedoucí ústavu:	Kolář Dušan, doc. Dr. Ing.
Datum zadání:	1.11.2023
Termín pro odevzdání:	17.5.2024
Datum schválení:	30.10.2023

Abstrakt

Tato práce se zabývá integrací vyhledávací technologie Elasticsearch do ERP systému společnosti K2 Software s.r.o. V rámci této integrace se řeší jak propojení systému s touto technologií, tak její konfigurace a správa skrze rozhraní klientské aplikace K2. Možnosti vyhledávače jsou pak využity ve stávajícím mechanismu filtrování a vyhledávání a také ve funkcích našeptávače a globálního vyhledávání napříč celým systémem. Výstup této práce je funkční a je nasazen v produkční verzi systému K2.

Abstract

This thesis focuses on the integration of Elasticsearch search engine technology into the ERP system of czech company K2 Software s.r.o. This integration deals with both the connection of the system with this technology and its configuration and management through the K2 client application interface. The search engine capabilities are then used in the existing filtering and search mechanism as well as in the autocomplete and global search functions across the entire system. The output of this work is functional and has been deployed in the production version of the K2 system.

Klíčová slova

Elasticsearch, K2, ERP systém, vyhledávač, našeptávání, filtrování dat

Keywords

Elasticsearch, K2, ERP system, search engine, autocomplete, data filtering

Citace

SZYMIK, Michal. *Integrace technologie Elasticsearch do ERP systému K2*. Brno, 2024. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Vladimír Bartík, Ph.D.

Integrace technologie Elasticsearch do ERP systému K2

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Ing. Vladimíra Bartíka, Ph.D. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Michal Szymik
15. května 2024

Poděkování

Rád bych touto cestou poděkoval Ing. Vladimíru Bartíkovi, Ph.D. za veškeré jeho rady a vedení, které mi poskytl během tvorby této práce. Velký dík patří také panu Ing. Tomáši Szkanderovi ze společnosti K2 Software s.r.o. za jeho velkou podporu, díky které mohla tato práce vzniknou.

Obsah

Seznam obrázků

Kapitola 1

Úvod

Téměř každý dodavatel softwaru je pod tlakem konkurence nebo svých zákazníků neustále zdokonalovat a inovovat svůj produkt. V neustále se měnícím světě je pro udržení se na trhu dodavatel nucen rychle reagovat na změny a přizpůsobovat se aktuálním trendům. A pokud se chce ještě navíc odlišit od své konkurence, musí sám přicházet s novými inovativními řešeními, které posunou jeho produkt zase o něco dále. Toto platí o to více, je-li předmětem diskutovaného softwaru ERP systém, jak je tomu v případě této práce.

ERP systémy jsou určené pro řízení všech možných oblastí a procesů firem, kterým mají usnadňovat každodenní práci v jejich podnikání. Zaměstnanci firem pracují s daty, ať už své vlastní společnosti, nebo svých zákazníků, a s nárůstem těchto dat přirozeně vzniká požadavek na efektivní a rychlé vyhledávání relevantních informací. Často je také potřeba pracovat se specifickou množinou těchto dat, které mají nějakou společnou vlastnost, k čemuž jsou využívány filtry s různě definovanými kritérii. Výše uvedené operace nad daty jde zahrnout do pojmů filtrování a vyhledávání.

Touto problematikou se zabývá i společnost K2 Software s.r.o., která na českém a slovenském trhu poskytuje již přes třicet let svůj ERP systém K2. Ze strany této společnosti vznikl požadavek na začlenění moderní vyhledávací technologie Elasticsearch přímo do jimi dodávaného softwaru, čímž by se zdokonalil stávající systém filtrů a možností vyhledávání. Chtějí tak svým zákazníkům nabídnout nový způsob, jak pracovat s jejich daty a zároveň ukázat, že jsou schopni sledovat aktuální technologické trendy a mají zájem vylepšovat svůj produkt. Na základě tohoto požadavku vzniklo zadání pro tuto diplomovou práci.

V rámci tohoto textu bude popsán celý proces integrace vyhledávací technologie a jejího využití v ERP systému K2. První kapitola se bude věnovat analýze samotného systému K2. Budou v ní popsány základní vlastnosti tohoto softwaru a jeho fungování v kontextu filtrování a vyhledávání dat. Tato část bude sloužit jako podklad pro pochopení následného návrhu a popisu implementace integrace technologie Elasticsearch. V další kapitole bude přiblížena samotná technologie Elasticsearch. Konkrétně budou popsány princip, na kterém je služba založena, vnitřní organizace dat, způsob vyhledávání a nakonec také rozhraní, kterým je se službou možné komunikovat a jež bude při implementaci využito. Závěrem této kapitoly bude kratší povídání o alternativních vyhledávacích technologiích a ohledně již existujících integracích Elasticsearch. Předposlední kapitola se bude věnovat návrhu a implementaci technologie Elasticsearch do systému K2. Postupně bude v kapitole přiblížen způsob, kterým se řeší napojení a správa této služby, a poté způsob, kterým je realizováno

samotné filtrování a vyhledávání dat. Součástí popisu budou také konkrétní implementační detaily, které jsou pro toto řešení podstatné a zajímavé. V neposlední řadě bude v kapitole také popsáno uživatelské rozhraní, skrze které lze konfigurovat a využívat nově integrovanou technologii. Poslední kapitolu tvoří text o testování a vyhodnocení hotového řešení.

Kapitola 2

Informační systém K2

Cílem této kapitoly je seznámit se s informačním systémem K2 od společnosti K2 Software s.r.o. a také obecně s pojmem ERP systém, kterým tento informační systém je. Nejdříve je zde definován pojem ERP systém a jsou popsány jeho základní funkce. Posléze se kapitola věnuje přímo programu K2, a to jeho architekturou, organizační strukturou, možnostmi filtrování a dalšími oblastmi, které souvisí s problematikou filtrování a vyhledávání dat, což je náplní této práce.

2.1 ERP systém

Podnikové organizace po celém světě se zabývají ve větší či menší míře činnostmi jako jsou plánování, zásoby, nákup, prodej, marketing apod. U menších podniků, které se skládají z malého množství zaměstnanců či zákazníků, může být řízení těchto činností relativně jednoduché, ale s postupným rozvojem a zvětšováním organizace se zvyšuje také potřeba efektivně a systematicky spravovat veškerá data a na jejich základě řídit jednotlivé oblasti podniku. Za tímto účelem vznikly takzvané Enterprise Resource Planning (dále jen ERP) systémy, česky systémy pro plánování podnikových zdrojů. Americká výzkumná společnost Gartner definuje ERP jako *„schopnost poskytovat integrovanou sadu nástrojů pro byznys aplikace“* [5]. Tyto nástroje často využívají společné úložiště dat a mají za cíl pokrýt jednotlivé oblastí organizace.

ERP systémy nejsou nijak standardizované[1] a každý takový software se může lišit, ať už v oblastech, které pokrývá, či v mechanismech, na kterých je založen. Nelze jednoduše určit, který ERP systém je nejlepší, jelikož takové hodnocení podléhá subjektivnímu posouzení jednotlivých zákazníků, kteří mají různé nároky a požadavky.

Mezi běžné činnosti firem se podle organizace SAP řadí správa financí (účetnictví), HR[2] (řízení lidských zdrojů), nákup, prodej, výroba, logistika a zásobování, výzkum a správa podnikových aktiv [11]. Různí dodavatelé však nabízejí řízení různých oblasti. Na úrovni softwaru se ERP systémy zpravidla člení do modulů, které řídí zpravidla jednu konkrétní oblast.

[1] Mají ale za cíl implementovat standardy některých oblastí, které řídí.
[2] Human Resources

V dnešní době jsou ERP systémy komplexní softwary distribuované v různých podobách, ať už jako desktopové aplikace, či cloudové služby. Systémy je běžně možné rozšiřovat i o další nástroje od třetích stran.

2.1.1 Existující ERP systémy

Na mezinárodní úrovni existuje obrovské množství ERP systémů. Jedná se například o systémy: SAP[3], Microsoft Dynamics 365[4], NetSuite[5] a další. Vývoj těchto systémů se posouvá obrovským tempem dopředu a snaží se sledovat aktuální trendy jak ve světě byznysu, tak ve světě informačních technologií. Různá porovnání vyzdvihují vlastnosti různých systémů, ale nelze najít jednoznačnou shodu v identifikaci nejlepšího. Při posuzování systému hraje důležitou roli, zda naplňuje očekávané požadavky na řízení a správu jednotlivých oblastí, a zanedbatelná není ani cena, se kterou se pořízení a správa takového systému pojí.

I na úrovni České republiky existuje mimo systém K2 také obrovské množství různých ERP systémů, jako je například: Helios[6], ABRA[7], Money[8] a další, viz seznam systémů od společnosti Asociace za lepší ICT řešení, o.p.s. [1].

2.2 Informační systém K2

Následující část textu se zabývá popisem ERP systému K2, do kterého je v rámci této práce cílem integrovat vyhledávací technologii Elasticsearch. Je zde popsána základní charakteristika systému, jeho architektura a části, ze kterých se skládá. Dále je popsán datový modul jakožto hlavní jednotka řídící konkrétní oblast systému, a také způsob, kterým lze v takovémto modulu filtrovat a vyhledávat data. V poslední části této sekce je stručný náhled na některé implementační detaily systému, které jsou z pohledu této práce podstatné.

Tato sekce vychází převážně z oficiálních webových stránek systému K2 [7], z oficiálních stránek nápovědy tohoto systému [8] a z autorových osobních zkušeností získaných při práci s tímto systémem.

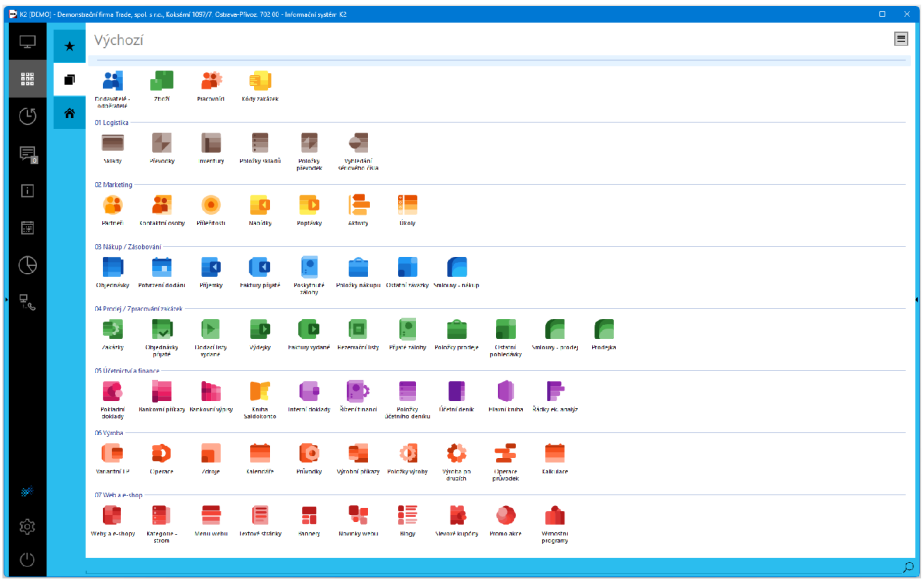
2.2.1 Základní charakteristika systému

Informační systém K2 je podnikový softwarový produkt společnosti K2 Software s.r.o. a je na trhu již více než 30 let. Systém se dotýká velkého množství oblastí plánování a řízení organizace, mezi které patří: výroba, workflow[9], personalistika a mzdy, účetnictví, obchod, prodej a zpracování zakázek, nákup a zásobování, WMS[10], logistika, marketing, CRM[11], web a e-shop, K2 API a další [9]. Společnost reaguje na rychlý vývoj tržního prostředí

[3] https://www.sap.com
[4] https://www.microsoft.com/cs-cz/dynamics-365
[5] https://www.netsuite.com
[6] https://www.helios.eu/
[7] https://www.abra.eu/
[8] https://money.cz/
[9] aplikace pro řízení pracovních postupů
[10] WMS – Warehouse Management System (systém řízení skladů)
[11] CRM – Customer Relationship Management (řízení vztahů se zákazníky)

každoročním vydáváním nové verze, a právě v rámci těchto aktualizací se do systému mají možnost dostat i nové technologie, jako je například vyhledávač Elasticsearch.

Jedním velkým specifikem, kterým se systém snaží odlišit od ostatních, je možnost provádět dodatečné modifikace prostřednictvím tzv. speciálů. Ty umožňují rozšířit standardní oblasti systému, ať už o novou byznys logiku, nebo specifické uživatelské rozhraní, anebo přidat úplně novou oblast plánování a řízení podle konkrétních potřeb zákazníka. Každé oblasti pak přísluší jeden či více datových modulů, které spravují data dané oblasti a poskytují potřebnou funkcionalitu. Obrázek 2.1 ukazuje nabídku datových modulů systému K2 rozdělených podle oblastí zaměření.



Obrázek 2.1: Nabídka běžně používaných datových modulů v systému K2.

Některé oblasti lze také dělit na vlastní firmy. Jedná se o koncept sdílení určitých částí systému mezi například dceřinými společnosti zákazníka, a naopak oddělení částí, které jsou specifické pro jednotlivé vlastní firmy. Na ještě jiné úrovni lze systém dělit na jednotlivé mandanty – klienty, kteří mají všechny oblasti systému oddělené[12], sdílená je pouze systémová konfigurace.

2.2.2 Architektura systému

Celý systém K2 je tvořen kromě desktopového klienta ještě dalšími komponentami. Architektura systému se dělí na tři celky podle možnosti připojení a umístění v síti – LAN[13], Internet, DMZ[14]. Schéma architektury je vizualizováno na obrázku 2.2. Na lokální úrovni systém tvoří v minimální sestavě desktopová aplikace K2 (tlustý klient[15]) a databázový

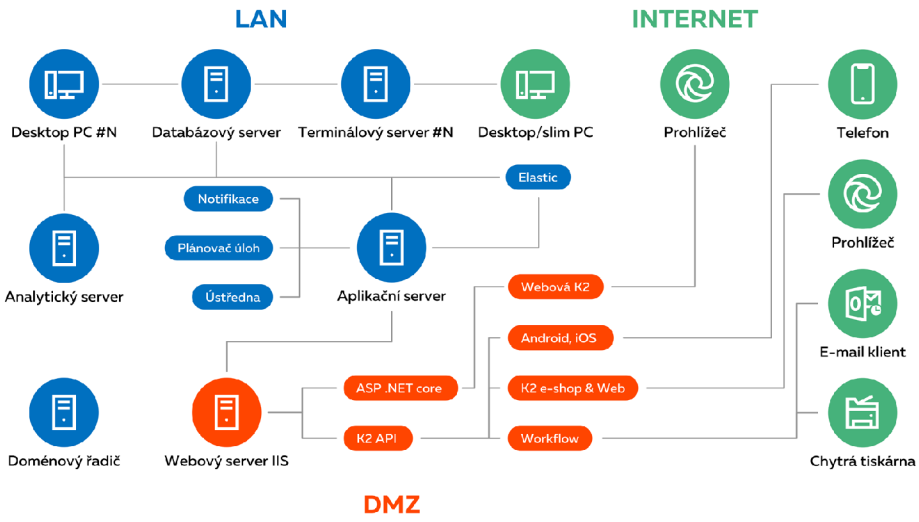
[12]Běžně se nastavuje jeden mandant pro produkci a druhý mandant pro testovací účely zákazníka či konzultantů společnosti K2.

[13]LAN – local area network (lokální síť)

[14]DMZ – demilitarized zone (síť zveřejňující rozhraní do veřejného internetu)

[15]Tlustým klientem se rozumí software, který řeší aplikační logiku i zobrazování (prezentační vrstvu). Tenký klient řeší primárně prezentační vrstvu a pouze minimální množství implementační logiky, je proto závislý na aplikačním serveru.

server (podporovaným je MS SQL Server[16] nebo Oracle Database). Tuto sestavu lze alternativně rozšířit o terminálový server, na který je možné napojit vícero počítačů s tenkým klientem K2.



Obrázek 2.2: Schéma architektury informačního systému K2 převzaté z oficiální nápovědy [8].

Dále se na úrovni lokálního propojení vyskytuje aplikační server, který obsahuje stejné funkce jako desktopová aplikace kromě vizuálních částí. Tento server slouží pro plánování úloh, zasílání notifikací, popř. jako ústředna, jelikož je neustále v provozu. Na aplikační server je pak přímo napojený webový server, který zveřejňuje určité části systému na internet. Mezi zveřejněné části patří webový klient K2 a prostřednictvím K2 API také asistenční mobilní aplikace K2 Assist[17], e-shop, web a workflow. Webový klient obsahuje většinu funkcí, které lze využít v desktopové aplikaci. Systém lze škálovat, a to zvyšováním počtu aplikačních serverů. Pro rozdělení pracovní zátěže a její vyrovnání mezi jednotlivými servery se používá komponenta Load Balancer – vyrovnávač zátěže.

V této architektuře už má místo i server Elasticsearch, který je aktuálně využíván v rámci K2 e-shopu a webu jako běžná webová služba.

Databázové úložiště

V databázi jsou uložená veškerá data organizace, resp. organizací (v případě vícero mandantů), a také společná systémová (konfigurační) data. Instalací K2 tak prakticky vznikají na databázovém serveru minimálně dvě databáze – jedna systémová (konfigurační) a další datové (pro každého mandanta jedna). Jednotlivé tabulky obsahují data konkrétních datových modulů, ke kterým klient K2 přistupuje či které modifikuje prostřednictvím vhodně

[16] MS SQL - Microsoft SQL server
[17] Aplikace není klientem systému, ale umožňuje dvoufaktorové přihlašování a příjem notifikací.

sestavených SQL dotazů. Databázový server lze dodatečně nakonfigurovat i pro účely full-textového vyhledávání, DMS[18] a analytické služby pro technologii OLAP[19].

2.2.3 Datové moduly

Jak již bylo několikrát uvedeno, jádrem řešených oblastí plánování a řízení organizace je datový modul. Datové moduly v systému K2 mají jasně danou hierarchickou strukturu počínaje třídou `TDataM`. Jednotliví potomci této třídy pak dále implementují logiku vlastních firem a legislativy a na vyšších úrovních se poté rozlišují moduly pro dokumenty, pro číselníky apod. Koncová třída datového modulu tedy obsahuje jak obecnou logiku (řešení přístupu, správa záznamů, ...), tak specializovanou funkcionalitu pro účely řešené oblasti.

Základem datového modulu jsou samotná data. Ta se skládají z polí, pokud se na data díváme z pohledu jednotlivých záznamů, resp. sloupců, když data vnímáme jako tabulku. Oba pojmy však bývají mezi sebou často zaměňovány. Pole mohou být buďto fyzická, nebo počítaná, podle toho, ve kterém okamžiku je stanovena jejich hodnota. Datový modul ve většině případů ukládá svoje data do tabulky na databázovém serveru a fyzická pole jsou pak taková pole, která přímo existují v databázi. Fyzické pole může mít v databázi také index, pak se v terminologii K2 hovoří o fyzickém poli s indexem. Počítaná pole se vyhodnocují až na samotném klientovi, tedy jejich hodnota není nikde fyzicky uložena. Je zřejmé, že práce s těmito poli nebude tak efektivní jako práce s fyzickými poli, jelikož jsou obtížena dodatečnou režií pro jejich vyhodnocení.

Mezi běžná pole, na které lze narazit ve většině datových modulů, patří například: identifikátor záznamu (RID[20]), zkratka (`Abbr`), popis (`Description`), časové razítko poslední modifikace (`Timestamp`), pole pro práva, skupiny práv atd. Moduly mají specifikované také tvz. *Quick Search Fields* (pole pro rychlé vyhledání), což jsou pole s indexem, které se používají ve funkci rychlého hledání. Jsou to tedy ve většině případů nejběžněji používaná pole daného datového modulu.

Datové moduly je možné mezi sebou vzájemně propojovat vytvářením vazeb. Vazby jsou realizovány relačně skrze primární a vedlejší klíče (pole datového modulu), tedy na stejném principu, jak je to realizováno v relačních databázích. Z uživatelského hlediska je možné si přes vazební pole[21] zobrazit libovolné hodnoty z jiného datového modulu.

Dalším typem datových modulů je položkový modul. Tento modul umožňuje vytvářet položky ze záznamů jednoho datového modulu k záznamům jiného datového modulu. Například položkový modul *Položky prodeje* může propojit záznam z modulu *Zakázky* se záznamy z modulu *Zboží*.

V rámci nástroje *návrhář objektů* lze vytvářet či modifikovat existující datové moduly, práva či funkční jednotky. Nástroj vyžaduje určitou úroveň znalosti vnitřních struktur, tedy primárně s ním nepracují samotní klienti, ale konzultanti společnosti K2 Atmitec s.r.o, která se zabývá implementací a správou programu K2 u zákazníků.

[18] DMS – Document Management System (systém pro správu dokumentů)
[19] OLAP – Online Analytical Processing (online analytické zpracování)
[20] RID – Row Identifier (identifikátor řádku)
[21] pole realizující vazbu do jiného datového modulu

2.2.4 Filtrování

Filtrování v K2 je jednou ze základních funkcí programu, jelikož se jedná o důležitou operaci s daty, která ovlivňuje efektivitu práce uživatele se systémem. K2 rozlišuje 4 stavy množin záznamů v datovém modulu:

- kniha – obsahuje data podle primárního klíče (Kniha/Období/Číslo, ...), datové moduly v tomto stavu zobrazují vždy data jedné knihy,
- vše – zobrazují se data všech knih najednou (běžně se nevyužívá),
- rychlý filtr – aplikován v režimu *kniha*, využívá indexy tabulek, má však omezené možnosti,
- kontejner – obsahuje záznamy podle libovolných kritérií pro vybranou množinu dat.

Tři z těchto množin (mimo stav *vše*) obsahují již nějakým způsobem filtrovaná data. V případě potřeby specifického filtrování dat lze v K2 vytvářet také samostatné filtry, ve kterých je možné specifikovat další kritéria na požadovanou množinu záznamů. Podmínky, které je možné ve filtrech definovat, jsou následujících typů:

- porovnávací podmínka – porovnání pole datového modulu s hodnotou (skrze operátory: `=`, `<`, `>`, `v seznamu`, `existuje`, `mezi`, ...),
- jiný filtr – křížový výběr, uložený filtr v jednom modulu použitý na pole s vazbou v jiném datovém modulu,
- položková podmínka – vybírá doklady na základě filtru na položkách,
- neřízený příkaz SQL – vlastní podmínky definované v SQL dotazu,
- vnořená podmínka – více podmínek v relaci `AND` nebo `OR`,
- záznamy z kontejneru – omezení filtru na aktuální záznamy v kontejneru,
- aktuální firmy – omezení filtru na záznamy z aktuální vlastní firmy,
- fulltextová podmínka – vyžaduje dodatečnou konfiguraci fulltextových indexů.

Podmínky je možné různě kombinovat a zanořovat v rámci konfiguračního formuláře filtru. Do tohoto systému filtrů je vhodné zavést novou podmínku, využívající technologie Elasticsearch, k rozšíření možností filtrování.

Je zjevné, že možnosti filtrování jsou tedy různorodé a velice komplexní. Rychlost vyhodnocení filtru vychází z rychlosti provedení SQL dotazu, který jej realizuje. Při výběru polí pro porovnávací podmínku je proto podstatné zohlednit typ pole (počítané, fyzické, fyzické s indexem), protože cokoliv, co není fyzicky uloženo v databázi, resp. není indexováno, přidá dodatečnou režii při vyhodnocování, a tím prodlouží celkový čas. Pomalé filtry využívající počítaná pole byly právě jednou z motivací, proč zavést Elasticsearch do K2.

Součástí podmínek filtrů jsou také systémové podmínky. Ty řeší bezpečností aspekt a omezují přístup k datům, na které uživatel nemá dostatečné oprávnění.

2.2.5 Vyhledávání

Pro vyhledávání dat v datovém modulu je možné manuálně procházet seznam záznamů nebo využít lokátor – vstupního pole nad seznamem. Lokátor lze nastavit na jedno z předdefinovaných pole s indexem, čímž dojde k seřazení záznamů podle tohoto pole, a v průběhu zadávání textu do vstupního pole lokátoru se pak automaticky nastavuje záznam odpovídající hledanému výrazu. Pole lokátoru lze přepnout také do režimu *Hledání*, který se chová spíše jako filtr nad poli s indexy.

Ještě další možností je funkce tzv. rychlého vyhledání. Tato funkce využívá opět polí s indexem nebo fulltextové indexy v případě, že jsou pro daný datový modul definované. Výstupem tohoto hledání není nastavení množiny odpovídajících záznamů, ale zvýraznění shod v seznamu záznamů a možnost přeskakování mezi jednotlivými shodami.

2.2.6 Práva, vlastní firmy a mandanti

Při práci s daty organizace je vždy třeba dobře spravovat přístup k datům a dbát na bezpečnost při manipulaci s nimi. Program K2 toto implementuje skrze systém rolí, kde každé roli přísluší daná množina práv a každý uživatel má přiřazenou jednu či více rolí. Role zpravidla odpovídají určitému zaměření zaměstnance – uživatele K2 – v rámci organizace. Jednotlivým rolím je možné libovolně přidávat či odebírat konkrétní práva, resp. skupiny, které určují oprávnění pro prohlížení či změnu předem definované množiny záznamů.

Interně se s právy pracuje na úrovni jednotlivých datových modulů. Různí předkové jednotlivých typů datových modulů (doklady, číselníky, ...) mají svá pole pro práva. Například u dokladů se rozlišuje pět různých polí pro práva, a sice:

- skupinové právo na záznamy (`RightGroupId`),
- skupinové právo na knihu dokladu (přes vazbu do modulu knih – `BookId;RightGroupId`),
- přístup do vlastní firmy (opět přes vazbu do modulu knih – `BookId;CompanyId`),
- skupinové právo na dodavatele/odběratele (přes vazbu do modulu Dodavatelé/odběratelé – `TradingPartnerId;RightGroupId`),
- skupinové právo na nadřazený doklad (přes vazbu do modulu nadřazeného dokladu – `SuperiorRID;RightGroupId`).

U dalších typů datových modulů se jedná zase o jiná pole.

Systém K2 umožňuje také odlišit více tzv. vlastních firem[22]. Lze tedy jednotlivým uživatelům nastavit přístupy do jednotlivých firem podle potřeby. Interně tuto problematiku řeší opět jednotlivé datové moduly, které jsou v případě potřeby rozděleny, a to pomocí speciálního pole, jež určuje, do kterých firem záznam, kniha apod. přísluší.

Pro případ, kdy je třeba firmy kompletně oddělit, se využívá rozdělení na mandanty. Mandant v systému K2 je samostatný klient v rámci organizace. Využívá svou vlastní databázi pro uložení dat a sdílí pouze základní konfigurace programu K2.

[22] Z anglického multi-company. Jedná se o schopnost systému spravovat a integrovat procesy více nezávislých firem nebo divizí v rámci jedné organizace.

2.2.7 Implementace K2

Program K2 je napsaný převážně v jazyce Delphi (objektový pascal)[23] s vlastní nadstavbou a některé části systému (například frontend webového klienta) jsou napsané v jazyce JavaScript. Systém má vlastní implementaci objektového modelu a prezentační vrstvy, čímž je schopný docílit konzistentního prostředí napříč celým programem. Celý projekt K2 je členěn do velkého množství jednotek, které realizují řešení konkrétních problémů, a tyto jednotky jsou dále sdruženy do logických celků podle jejich zaměření nebo typu. Například jednotky implementující funkce jádra, datových modulů, prezentační vrstvu apod. lze v projektu K2 nalézt pospolu.

Ve zdrojovém kódu K2 je k dispozici také spousta pomocných funkcí a tříd, kterých lze v této práci využívat. Za zmínku stojí například třída implementující endpoint klienta `TEndpointClient`, kterou lze využít pro napojení na externí webovou službu či API[24]. Další sadou pomocných funkcí jsou funkce pro ukládání a načítání dat objektů na různých umístěních v databázi. Objekty lze ukládat jak na úrovni mandantů, tak i celého systému. Toho lze využít například při ukládání objektů s globálními konfiguracemi, které je třeba načítat při každém spuštění systému.

Objektový model

Jak již bylo výše uvedeno, systém má také implementován svůj vlastní objektový model, který specifikuje základní třídy pro reprezentaci objektů, kolekcí, položek kolekcí atd. Základní třída pro objekt je rozšířena o metadata a virtuální metody, které pak umožňují konzistentní práci s těmito K2 objekty.

Prezentační vrstva

Prezentační vrstva je realizována přes tzv. modré formuláře. Jedná se o hierarchický systém tříd – tzv. prezentérů, které odpovídají příslušným třídám objektového modelu a třídám datových modulů a řeší jejich zobrazování v rámci uživatelského rozhraní programu. Výsledná podoba zobrazení je dána tzv. fragmenty – definicemi jednotlivých komponent objektu v jazyce `XML`, podle kterých je následně generován konkrétní formulář pro daný prezentér. Primární barvou v prostředí K2 je modrá, odtud tedy pojem modré formuláře.

Fragmenty se ve výchozím stavu generují automaticky pro každý prezentér a lze je poté modifikovat prostřednictvím *návrháře formulářů*, a to na úrovni továrního nastavení, instalace konkrétní instance, skupiny uživatel či jednoho konkrétního uživatele. Mezi standardní komponenty formuláře patří na úrovni polí objektů či datových modulů komponenty `label`, `input`, `checkbox` a další. Slouží k zobrazování různých typů hodnot polí ve specifických formátech a komponenta `input` navíc umožňuje zaznamenávat textové vstupy uživatele a ukládat je do příslušného objektu. Na úrovni kolekcí záznamů se pak používají komponenty `grid data` či `grid data simple`, které zobrazují seznamy položek v podobě tabulky. Řádky této tabulky jsou jednotlivé záznamy a sloupce jsou hodnoty jejich polí. Záhlaví tabulky navíc obsahuje lištu, ze které je možné vyvolávat akce související s danou kolekcí

[23] https://www.embarcadero.com/products/delphi/features/delphi

[24] API – Application Programming Interface (aplikační programovací rozhraní)

záznamů. Je možné také ovlivnit, které sloupce (pole) se mají v tabulce zobrazit. Pro specifikaci rozložení formuláře se pracuje s komponentami `group box`, `panel`, `stack panel` apod. Akce formuláře jsou běžně k dispozici na komponentě lišty `ribbon` a hlavní navigace v rámci formuláře je řízena komponentou `tab control`, která rozděluje komponenty do jednotlivých záložek. Tímto nástrojem je možné měnit rozložení jednotlivých komponent na formuláři, jejich styl a podobu, a také jím lze přidat nové akce, ať už v podobě tlačítek, či položek na liště.

V rámci prezentační vrstvy lze přidávat dodatečnou logiku pro zobrazované objekty tak, aby práce s nimi naplnila konkrétní uživatelské požadavky. Rozšiřování funkcionality je možné buď připojením vlastní akce k danému prezentéru, nebo specifickou implementací virtuálních metod daného prezentéru. Vlastní akce se používají v případě, že je třeba implementovat logiku nad rámec existujících standardních akcí jednotlivých typů prezentérů. Akce mají přístup k zobrazovanému objektu a můžou tak realizovat například kontrolu dat, komunikaci s jiným datovým modulem či externím nástrojem. Každý typ prezentéru poskytuje sadu virtuálních metod, kterými lze specifikovat chování v určitých situacích. Jde tak například provést akce před uložením formuláře, po zadání vstupu do pole, anebo automatická kontrola vstupu podle požadovaných kritérií.

Kapitola 3

Vyhledávací technologie Elasticsearch

Snad každá aplikace, která pracuje s větším množstvím dat, nabízí svým zákazníkům nějakou možnost v nich vyhledávat či filtrovat. Typickým příkladem jsou e-shopy, které nabízejí řady produktů, zatímco jejich zákazník hledá jenom jeden konkrétní z nich nebo nějakou kategorii, ze které by si chtěl vybrat. A aby nemusel sekvenčně procházet celý seznam produktů, nasazují se různé pomocné nástroje, díky kterým lze na základě konkrétního dotazu vymezit určitou množinu záznamů, které uživatel hledá.

Existuje spousta vyhledávacích služeb, které se liší ať už technikou vyhledávání, dotazováním, či způsobem uložení dat. Každá služba má své výhody a nevýhody a při jejím výběru je třeba zvážit veškeré požadavky a technické okolnosti. Pro účely tohoto projektu byla vybrána právě technologie Elasticsearch, a to z toho důvodu, že není v systému K2 úplně novinkou. Tohoto nástroje už využívá K2 e-shop v rámci produktu SmartSearch, který realizuje komplexní vyhledávání zboží. Na základě dobré zkušenosti vznikl nový požadavek na integraci této technologie i do samotného klienta K2, ať už webového, či desktopového.

Tato kapitola se věnuje základní analýze vyhledávače Elasticsearch. Je zde popsána základní charakteristika této technologie, prvky, ze kterých se skládá a se kterými pracuje, a také možnosti analýzy dat. Dále je zde popis dotazovacího jazyka a rozhraní, přes které lze se službou komunikovat. Poslední části této kapitoly se věnují popisu alternativních vyhledávacích technologií a existujícími řešeními integrací těchto nástrojů do ERP systémů.

Následující kapitola čerpá převážně z oficiálních stránek technologie Elasticsearch [3] a její oficiální dokumentace [2].

3.1 Základní charakteristika

Elasticsearch je vyhledávací a analytická distribuovaná technologie z roku 2010. Jedná se o část platformy Elastic Stack vyvinuté společností Elastic NV. V rámci této platformy se nabízejí kromě Elasticsearch vyhledávače také produkty Kibana, Beats a Logstash, které umožňují další práci s daty, jako je vizualizace, analýza, transformace apod. Tyto další produkty však nejsou předmětem této práce. Vyhledávač Elasticsearch je založený na knihovně `Lucene`, volně dostupném softwaru od vývojářů společnosti The Apache Software Foun-

dation, „standardem“ mezi vyhledávacími aplikacemi. Elasticsearch nabízí také implementace klientů, kteří jsou oficiálně podporovaní v jazycích Java, .NET, PHP, Python a Ruby. Pro práci s vyhledávačem však stačí využívat rozhraní typu REST. [6]

3.1.1 Apache Lucene

Spousta vyhledávacích technologií je založena na fulltextové vyhledávací knihovně Lucene. Jedná se o open source produkt od společnosti The Apache Software Foundation napsaný v programovacím jazyce Java, jehož cílem je poskytovat funkce pro indexování a vyhledávání dat s možností textové analýzy. Knihovna pracuje s efektivními algoritmy, které umožňují různé druhy vyhledávání indexovaných dat. Mezi ty patří například `rank searching` (seřazuje výsledky hledání podle skóre), `fielded searching` (vyhledávání v různých polích), `nearest-neighbor search` (vyhledávání „podobných“ záznamů), tolerance překlepů a mnoho dalšího. Tím, že je knihovna implementována v jazyce Java, je snadno přenositelná mezi různými platformami. [13]

3.1.2 Struktura vyhledávače

Elasticsearch je schopný vyhledávat data s odezvou téměř v reálném čase. To je důsledkem jak efektivního indexování a vyhledávání pomocí knihovny Lucene, tak také díky jeho distribuovanému charakteru. Na nejvyšší úrovni se vyskytuje tzv. *cluster* (česky shluk). Ten se skládá z jednoho a více *uzlů*, které je možné do clusteru přidávat, a tím rozložit zátěž služby. Přidáváním či odebíráním uzlů se také řeší problematika škálování. Uzly umí mezi sebou komunikovat a předávat si vzájemně informace podle potřeby. Na uzly jsou ukládány a indexovány konkrétní množiny dat ve formě tzv. *shards* (česky střepy). Ty mohou být buď primární, nebo replikované, podle toho, zda obsahují originální data, nebo jejich kopie. Jeden či více shards tvoří *index*, který sdružuje data, jež spolu nějakým způsobem souvisí. Jednotlivé *shards* jsou rozmístěny mezi uzly tak, aby byla zátěž rovnoměrně rozložená, a v případě potřeby jsou mezi nimi přesouvány. Replikované *shards* zpravidla bývají na jiných uzlech, aby v případě výpadku jednoho uzlu byla data k dispozici na jiném.

Výše uvedené komponenty Elasticsearch řeší interní způsob rozložení dat a realizují tak distribuovaný charakter této technologie. Běžný uživatel se však spíše setká s abstrakcí tohoto interního mechanismu, kdy jsou data členěná do indexů. Následující odstavce popisují jednotlivé prvky a pojmy spojené s touto datovou strukturou.

Dokument v Elasticsearch je základní jednotka zpracovávané informace. Tato entita je definována v jazyce JSON, tedy umožňuje popsat data prakticky libovolné struktury. Každý dokument se skládá z metadat, jako je například jedinečný identifikátor dokumentu (`_id`), identifikátor indexu (`_index`). originálního JSON popisu dat dokumentu (`_source`) a dalších, a ze samotných uživatelských dat, která jsou reprezentována poli příslušného datového typu. Elasticsearch rozlišuje následující datové typy: binární, logické (boolean), klíčová slova, čísla, data, aliasy, objekty, strukturované datové typy, jako jsou např. ip adresy, rozsahy, . . . , prostorové datové typy a mnoho dalších. Určení datového typu pole je klíčové pro efektivní interní uložení dat. Na dokumenty lze pohlížet jako na řádky v terminologii relačních databází. Dokumenty se stejnou strukturou polí jsou sdružovány do indexů.

Vnitřní struktura dokumentů může být definována buďto dynamicky, tedy systém sám odvozuje datový typ daného pole, nebo explicitně pomocí tvz. **mapování polí**. Při procesu

mapování jsou specifikovány datové typy polí, a tím i způsob, jakým je dokument interně uložený a indexovaný. Kromě datových polí dokumentu lze také v definici zahrnout pole s metadaty dokumentu, a tím ovlivnit způsob uložení i těchto polí. Přidání polí do dokumentu je možné i do již existujícího dokumentu. Naopak odebrání či změna typu pole již není reálná bez opětovné indexace dat.

Index je tedy kolekce dokumentů. Každý index má svůj název, který je jedinečný v rámci jedné instance služby Elasticsearch a který se využívá při veškerých operacích s ním. Interně se indexem v terminologii Elasticsearch myslí tzv. *invertovaný index*. Jedná se o mechanismus, který mapuje obsah dokumentů na úrovni dále nedělitelných entit, tzv. *termů*, na jejich příslušející umístění v jednotlivých dokumentech. Zjednodušeně, index si lze představit jako tabulku se dvěma sloupci. První obsahuje termy a druhý seznam dokumentů, do kterých term na daném řádku přísluší. Tento mechanismus „od termů k dokumentům“ v kombinaci s rozdělením indexu mezi jednotlivé uzly distribuovaného systému umožňuje značně zefektivnit fulltextové vyhledávání, což je jednou z hlavních výhod Elasticsearch vyhledávání.

3.1.3 Analýza dat

Způsob, kterým Elasticsearch analyzuje data dokumentů a kterým vytváří jednotlivé termy, je ovlivnitelný použitím tzv. analyzátorů. Ty jsou buďto výchozí, nebo uživatelsky definované při vytváření indexu. Analyzátorem se rozumí množina analytických pravidel, kterými se zpracovávají data jednotlivých polí dokumentu. V rámci analyzátoru lze definovat různé filtry, které se aplikují na vstupní text, nebo tokenizéry[1], podle kterých se text rozděluje na jednotlivé termy. Analýza probíhá jak během indexování dat, tak při samotném vyhledávání, kdy je možné zpracovat analyzátorem samotný vyhledávaný text. Elasticsearch nabízí celou řadu vestavěných analyzátorů, kdy za zmínku stojí například `standardní analyzátor`, který rozděluje text podle hranic slov, přičemž odstraní většinu diakritiky a převede text na malé písmo. Kromě možnosti napsat si pravidla pro vlastní analyzátor lze také využít existující analyzátory, které jsou veřejně dostupné v rámci velké komunity, která existuje okolo technologie Elasticsearch.

Analyzátory je možné definovat v rámci šablony indexu, která se používá k prvotní a základní konfiguraci při vytváření indexu na Elasticsearch serveru. Analyzátory definované v šabloně indexu pak lze použít při definici mapování polí a také při samotném vyhledávání dat. Příklad takové šablony definující dva analyzátory je v kódu 3.1.

```
{
  "settings": {
    "index": {
      "analysis": {
        "filter": {
          "lowercase": {
            "type": "lowercase"
          }
        },
        "analyzer": {
          "keywords": {
            "filter": ["lowercase","icu_folding"],
            "tokenizer": "keyword"
```

[1] nástroj, který rozděluje text na menší celky

```
14          },
15          "keywords_autocomplete": {
16            "filter": ["lowercase","icu_folding"],
17            "tokenizer": "autocomplete_keywords"
18          }
19        },
20        "tokenizer": {
21          "autocomplete_keywords": {
22            "max_gram": "10",
23            "min_gram": "2",
24            "type": "edge_ngram"
25          }
26        }
27      }
28    }
29  }
30 }
```

Výpis 3.1: Příklad Elasticsearch analyzátoru.

Analyzátory jsou definované v JSON objektu `analyzer`. Nazývají se postupně `keywords` a `keywords_autocomplete` a jejich účelem je zpracovat indexovaný text tak, že na něj aplikuje filtr `lowercase`, který převede celý text na malá písmena, a filtr `icu_folding`[2], který provádí *unicode* normalizaci (zbaví se diakritiky apod.). První analyzátor používá tokenizér `keyword`. Jedná se o vestavěný tokenizér, který vytváří výstupní term, jež přesně odpovídá vstupnímu textu (tedy jej nijak nerozdělí). Druhý analyzátor zase používá uživatelsky definovaný tokenizér `autocomplete_keywords`, typu `edge_ngram`[3], který vstupní text rozděluje na termy obsahující minimálně dva první znaky (dáno parametrem `min_gram` nastaveným na hodnotu 2) a maximálně prvních 10 znaků (dáno parametrem `max_gram`, který je roven 10). Tento tokenizér je vhodný například pro implementaci našeptávače.

3.1.4 Dotazovací jazyk

Jádrem vyhledávače Elasticsearch je realizace vyhledávání prostřednictvím dotazu, jehož podoba je přesně definovaná pomocí abstraktního syntaktického stromu. Dotaz se skládá ze dvou typů klauzulí:

- koncové dotazy - samostatně použitelné dotazy, jejichž podstatou je hledat konkrétní hodnotu v konkrétním poli,
- složené dotazy - dotazy kombinující jiné složené dotazy nebo přímo koncové (listové) dotazy.

Při zpracování dotazu se zohledňují dva kontexty – dotazovací a filtrovací. Hlavním rozdílem mezi těmito dvěma kontexty je, jestli je výsledek hledání zohledněný v meta poli `_score`, které určuje, jak moc výsledek odpovídá hledanému dotazu relativně ke všem nalezeným

[2]Použití tohoto filtru vyžaduje nainstalovat rozšiřující modul ICU Analysis do služby Elasticsearch.

[3]N-gram je pojem definovaný jako posloupnost `n` po sobě jdoucích souvisejících symbolů, viz https://www.mathworks.com/discovery/ngram.html. Edge n-gram vychází z předchozí definice, ale omezuje posloupnost pouze na prvních n symbolů posloupnosti, viz https://www.elastic.co/guide/en/elasticsearch/reference/8.13/analysis-edgengram-tokenfilter.html.

výsledkům. Filtrovací kontext na skóre vůbec nehledí, a navíc často používané filtry jsou interně v Elasticsearch kešované[4].

Složené dotazy tvoří tzv. *boolean query* (logický dotaz), který umožňuje kombinovat dotazy pomocí logických operátorů, a pak dotazy, které různým způsobem ovlivňují výsledné skóre. Boolean query v terminologii Elasticsearch pracuje s pojmy `must`, `should` a `must_not`, které odpovídají po řadě logickým operátorům `AND`, `OR` a `NOT`. Ty pracují v dotazovacím kontextu, tedy kromě výsledku se vrací i jeho skóre. Alternativně lze místo `must` použít `filter`, který funguje stejně, ale ve filtrovacím kontextu. Boolean query je základem téměř každého netriviálního vyhledávacího dotazu.

Koncové dotazy se dále dělí na fulltextové, prostorové, tvarové, specializované a další. Fulltextové dotazy slouží k prohledávání analyzovaného textu. Standardním příkladem je třeba `match query` (dotaz na shodu), který na základě analyzovaného hledaného textu vrací doklady, ve kterých v zadaném poli nalezne shodu. Tento typ dotazu umožňuje také možnost *fuzzy* shody, která vrací i výsledky, které se liší v určeném počtu znaků. Dotaz je možné dále vyladit velkým množstvím parametrů tak, aby odpovídal požadavkům tázajícího. Za zmínku stojí také `multi_match query` (dotaz na více shod), který oproti předešlému dotazu `match query` hledá shodu ve více specifikovaných polích dokumentu najednou.

Příklad možného dotazu demonstrujícího jeho různé komponenty je možné vidět v JSON kódu 3.2.

```
{
  "size": 100,
  "query": {
    "bool": {
      "should": [
        {
          "multi_match": {
            "query": "ab group",
            "fields": [ "Name.hunspell^2", "Name.words" ],
            "operator": "or",
            "fuzziness": "auto",
            "prefix_length": 2,
            "boost": 1.5
          }
        },
        {
          "match": {
            "Abbr": {
              "query": "AB GROUP"
            }
          }
        }
      ],
      "minimum_should_match": 2
    }
  }
}
```

Výpis 3.2: Příklad Elasticsearch dotazu.

Dotaz se skládá z parametru `size`, který omezuje počet nalezených shod na 100 dokumentů, a ze složeného logického dotazu `query.bool.should`. Ve složeném dotazu jsou nadefinované

[4]Ukládané v mezipaměti za účelem rychlejšího přístupu.

dva koncové dotazy, které podle parametru `minimum_should_match`, jenž je nastaven na hodnotu dva, musí být oba splněny. Prvním koncovým dotazem je `multi_match`, ve kterém je v parametru `query` nastavený hledaný výraz „ab group“. Dále jsou v dotazu specifikována pole, ve kterých má hledání probíhat (`fields`), počet znaků, které se nemusí přesně shodovat (`fuzziness`)[5], a hodnota, o kterou má být vynásobeno skóre nalezeného dokumentu, pokud je shoda právě v tomto dotazu (`boost`). Skóre lze ovlivnit i na úrovni jednotlivých polí, a to pomocí symbolu stříšky, jak je to v poli `Name.hunspell^2`. Druhý dotaz je o něco primitivnější, jelikož předmětem vyhledávání je pouze jedno pole `Abbr`. Parameter `query` opět specifikuje hledaný výraz.

Dokumentace Elasticsearch poskytuje mnohem širší popis jednotlivých vyhledávacích dotazů, to již však není předmětem této práce.

3.1.5 Rozhraní Elasticsearch

Pro komunikaci s Elasticsearch serverem se používá `REST API`[6]. Přes toto rozhraní lze jak konfigurovat Elasticsearch server, tak také využívat jeho funkce. Komunikace probíhá prostřednictvím kanálu `HTTP`, reps. `HTTPS` pro zabezpečenou komunikaci. Tělo požadavku má být ve formátu JSON v kódování UTF-8. Jednotlivá volání API sdílí společné možnosti týkající se převážně formátování vrácených výsledků nebo také filtrování částí JSON odpovědi.

Služba Elasticsearch zveřejňuje velké množství API, které se zabývají konkrétními oblastmi[7]. Mezi nejvýznamnější oblasti patří správa indexů (`Index API`), která se zabývá jejich tvorbou, mazáním, definicí mapování a dalšími. Další oblastí je správa dokumentů (`Document API`), která řeší nahrávání či mazání dokumentů z indexu. V rámci tohoto rozhraní je možné hromadně nahrávat data skrze `bulk` (česky hromadné) operace. Mazání dokumentů je možné i podle zadaných kritérií použitím koncového bodu `delete_by_query`. Neméně důležitá oblast zahrnuje funkce pro vyhledávání (`Search API`). Toto API podporuje zpracovávání jednoho či více vyhledávacích dotazů, asynchronní hledání, umožňuje realizovat funkce našeptávače či skrolování. Vyhledávání je také možné testovat, ať už formou profilování, či validace analyzátorů apod. Další funkce API se zabývají například logováním, skriptováním, analýzou dat nebo ověřením stavu clusterů či jednotlivých uzlů a mnoha dalšími.

3.1.6 Nástroje pro práci s Elasticsearch

Kromě oficiálních nástrojů pro práci s Elasticsearch, jako je třeba Kibana, stojí za zmínku nástroj `Cerebro`, který nabízí webové uživatelské rozhraní pro přehled indexů a aktuálního stavu Elasticsearch serveru. Prostřednictvím tohoto nástroje lze jednoduše sledovat a spravovat vytvořené indexy a přímo volat API, čímž lze také testovat hledací dotazy. Nástroj je veřejně dostupný ve formě projektu na serveru GitHub patřícímu vývojáři Imenezes[8].

[5] V případě tohoto dotazu je hodnota nastavená na `auto`, což stanovuje počet znaků, které se nemusí shodovat, podle délky hledaného výrazu.

[6] REST API – Representational State Transfer Application Programming Interface (česky aplikační programovací rozhraní reprezentačního stavového přenosu)

[7] Kompletní seznam viz https://www.elastic.co/guide/en/elasticsearch/reference/current/rest-apis.html

[8] https://github.com/lmenezes/cerebro

3.2 Alternativní vyhledávací technologie

Pro úplnost následuje krátké shrnutí dalších vyhledávacích nástrojů, které jsou více či méně podobné technologii Elasticsearch.

Apache Solr[9] je otevřená platforma založená také na knihovně Lucene. Nabízí fulltextové indexování a vyhledávání, umožňuje škálování a slibuje odolnost vůči chybám. Komunikaci se službou je možné realizovat přes rozhraní REST. Je dodávána s administrátorským rozhraním, skrz které jej lze konfigurovat[14]. Oproti Elasticsearch se liší například ve způsobu rozmísťování *shards*, které probíhá na této platformě staticky a vyžaduje v případě potřeby manuální zásah[10].

Sphinx[10] je také otevřená platforma, která slouží pro fulltextové vyhledávání, napsaná v jazyce `C++`. Umožňuje indexovat data přímo z SQL či NoSQL databází nebo z libovolných jiných zdrojů. Zaměřuje se na výkon indexace a vyhledávání a podobně jako Elasticsearch i na škálovatelnost. Mezi klíčové vlastnosti patří indexace dat v reálném čase, podpora jiných než textových hodnot, přímá indexace dat z SQL či NoSQL serveru, jednoduchá integrovatelnost, ohodnocování výsledků hledání podle relevance apod. [12]. Sphinx nabízí dost podobné možnosti jako Elasticsearch. Rozdíl je zde především ve škálovatelnosti, kdy Sphinx potřebuje ke svému fungování méně paměti, naproti tomu Elasticsearch potřebuje více, a proto se škáluje na úrovni uzlů, které mohou být obecně na různých zařízeních. Elasticsearch také není provázán s jinými databázovými úložišti, ale zato umožňuje spravovat více typů dat. [10]

3.3 Existující řešení integrace Elasticsearch

Při studiu existujících produktů ERP softwaru nebyl nalezen žádný způsob integrace technologie Elasticsearch. Elasticsearch bývá hlavně využíván na webových aplikacích či na e-shopech a pro jeho integraci se primárně využívá přímo platforma Elastic Stack[11], v rámci které lze spravovat, vizualizovat a analyzovat data dané aplikace. Toto řešení by však nevyhovovalo požadavkům tohoto projektu, jelikož jedním z jeho cílů je začlenit tuto technologii do systému tak, aby vyžadovala pokud možno co nejmenší nutnost zásahu uživatele do jejího zprovoznění a následného využívání. Bylo by navíc třeba společně se systémem K2 distribuovat i platformu Elastic Stack, u každého zákazníka ji nasadit a nakonfigurovat a pak ještě poskytovat zákaznickou podporu při různých problémech, které by mohly vzniknout.

Elasticsearch je využíván v systému K2 v rámci produktu SmartSearch, který realizuje vyhledávání na e-shopu. Tento produkt indexuje zboží, značky a kategorie, které jsou zobrazovány na e-shopu zákazníka. V rámci systému K2 lze nastavit automatickou synchronizaci dat, případně ručně upravit strukturu dokumentů v případě specifických požadavků zákazníka. Tato forma „integrace“ technologie tak zdaleka není využitelná pro účely této práce, kterým je vyhledávání a filtrování v rámci systému K2. Neumožňuje vlastní definice vyhledávacích dotazů a hlavně není schopná indexovat data libovolných datových modulů.

[9] https://solr.apache.org/
[10] http://sphinxsearch.com
[11] https://www.elastic.co/elastic-stack

Kapitola 4

Návrh řešení a implementace

Hlavním úkolem této práce je vymyslet způsob, kterým lze integrovat technologii Elasticsearch do systému K2 tak, aby bylo možné využívat její funkce a možnosti při různých vyhledávacích a filtrovacích problémech. Převážně se práce zaměřuje na využití této technologie jako alternativy a rozšíření stávajícího způsobu filtrování a vyhledávání, který má jistá omezení, obzvlášť pokud dotaz nad daty obsahuje počítaná pole, jež omezují rychlost jeho provedení. V neposlední řadě je snahou navrhnout integraci tak, aby byla rozšiřitelná i do dalších oblastí systému, kde by mohlo být také možné z této technologie profitovat.

Náplní této kapitoly je podrobný popis návrhu možného řešení a souběžně také popis jeho implementace. Některé implementační detaily vyvstaly na povrch až v průběhu samotné práce, jsou ale také zohledněny v následujícím textu. Kapitola je členěna do více logických celků, které adresují jednotlivé řešené problémy. Postupně je tak řešen způsob konfigurace připojení k Elasticsearch serveru a další společná nastavení a poté způsob správy jednotlivých indexů. Dále se text bude věnovat možnostem filtrování a vyhledávání za pomocí nové Elasticsearch podmínky a také funkci našeptávače. Poté bude následovat popis globálního vyhledání napříč více datovými moduly. Na závěr kapitoly je uveden stručný popis zdrojových kódů, do kterých bylo při implementaci zasaženo.

4.1 Konfigurace Elasticsearch

V prvé řadě bylo třeba vyřešit problém připojení a komunikace se serverem Elasticsearch a také vymyslet způsob, jak provádět společná nastavení pro integraci této nové technologie. Za tímto účelem vznikl v K2 nový modul, který se nazývá *Konfigurace Elasticsearch*. Pod tímto pojmem se bude dále v práci rozumět obecné nastavení vyhledávače Elasticsearch v systému K2, zahrnující výše zmíněné řešené problémy. Jedná se o nastavení, které je společné napříč všemi mandanty dané organizace. Mimo to je tato konfigurace i vhodným místem pro nastavení a správu šablon pro tvorbu indexů, u kterých lze očekávat a je i žádoucí, že budou společné pro všechny indexy.

4.1.1 Komunikace se serverem

Komunikace s Elasticsearch serverem je možná buď skrze existující implementace klienta, nebo přes rozhraní REST API. Vzhledem k tomu, že v době psaní tohoto textu neexistovala žádná oficiální implementace klienta v jazyce Delphi[1], který by šel jednoduše zakomponovat do systému K2, bylo jednoznačnou volbou využít rozhraní REST API. Jádro systému K2 navíc již obsahuje potřebné třídy a metody pro komunikaci s tímto druhem rozhraní.

Pro vytvoření spojení se serverem, na kterém je technologie Elasticsearch instalována, je potřeba znát jeho adresu a číslo portu, na kterém služba běží. Je tu také požadavek na zabezpečení komunikace se serverem, během které dochází k přenosu dat, které mohou být pro zákazníky citlivé. Tomu jde samotná služba Elasticsearch naproti a sama ve výchozím stavu komunikuje skrze zabezpečený protokol HTTPS, díky kterému je přenos šifrován. Pro autentizaci klienta se používá tzv. *basic authentication*[2], podle kterého server na základě platných přihlašovacích údajů (jméno a heslo) umožňuje klientovi komunikovat. Elasticsearch umožňuje i alternativní způsoby autentizace, pro účely tohoto projektu však postačí výše uvedený způsob.

Z pohledu integrace do systému K2 bylo na úrovni uživatelského rozhraní řešeno získávání a uchovávání potřebných údajů o serveru a přihlašovací údaje uživatele služby Elasticsearch. Na úrovni logiky je pak implementováno zpracování těchto údajů – jejich uložení a zakomponování do jednotlivých volání dotazů API.

4.1.2 Šablony indexů

Jednou z užitečných funkcí Elasticsearch vyhledávače je možnost hloubkové textové analýzy. Tato funkce umožňuje přizpůsobit vyhledávání na míru jednotlivým textovým polím indexu a samotnému vyhledávanému výrazu, jejichž hodnoty mohou mít obecně různou strukturu. V rámci datových modulů K2 se lze setkat s určitými typy polí, jejichž funkce je napříč moduly stejná. Takovými poli jsou například zkratka, název, popis a další. Každý takovýto typ pole se pak řídí jinými pravidly a má jinou strukturu, například pole pro zkratku může obsahovat libovolný textový řetězec (technickou zkratku), zatímco pole pro název bude spíše nějaké lidsky srozumitelné označení. Pro každý typ pole tak může být efektivnější použít jiný analyzátor, které nejlépe respektuje jeho strukturu či funkci. Jednotlivé analyzátory se pak definují právě při vytváření indexu.

Šablonou indexu se v tomto případě rozumí společná definice analyzátorů a dalších nastavení indexu, které zohledňují odlišnou strukturu indexovaných dat a specifikují tím zaměření samotného indexu. Cílem šablon je zjednodušit tvorbu nových indexů v rámci systému K2, jež sdílejí stejnou funkci.

Vzhledem k tomu, že nastavení textové analýzy není triviální záležitostí, pracuje se v tomto projektu s výchozími šablonami, které definují sadu analyzátorů, které jsou vhodné pro většinu typů polí v K2. Je také respektována potřeba zákazníka přizpůsobit si analýzu přesně na míru svým datům, a proto je umožněno tyto výchozí šablony editovat či přímo vytvářet nové.

[1] https://www.elastic.co/guide/en/elasticsearch/client/index.html
[2] česky jednoduché ověření

Koncept *šablon indexů* existuje i v samotné technologii Elasticsearch, která umožňuje ukládat společné definice přímo na serveru služby. Toto řešení však nedává z pohledu integrace Elasticsearch úplně smysl, jelikož se šablonami je třeba pracovat právě na úrovni systému K2, v rámci kterého je lze případně i modifikovat.

4.1.3 Implementace logiky

Pro modul *Konfigurace Elasticsearch* vznikla třída `TESConfiguration`, v rámci které jsou uložené a spravované všechny potřebné informace. Při prvním přístupu k této konfiguraci dojde k vytvoření globálního objektu v databázi systému K2, a tím je tak k dispozici všem mandantům dané organizace.

V rámci tohoto objektu je možné spravovat jak připojení na Elasticsearch servery, tak i šablony pro vytváření indexů. Připojením se zde rozumí uchování údajů potřebných k realizaci komunikace se serverem. Těmito údaji jsou:

- adresa serveru a číslo portu služby (např. `https://localhost:9200`[3]),
- uživatelské jméno
- a heslo.

Kromě toho je zde také možnost definovat hodnotu `timeout`, která určuje, jak dlouho má klient čekat na odpověď serveru. Pro každé nově vytvořené připojení se určuje jeho zaměření a to buď „pro K2“, nebo „pro Eshop“. Toto umožňuje rozlišit mezi nastavením Elasticsearch serveru pro účely již existujícího vyhledávání na K2 e-shopu, které je řešeno nezávisle na tomto projektu, a pro účely vyhledávání v K2, které je předmětem této práce. Jedno připojení jde použít i pro obojí zaměření, tedy jeden Elasticsearch server pro oba účely, v tom případě je třeba jednoznačně rozlišit názvy indexů, aby nemohlo dojít ke konfliktu jmen. Uživatelské nastavení objektu konfigurace lze vidět na obrázku 4.1 a je blíže popsáno níže.

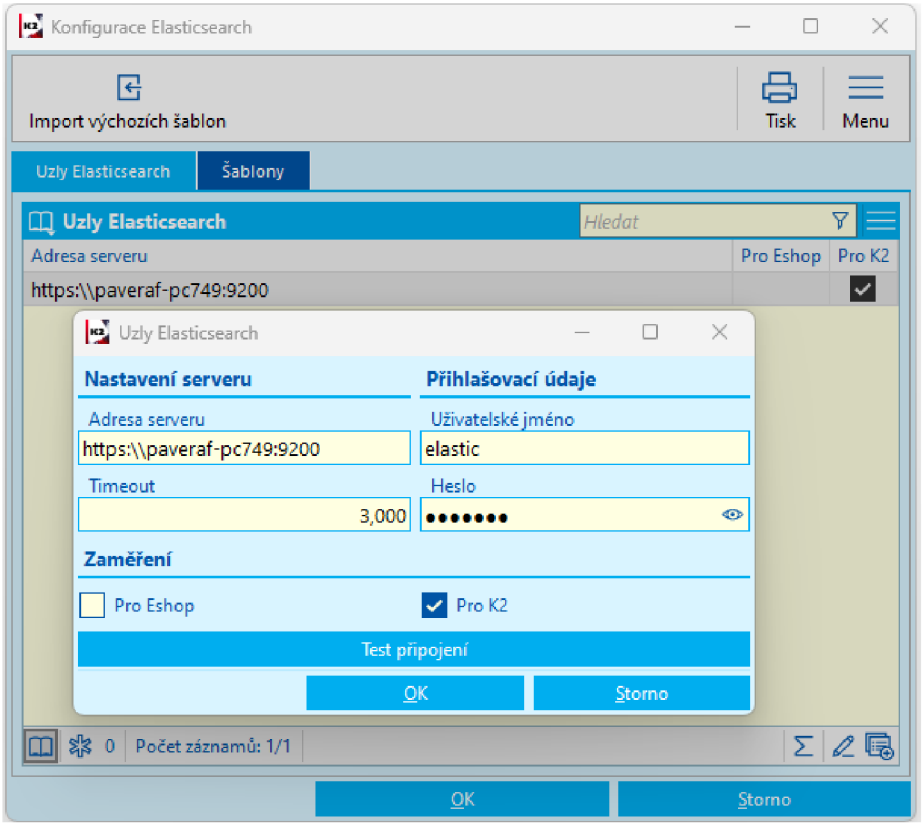
Pro přístup ke globálnímu objektu slouží funkce `GetESConfig`, která načítá objekt buď z databáze, anebo z globální proměnné aktuálně běžící instance K2, pokud již konfigurace byla dříve načtena. Údaje pro komunikaci s Elasticsearch serverem jsou k dispozici skrze pomocné funkce `GetK2Settings`, respektive `GetEshopSettings`.

Šablony indexů jsou reprezentovány objekty třídy `TESIndexTemplate`, v rámci které jsou uloženy jejich definice ve formátu `JSON`. Přes metodu `GetAnalyzers` lze ze šablon vytáhnout seznam analyzátorů, aby je pak bylo možné přiřazovat k jednotlivým polím při tvorbě indexu. Uživatelské rozhraní pro správu šablon lze vidět na obrázku 4.2.

4.1.4 Uživatelské rozhraní

Uživatelské rozhraní pro konfiguraci Elasticsearch je realizováno pomocí modrých formulářů a na ně připojených akcí, viz obrázek 4.1. Skrze formulář, kterým je vizualizován globální objekt konfigurace (`TESConfiguration`), lze vytvářet a modifikovat jak jednotlivá připojení, tak i šablony indexů. Skrze připojené akce jsou zase implementovány ostatní funkce spojené s touto konfigurací.

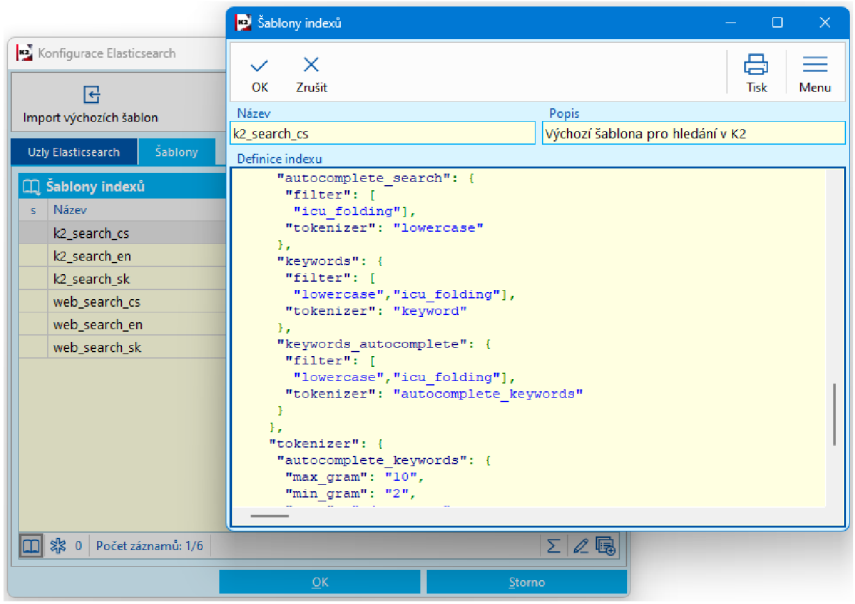
[3]Služba Elasticsearch běží ve výchozím stavu na portu 9200.



Obrázek 4.1: Uživatelské rozhraní pro nastavení potřebných údajů pro komunikaci s Elasticsearch serverem.

Formulář pro správu připojení je členěn do 3 skupin, které sdružují funkčně související parametry. První skupinou je **Nastavení serveru**, kde lze nastavit adresu ke službě Elasticsearch a timeout. Další je skupina **Přihlašovací údaje**, ve které se specifikuje jméno uživatele a jeho heslo pro autentizaci komunikujícího, a poslední skupinou je **Zaměření**, kde se zadává, jestli je toto připojení určeno pro vyhledávání v K2 systému nebo na K2 e-shopu, popřípadě pro obojí najednou. Mimo to je zde i možnost provést **test připojení** pomocí příslušného tlačítka. Akce realizuje zaslání dotazu na Elasticsearch server, jehož odpověď poté zobrazí uživateli. Z této odpovědi je pak možné vyhodnotit, jestli se podařilo připojit nebo ne.

Na záložce Šablony, viz obrázek 4.2, je uživateli umožněno vytvářet, modifikovat či odstraňovat šablony pro vytváření indexů. V rámci rozhraní lze editovat jejich názvy, popisy, anebo přímo definice ve formátu JSON. Se šablonami se pojí další akce, kterou lze z této konfigurace vyvolat, a tou je **import výchozích šablon**. Výchozí šablony jsou k dispozici v adresářové struktuře programu K2 v XML souborech, které obsahují deserializovaný objekt jazyka Delphi. Ten je následně transformován do objektu třídy `TESIndexTemplate`, jenž je v rámci příslušného formuláře možné dále spravovat.



Obrázek 4.2: Uživatelské rozhraní pro správu šablon Elasticsearch indexů. Zobrazená část šablony obsahuje definice analyzátorů.

4.2 Elasticsearch indexy

Další řešenou oblastí je správa indexů a obecně způsob, kterým se přistupuje k indexování potřebných dat. V této sekci jsou postupně rozepsány všechny záležitosti této problematiky.

4.2.1 Index jako datový modul

V prvé řadě bylo třeba vyřešit způsob indexování dat. Pro vyřešení tohoto problému je vhodné specifikovat konkrétní požadavky a případy využití vyhledávače Elasticsearch v systému K2. Na tomto místě je vhodné zmínit, že cílem integrace Elasticsearch není nahradit stávající způsob uložení a správy dat, ale zefektivnit přístup k nim v rámci systému K2. Stejně tak nemá integrace za cíl sloužit jako redundantní úložiště dat.

Datový modul může obsahovat tisíce záznamů, které se skládají jak z fyzických polí, tak z počítaných polí, které vznikají na míru jednotlivým klientům. Právě tato pole, která nejsou fyzicky uložená v databázi, vedla v systému často k až nepřijatelně dlouhému načítání přednastavených filtrů, či znemožnila v rozumném čase dohledat konkrétní záznam nebo záznamy, se kterými uživatel potřeboval zrovna pracovat. Prvním požadavkem na tuto integraci tedy bylo umožnit **indexování a vyhledávání v libovolném poli**.[4]

Každý zákazník K2 je zvyklý na jiný způsob práce se svými daty. Často se objevují požadavky na velice specifické filtry dat, se kterými potřebuje zákazník provádět určité úkony. Jedná se například o filtraci přes několik úrovní vazeb, či specifické nastavení priorit jednotlivých filtrovacích podmínek. V každém případě se naráží na limity stávajících možností filtrování, které sice je možné specifikovat i přímo SQL dotazem, ale vytvoření takového dotazu už vyžaduje určitou úroveň expertízy a pro běžného pracovníka ERP systému se

[4] Avšak stále s jistými omezeními, jak vyplyne dále v textu.

jedná o nereálný úkol. Nová technologie s sebou nese i nové možnosti filtrování a dalším požadavkem tedy je **vytvoření uživatelsky přívětivého rozhraní pro vytváření Elasticsearch filtrovacích dotazů**.

V neposlední řadě je také vyžadována snadná škálovatelnost a rozšiřitelnost této integrace v případě potenciálně dalších možností využití vyhledávače i na jiných místech systému.

Jako nejsmysluplnější a nejintuitivnější varianta indexace dat se jeví kopírovat již existující organizační strukturu dat, na úrovni datových modulů. Práce s daty na této úrovni umožní naplnit veškeré požadavky na integraci Elasticsearch a mimo to také respektuje základní princip fungování systému, kdy naprostá většina funkcionality je provázána právě s jednotlivými datovými moduly. Tedy jeden Elasticsearch index bude zahrnovat data právě jednoho datového modulu.

4.2.2 Pole indexu

Při indexování dat na úrovni datového modulu se přirozeně nabízí mapovat jednotlivá pole indexu na pole datového modulu. Jak již ale bylo výše zmíněno, není cílem redundantně indexovat data, ale indexovat pouze ta pole, která mají pro účely vyhledávání a filtrování v daném datovém modulu smysl. Jedná se tedy o běžně používaná pole, jako jsou zkratka, název, identifikační číslo apod., a mimo to také o počítaná pole, které často vznikají pro specifické potřeby zákazníků a jejichž vyhodnocení zdržuje vyhodnocení filtru.

Při vytváření mapování polí indexu se nastavují také analyzátory, díky kterým lze indexovaným datům přidat další vlastnosti. Lze použít pouze analyzátory, které jsou specifikovány v definici indexu (viz kapitola 4.1.2). Každé pole datového modulu může obsahovat různé hodnoty a je pro ně vhodný jiný typ analyzátoru. Na pole s kódovou zkratkou například není vhodné použít analyzátor, který text rozděluje podle pomlček nebo mezer apod.

Vybírat do indexu lze libovolné textové pole, případně pole, jejichž hodnoty lze na textové řetězce převést. Vazební pole, která se odkazují do jiného datového modulu, je také možné indexovat. Elasticsearch však oproti relačním databázím není uzpůsobený na práci s cizími klíči a proto je v této integraci třeba hodnoty vazebních polí indexovat přímo. S tím je pak spojena nutnost pravidelné synchronizace a aktualizace dat indexu.

4.2.3 Synchronizace dat

Vzhledem k tomu, že data v ERP systému vznikají, jsou modifikována či odstraňována, je třeba je pravidelně synchronizovat vůči datům na Elasticsearch serveru. Nebylo by úplně efektivní synchronizovat vždy po každé změně v datovém modulu, ale podle vytíženosti datového modulu provádět synchronizaci v předem stanovených pravidelných intervalech. Systém K2 umožňuje plánovat pravidelné úlohy prostřednictvím tzv. *plánovače úloh*, který běží na pozadí v rámci aplikačního serveru, tedy je v provozu i pokud neběží klient K2. Tento plánovač lze tedy vhodně využít právě pro účely pravidelné synchronizace.

Naivní způsob synchronizace by mohl fungovat tak, že se vždy smažou všechna data indexu a poté se datový modul indexuje celý znova. Tento způsob by však byl pro rozsáhlé datové moduly časově náročný. Lze však využít vlastnosti většiny datových modulů, které obsahují pole s časovým razítkem identifikujícím poslední provedenou změnu. Tak lze jednoduše identifikovat ty záznamy, které prošly od poslední synchronizace změnou, a aktualizovat

index pouze na této množině záznamů. Vyžaduje to pouze uložení časového razítka poslední synchronizace, ale také to omezuje indexování dat pouze těch datových modulů, jež toto pole obsahují[5]. Toto pole však obsahuje většina běžně používaných datových modulů.

Pro sledování změn ve vazebních polích je třeba kromě pole s časovým razítkem daného datového modulu sledovat tato pole i ve všech provázaných modulech. Pro zjednodušení a vyhnutí se nevyžádané rekurze při kontrole vazeb je umožněno pracovat s vazebními poli pouze do hloubky úrovně jedna. Toto omezení by však mohlo být do budoucna posunuto i do hlubší úrovně zanoření vazeb.

4.2.4 Práva a přístup k datům

Další nezbytným úkonem je zajistit přístup k datům pouze osobám, jež jsou k tomu oprávněny. V systému K2 již existuje zaběhnutý způsob, kterým se tato problematika řeší. Konkrétně se jedná o hierarchickou strukturu rolí a k nim příslušných práv, které jsou přiřazeny jednotlivým uživatelům podle potřeby, viz kapitola 2.2.6. Přístup k datům se rozlišuje na vícero úrovních. Například u datových modulů typu *doklad* se zohledňují práva podle skupiny dokladu, knihy dokladu, vlastní firmy, dodavatele/odběratele a také nadřízeného dokladu. Elasticsearch sice umožňuje jednotlivým uživatelům služby nastavovat určité role práv, avšak toto řešení je z pohledu tohoto projektu hodně komplikované a vedlo by k nepřiměřenému ztížení konfigurace. Proto se rozhodlo využít existujících polí datových modulů, která indikují příslušná přístupová práva, a nahrávají se společně s ostatními daty na Elasticsearch server. Při vytváření vyhledávacího dotazu jsou pak vyhodnocena práva dotazujícího se uživatele a jsou do dotazu zakomponována. Výsledky tak obsahují pouze ty záznamy, na které má uživatel oprávnění.

Nutnost řešit přístupová práva je potřeba pro účely funkce *globálního vyhledávání*, pro které je specificky implementováno i zobrazení nalezených výsledků[6]. Při filtrování v datových modulech si řeší zobrazování dat stávající systém filtrů standardně podle práv daného uživatele, tedy pro tuto část není třeba tento problém řešit.

4.2.5 Vyhledávání na Elasticsearch

Vyhledávání pomocí technologie Elasticsearch je realizováno pomocí vyhledávacího dotazu, viz 3.1.4. Jedná se přímo o dotazovací jazyk[7], který je specifikován v oficiální dokumentaci Elasticsearch. Jak již bylo několikrát zmíněno, snahou tohoto projektu je vytvoření uživatelsky přívětivého rozhraní za účelem snadného využití možností technologie Elasticsearch. Bylo tedy třeba vytvořit abstrakci dotazovacího jazyka do takové míry, aby i nezkušený uživatel byl schopen vytvořit vyhledávací dotaz podle svých potřeb.

Nejdůležitějšími parametry vyhledávacího dotazu jsou pole, ve kterých má hledání probíhat. Tato pole lze v dotazu specifikovat více způsoby. Jako nejvhodnější způsob se jeví použití tzv. *bool dotazu*, který je složený z více vnořených dotazů, čímž je možné realizovat hledání v jednotlivých polích indexu. Tímto způsobem je navíc možné konfigurovat dotaz na úrovni jednotlivých polí zvlášť. Pro každé pole je umožněno nastavovat prioritu shody v daném

[5]Toto pole by však v případě potřeby bylo možné do datového moduly přidat.

[6]Funkce globálního vyhledávání je povolená pouze na webovém klientovi a zobrazování výsledků není součástí této práce a je řešeno jiným vývojářem společnosti K2.

[7]Query DSL – Domain Specific Language

poli vůči ostatním polí, zda se mají v daném poli tolerovat překlepy a také pro něj lze specifikovat analyzátor, podle kterého je na serveru pole zpracováno a indexováno. Na úrovni celého dotazu je možné nastavit například limit na počet nalezených záznamů.

Abstrakcí při sestavování dotazu ale není možné pokrýt veškeré možnosti technologie Elasticsearch. Toto by již bylo nad rámec projektu. Aby však bylo možné využívat plné síly dotazovacího jazyka, je zde prostor definovat si dotaz ručně přímo ve formátu JSON. Vzhledem k tomu, že manuální vytvoření dotazu je komplexní operací a může vést k chybovosti, je tato možnost omezena pouze pro uživatele, kteří mají s touto technologií vlastní zkušenosti a je to ponecháno pouze na jejich vlastní uvážení.

4.2.6 Správa indexů

Dále bylo třeba vymyslet rozhraní pro správu indexů v rámci systému K2. Samotnou technologii Elasticsearch je možné konfigurovat velkým množstvím způsobů a pro efektivní využití jejích možností je třeba už určité úrovně expertízy, na kterou běžný uživatel ERP systému nemá možnost dosáhnout. Cílem tedy je vytvořit uživatelsky přívětivé rozhraní, které realizuje nejen běžnou správu služby Elasticsearch, ale také umožňuje nastavení vyhledávacích dotazů tak, aby to bylo pro nezkušené uživatele snadno pochopitelné. Mezi úkony tohoto rozhraní má patřit:

- vytváření/modifikace/odstraňování indexů.
- import dat a synchronizace,
- definice mapování jednotlivých polí indexu a jejich analyzátorů,
- definice vyhledávacích dotazů,
- nastavení funkce globálního vyhledávání a
- výchozí nastavení konfigurace indexů.

Jednotlivé úkony jsou popsány níže při popisu implementace v sekci 4.2.8.

4.2.7 Globální vyhledávání

Jedna z nově vzniklých funkcí postavených na technologii Elasticsearch je *globální vyhledávání napříč „všemi“ datovými moduly K2*. Jedná se o funkci, která má fungovat jako exkluzivní způsob vyhledávání v rámci webového klienta K2. Vyhledávání napříč systémem má probíhat podle následujícího popisu: datové moduly jsou rozděleny do tří skupin na doklady, číselníky (ostatní) a menu[8]. Každá skupina má předem určená pole, ve kterých lze vyhledávat. Těmi jsou pro jednotlivé skupiny:

- pro skupinu doklady – číslo dokladu, dodavatel/odběratel, popis, sloupec 1 a sloupec 2,
- pro skupinu číselníky – zkratka, název, sloupec 1, sloupec 2 a sloupec 3,

[8]Technicky se nejedná o skupinu, ale o samostatný datový modul `TMainMenu`.

- pro skupinu menu – název, popis, ikona, cesta a sloupec 1.

Pole `sloupec` následované číslem je určené pro mapování libovolného dalšího pole z datového modulu příslušné skupiny a jejich počet je stanoven tak, aby každá skupina měla celkem pět polí určených pro funkci tohoto vyhledávání. Ve všech těchto polích se nejenom vyhledává, ale zároveň jsou to pole, která se zobrazí v rámci výsledků hledání, proto je jejich počet pevně stanoven, aby šlo zobrazení výsledných hodnot vhodně naformátovat. Kromě těchto polí je součástí výsledku také identifikátor záznamu, aby bylo možné se „proklikem“ dostat přímo do datového modulu na detail záznamu.

Pro hledání se využije vlastnosti vyhledávače Elasticsearch, kdy je možné jedním dotazem adresovat více indexů, a tím vyhledávat ve více datových modulech najednou.

4.2.8 Implementace logiky

Hlavním objektem zastřešujícím veškerou konfiguraci Elasticsearch indexů je objekt třídy `TElasticsearchSettings`. Úkolem tohoto objektu je odrážet aktuální stav indexů na serveru a zprostředkovat jejich správu na úrovni systému K2. Tento objekt udržuje kolekci aktuálních indexů `CurrentIndexes`, kterou je možné prostřednictvím uživatelského rozhraní rozšiřovat nebo zmenšovat.

Index je reprezentován třídou `TESIndex`. Zde jsou specifikovány všechny potřebné hodnoty, od identifikace datového modulu, který je indexován, přes mapování polí modulu na pole indexu, až po definice vyhledávacích dotazů, které jsou nad daným indexem prováděny. Ideálním stavem je konzistence mezi nastavením indexů v systému K2 a na Elasticsearch serveru. Proces definice nového indexu vypadá následovně:

1. vytvoření nového indexu v rámci konfigurace Elasticsearch indexů v systému K2,
2. nastavení parametrů nově vytvořeného indexu (mapování polí, analyzátorů apod.),
3. vytvoření indexu na Elasticsearch serveru (pomocí akce v rámci uživatelského rozhraní),
4. úplný import dat do indexu,
5. nastavení pravidelné synchronizace dat,
6. kontrola konzistence polí datového modulu a indexu.[9]

Pomocí příznaku *existence indexu na Elasticsearch serveru* lze pak zjistit, jestli operace *vytvoření* proběhla v pořádku, případně jestli nedošlo k nějaké serverové chybě.

Mezi operace, které je třeba provádět na Elasticsearch serveru, patří vytvoření/odstranění indexu, synchronizace dat a případně také změna mapování polí. K těmto úkonům se využívá REST rozhraní služby. Změna mapování polí má však to omezení, že jde pouze přidávat nová mapování, ale nelze je odebírat nebo modifikovat, jelikož by tak došlo k narušení konzistence mezi již indexovanými daty a nastavením indexu. K narušení konzistence dochází

[9] Nutné pouze po aktualizaci systému K2 na vyšší verzi, ve které může dojít ke změnám vnitřní struktury datových modulů.

sice i při přidávání nových mapování polí, ale to lze vyřešit následným provedením úplného importu dat.

Základní třídou pro definici vyhledávacích dotazů je `TESSearchBase` a její potomci `TESSearchRaw`, `TESSearchConfigurable` a `TESSearchGlobal`. Jedná se o abstraktní třídu, která definuje společné vlastnosti a metody pro její potomky. Deklaruje abstraktní metody `InternalGetJSON` a `InternalSetJSON`, kterými její konkrétní potomci musí implementovat sestavení výsledného JSON dotazu. Třída `TESSearchRaw` reprezentuje třídu, která definici dotazu spravuje přímo v textové podobě a vrací dotaz tak, jak jej uživatel sepsal. Oproti tomu třída `TESSearchConfigurable` sestavuje dotaz z jednotlivých parametrů, které uživatel nastaví v příslušném formuláři. Pro složení dotazu využívá šablonu, viz kód 4.1.

```
{
  "size": MATCH_LIMIT,
    "query": {
    "bool": {
      "should": [
        {
          "multi_match": {
            "query": "{{QueryText}}",
            "fields": [ LIST_OF_SEARCH_FIELDS ],
            "operator": OPERATOR,
            "fuzziness": FUZZINESS,
            "prefix_length": PREFIX_LENGTH,
            "boost": BOOST
          }
        },
        ...
      ],
      "minimum_should_match": MINIMUM_SHOULD_MATCH
    }
  }
}
```

Výpis 4.1: Šablona dotazu pro konfigurovatelné vyhledávání.

Výrazy označené kapitálkami znázorňují v této šabloně konfigurovatelné hodnoty skrze uživatelské rozhraní. `MATCH_LIMIT` omezuje počet nalezených shod. `LIST_OF_SEARCH_FIELDS` je seznam polí, respektive analyzátorů daného pole, které jsou předmětem dotazu. `OPERATOR` určuje, zda musí být shoda nalezená ve všech polích, nebo alespoň v jednom. `FUZZINESS` specifikuje míru tolerance překlepů a `PREFIX_LENGTH` zase, kolik znaků se na začátku porovnávaných výrazů musí vždy shodovat. `BOOST` slouží k možnosti zvýhodnit shodu v daných polích oproti ostatním polím. Hodnota `{{QueryText}}` je v šabloně pevně daná a do ní je v okamžiku zaslání dotazu na Elasticsearch server doplněn vyhledávaný výraz. Objekt `multi_match` reprezentuje nastavení vyhledávání pro jedno pole, resp. pro vybrané analyzátory daného pole. Tento se tedy v rámci objektu `should` objevuje tolikrát, kolik polí je pro dotaz vybraných. Nastavením příznaku `ProMode` lze i v této konfigurovatelné třídě specifikovat dotaz přímo pomocí JSON kódu. Uživatelské rozhraní konfigurace lze vidět na obrázku 4.6.

Třída `TESSearchGlobal` je určená pro definici dotazu pro funkci globálního vyhledávání. Podobně jako v předchozí konfigurovatelné třídě je i zde možnost prostřednictvím formuláře nastavovat jednotlivé parametry dotazu a v případě potřeby také napsat dotaz přímo ve formátu JSON. Parametry a samotná šablona jsou však pro tento dotaz jiné, viz 4.2. Pro každou skupinu globálního vyhledávání vzniká jeden dotaz, který zahrnuje všech pět polí

skupiny. Jelikož názvy polí datových modulů se nemusí vždy shodovat, i když reprezentují ve skupině pole stejného významu, využívá se v Elasticsearch indexu pomocných polí, které drží kopie jiných polí. Při definici indexu se definují navíc pole pro globální vyhledávání (s předem danými názvy), která reprezentují pole globálního hledání, a tato pole jsou použita v šabloně dotazu.

```
{
  "script_fields": {
    SCRIPT_FIELD: {
      "script": {
        "source": "$('ES_SCRIPT_FIELD','')",
        "lang": "painless"
      }
    },
    ...
  },
  "_source": ["Id","RightIdBrowse","RightGroupIdValue"],
  "size": MATCH_LIMIT,
  "query": {
    "bool": {
      "should": [
        {
          "multi_match": {
            "fields": [ LIST_OF_SEARCH_FIELDS],
            "prefix_length": PREFIX_LENGTH,
            "operator": OPERATOR,
            "fuzziness": FUZZINESS,
            "boost": BOOST,
            "query": "{{QueryText}}"
          }
        },
        ...
      ],
      "filter": [
        {
          "terms": {
            RIGHT_FIELD: [RIGHT_FIELD_VALUES]
          }
        },
        ...,{
          "bool": {
            "should": [
              {
                "terms": {
                  "CompanyList": [COMPANY_VALUES]
                }
              },{
                "bool": {
                  "must_not": {
                    "exists": { "field": "CompanyList" }
                  }
                }
              }
            ]
          }
        }
      ],
      "minimum_should_match": MINIMUM_SHOULD_MATCH
    }
```

```
  },
  "indices_boost": [ INDICES_BOOST ]
}
```

Výpis 4.2: Šablona dotazu pro globální vyhledávání.

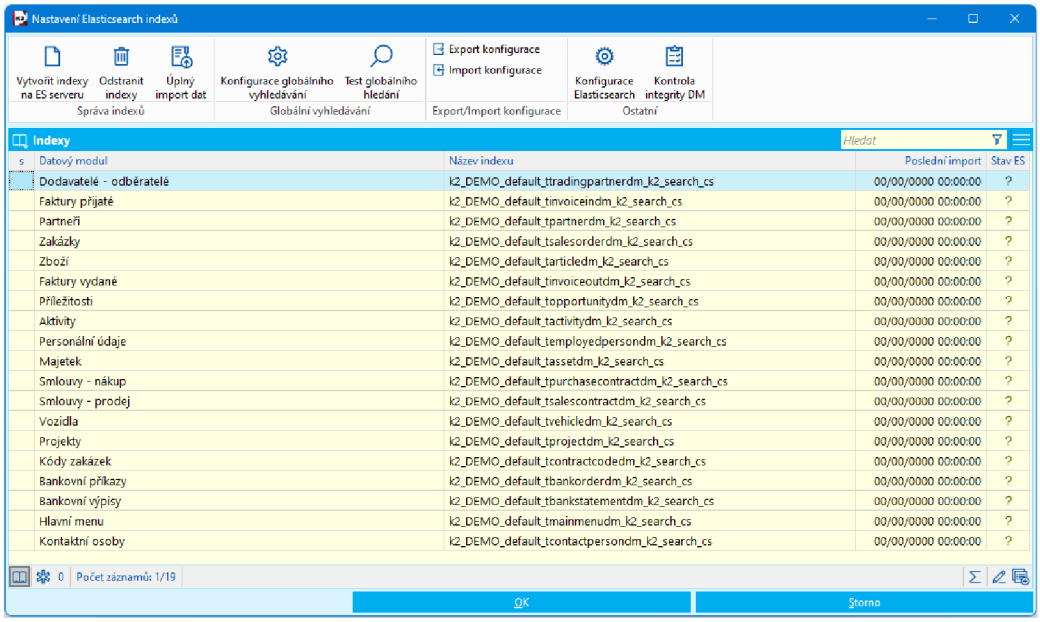
První část dotazu obsahuje tzv. skriptová pole. Hodnota `SCRIPT_FIELD` reprezentuje název pole v rámci objektu třídy `TESSearchResult`, do kterého je výsledek dotazu transformován, a hodnota `ES_SCRIPT_FIELD` je název pomocného pole pro globální vyhledávání v rámci Elasticsearch indexu. Dále je v dotazu specifikován filtr s poli pro přístupová práva a vlastní firmy, které je nutné vyhodnocovat pro funkci globálního vyhledávání, jak již bylo napsáno dříve. `RIGHT_FIELD` reprezentuje název příslušného pole práva a `RIGHT_FIELD_VALUES` pak seznam jemu odpovídajících hodnot. `COMPANY_VALUES` je zase seznam vlastních firem, do kterých má uživatel přístup. Využití mechanismu Elasticsearch filtru v dotazu má výhodu toho, že tento filtr nemá vliv na skóre nalezeného dokumentu a navíc jej služba umí kešovat. Posledním parametrem šablony je `INDICES_BOOST`, kterým se nastavuje priorita jednotlivých indexů skupiny. Zbylá část šablony a jejich parametrů je shodná s předchozí šablonou. V případě vyhledávání ve skupině menu obsahuje objekt `bool` ještě navíc položku `"must": "term": "IsFolder": false`, která vynucuje vyhledávání pouze v listových položkách stromové struktury záznamů datového modulu `TMainMenu`.

Pro úplný import dat a synchronizace je klíčová funkce `ProcessDMData`. Ta realizuje načtení potřebných dat z datového modulu, podle definice mapování polí daného indexu. Mimo tato pole se v rámci této funkce získávají i potřebná pole pro realizaci kontroly přístupových práv. Načtená data jsou následně formátována do požadovaného JSON dotazu, který je skrze REST rozhraní po dávkách (pomocí *bulk* operací) poslán na Elasticsearch server. Úplný import dat realizuje akce, která je spustitelná z rozhraní konfigurace indexů, a synchronizace dat je pak akce spustitelná z rozhraní detailu indexu. Jak již bylo uvedeno výše, synchronizaci je také možné naplánovat, aby probíhala v pravidelných intervalech. Toto realizuje třída `TESIndexAutoUpdater`, která obsahuje potřebné metody pro získání změn v daném datovém modulu a následné volání funkce `ProcessDMData`. Pravidelnou synchronizaci pak realizuje třída `TBShortcutElasticsearchSynchronize`, která reprezentuje v systému K2 *dávku*, již je možné naplánovat v rámci *plánovače úloh*. V této třídě je možné nastavit, zdali se má provádět synchronizace pro jeden konkrétní index, nebo pro všechny indexy. Tímto lze pro různé datové moduly nastavit různou frekvenci synchronizace podle konkrétních potřeb dané organizace.

Za účelem efektivního přístupu k *Nastavení Elasticsearch indexů* je objekt, který ho reprezentuje, při prvním přístupu načten z databáze do globální proměnné, která existuje v rámci aktuálně běžící instance programu K2. V případě provedení změn v tomto nastavení je tak třeba obeznámit všechny běžící instance, že je nutné objekt znovu načíst, aby uživatelé pracovali s aktuálními informacemi. V K2 se pro tento účel využívá tzv. mechanismu `synccache`. Princip tohoto mechanismu je následující: ve sdílené tabulce se uchovává informace o časovém razítku poslední změny, respektive posledního uložení objektu. Každá běžící instance si pak při přístupu k objektu vytáhne z tabulky tuto hodnotu a uloží si ji k dané instanci objektu. Při každém dalším přístupu se pak kontroluje, zdali v tabulce není novější časová hodnota, a pokud ano, tak se objekt znovu načte z databáze.

4.2.9 Uživatelské rozhraní

Implementace uživatelského rozhraní opět využívá modrých formulářů. Hlavním formulářem je *Nastavení Elasticsearch indexů*, který vizualizuje třídu `TElasticsearchSettings`. Tento formulář je zobrazený na obrázku 4.3. V tzv. *gridu*[10] je k dispozici seznam aktuálních indexů. Každý záznam obsahuje informace o datovém modulu daného indexu, názvu indexu na Elasticsearch serveru a časovém razítku posledního importu. Grid také obsahuje sloupec *Stav ES*, který pomocí ikony znázorňuje stav daného indexu v kontextu Elasticsearch serveru. Tento sloupec může nabývat tří hodnot: `Neznámý`, `Existuje` a `Neexistuje`, které po řadě znamenají, že stav indexu nelze ověřit, že index existuje na Elasticsearch serveru a že index na daném serveru nelze nalézt, nebo došlo k nějaké jiné chybě.



Obrázek 4.3: Formulář nastavení Elasticsearch indexů.

Na liště v horní části formuláře jsou k dispozici akce, které lze provádět z této konfigurace. První tři akce realizují operace *vytvoření*, *odstranění* a *úplný import dat* indexu v rámci Elasticsearch serveru. Jedná se o nevratné operace, které modifikují stav serveru a uživatel je po jejich provedení nucen uložit stav formuláře, respektive jemu příslušného objektu, aby nedošlo k narušení konzistence mezi systémem K2 a Elasticsearch serverem. Tyto operace lze provádět buď pro jeden záznam, který je v gridu podbarvený modrou barvou, nebo pro všechny označené záznamy.[11]

Další akce souvisí s funkcí *globálního vyhledávání*. *Konfigurace globálního vyhledávání* otevře formulář, v rámci kterého se definují dotazy pro prohledávané skupiny datových modulů. Akce *Test globálního hledání* pak slouží k ověření provedeného nastavení. Probíhá zde sestavení výsledného dotazu pro všechny skupiny a jeho zaslání na koncový bod Elasticsearch API `_msearch`. Odpověď serveru je následně zobrazena jak ve formě JSON odpovědi, tak ve formě záznamů v gridu formuláře.

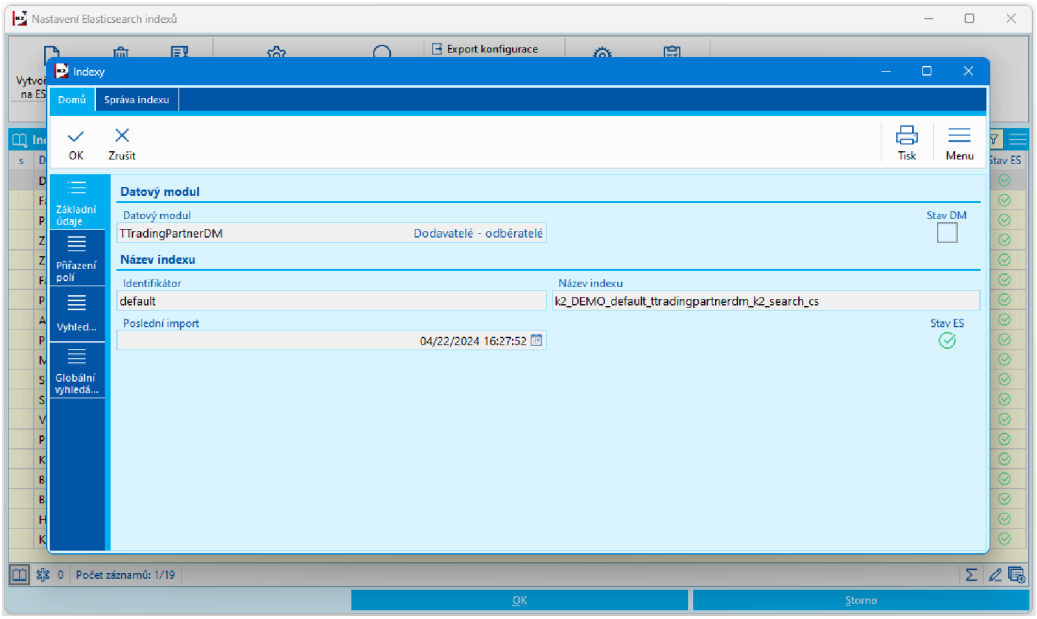
[10] V terminologii systému K2 se výrazem *grid* rozumí tabulka záznamů datového modulu či objektu, viz kapitola 2.2.7.

[11] Označování záznamů je záležitostí standardní funkcionality *gridu*.

Dále jsou zde funkce pro export a import konfigurace do souboru. Tyto funkce slouží pro ukládaní aktuální konfigurace, případně načtení již dříve vytvořené konfigurace. V rámci těchto operací dochází k serializaci, resp. deserializaci objektu konfigurace do XML souboru. Díky těmto funkcím lze společně s programem K2 distribuovat také výchozí nastavení indexů pro jednodušší zavedení vyhledávače Elasticsearch u zákazníků.

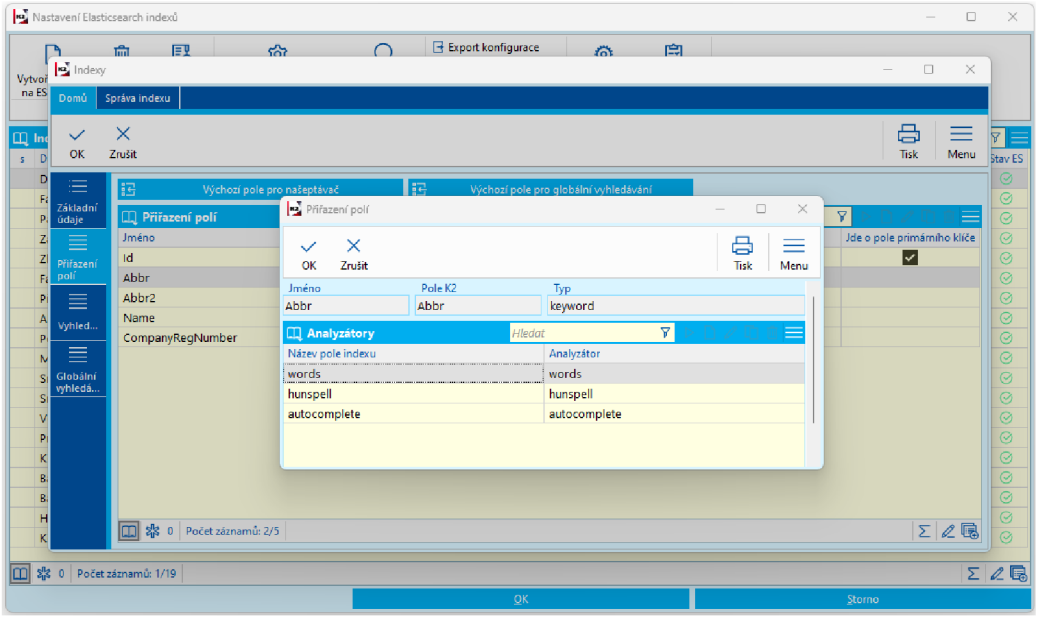
V poslední skupině se nachází akce pro vyvolání formuláře s *Elasticsearch konfigurací* a akce pro kontrolu integrity datových modulů. První akce slouží pouze jako propojení do obecné konfigurace služby, zatímco druhá slouží pro kontrolu struktur datových modulů po reinstalaci systému K2.

„Rozkliknutím“ záznamu v seznamu indexů se otevře formulář s detailem příslušného indexu, viz obrázek 4.4. Ten se skládá z několika záložek, které seskupují jednotlivá nastavení indexu. Na záložce *Základní údaje* jsou informace ohledně příslušného datového modulu a ohledně názvu indexu. Název indexu se skládá z názvu mandanta K2, identifikátoru, datového modulu a šablony použité při vytváření indexu, a to proto, aby byly indexy existující v rámci jedné instance Elasticsearch služby snadno rozlišitelné a nedocházelo k nevyžádanému vzniku dvou indexů se stejným názvem. Uživatel má možnost ovlivnit pouze hodnotu identifikátoru indexu. Kromě toho lze na této záložce také najít informaci o stavu indexu na Elasticsearch serveru a jeho stav po kontrole integrity příslušného datového modulu.



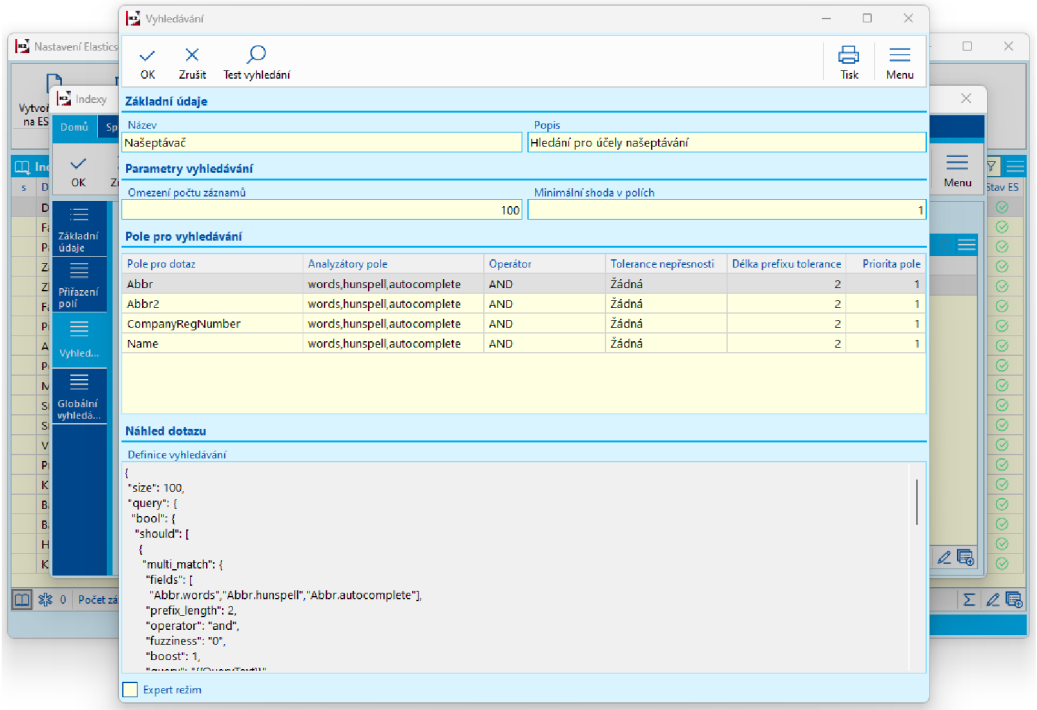
Obrázek 4.4: Detail konfigurace Elasticsearch indexu.

Na záložce *Přiřazení polí* se pak specifikuje, která pole datového modulu a podle jakých analyzátorů chce uživatel indexovat. Na obrázku 4.5 lze vidět detail pole s názvem `Abbr` a k němu přiřazené analyzátory `words`, `hunspell` a `autocomplete`. Výběr pole z datového modulu je realizován skrze již existující formulář systému K2, který umí načíst a zobrazit všechna pole datového modulu. Je zde také možnost automaticky načíst přednastavená pole z datového modulu, která jsou vhodná pro funkci našeptávače a globálního vyhledání. K tomu slouží tlačítka *Výchozí pole pro našeptávač* a *Výchozí pole pro globální vyhledávání*.



Obrázek 4.5: Formulář mapování polí indexu.

Záložka *Vyhledávání* slouží k definici hledacích dotazů. Detail konfigurovatelné definice dotazu pro našeptávač je znázorněný na obrázku 4.6. Skrze toto rozhraní lze vybírat pole pro daný dotaz a nastavovat parametry jak celého dotazu, tak jednotlivých polí. Pro každé



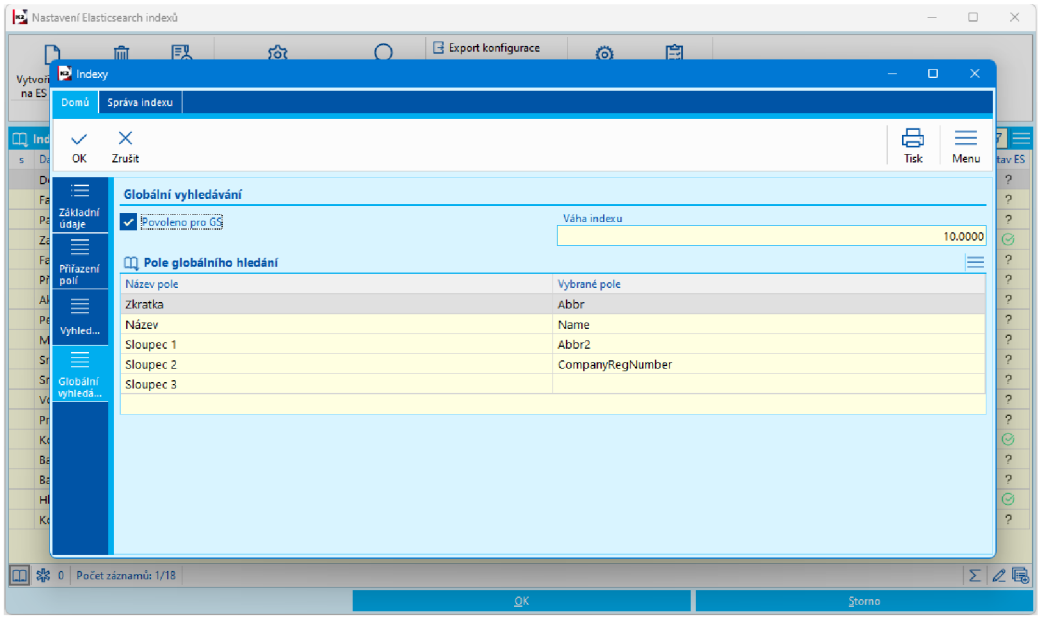
Obrázek 4.6: Formulář vyhledávacího dotazu Elasticsearch.

pole lze vybrat jemu příslušné analyzátory, které k němu byly přiřazeny při indexování,

operátor, který určuje, zda musí být shoda ve všech analyzátorech daného pole či alespoň v jednom, toleranci nepřesnosti, díky které je možné najít shody i případě překlepů v hledaném výrazu, délku prefixu tolerance, která specifikuje minimální počet znaků na začátku výrazu, který se musí shodovat přesně, a nakonec prioritu pole, jež ovlivňuje skóre nalezené shody, podle daného pole. K dispozici je také náhled výsledného dotazu ve formátu JSON, který se dynamicky mění podle aktuálně nastavených parametrů. Zaškrtnutím příznaku `Expertní režim` dojde k přepnutí formuláře do režimu přímého zadávání JSON dotazu. Skrze tento režim lze sestavit libovolný hledací dotaz, neprobíhá však žádná kontrola jeho správnosti. Lze však tímto způsobem dodatečně parametrizovat i jeho další části. Hodnoty, které chce uživatel specifikovat dodatečně, například až před samotným vyhledáváním, je třeba ohraničit dvěma složenými závorkami z obou stran, například `{{QueryText}}`, což je zároveň jediný povinný parametr, který je následně nahrazován hledaným výrazem.

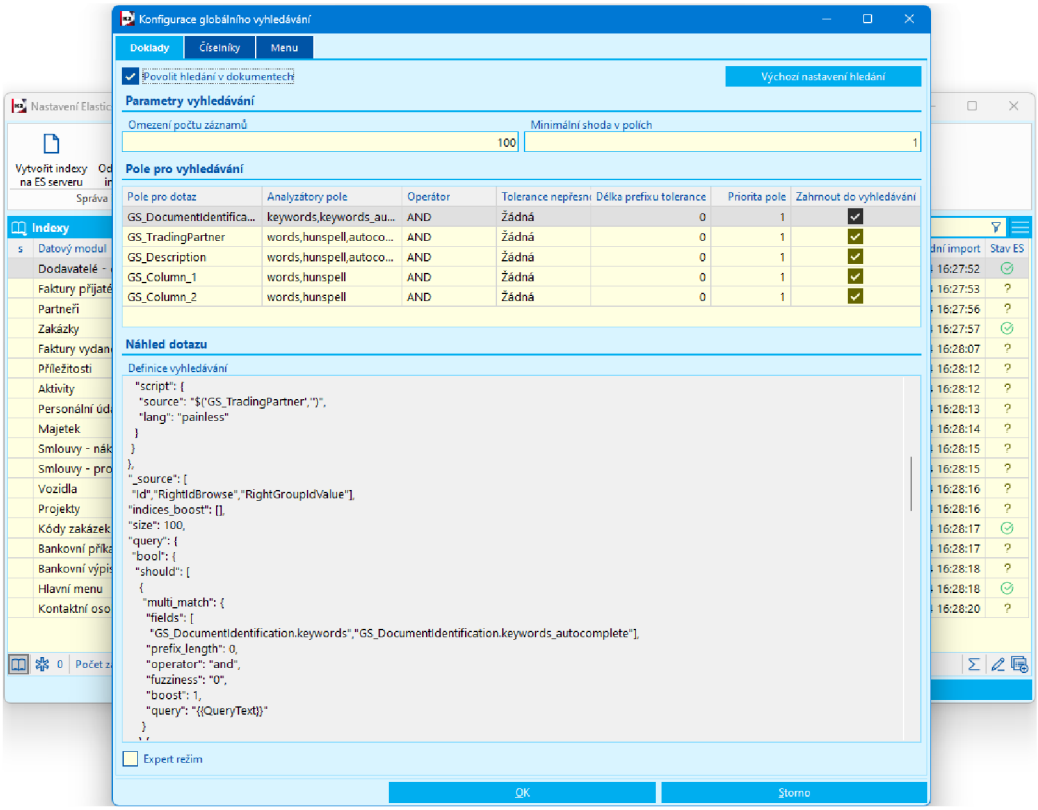
Součástí rozhraní je také akce pro otestování vytvořeného dotazu. Tato akce volá vyhledávací dotaz na Elasticsearch serveru a zobrazí uživateli odpověď, která obsahuje buď nalezené dokumenty, nebo v případě nesprávně definovaného dotazu příslušnou chybovou hlášku.

Na záložce *Globální vyhledávání*, která je na obrázku 4.7, se provádí nastavení indexu spojené s funkcí *globálního vyhledávání*. Příznak `Povoleno pro GS` určuje, zda index má být součástí vyhledávání. Váha indexu stanovuje prioritu daného indexu oproti ostatním ve skupině. V tabulce *Pole globálního hledání* je možné vybírat z příslušného datového modulu pole, ve kterých má hledání probíhat.



Obrázek 4.7: Nastavení indexu pro funkci globálního vyhledávání.

Další nastavení funkce *globálního vyhledávání* lze zpřístupnit přes akci *Konfigurace globálního vyhledávání* z formuláře *Nastavení Elasticsearch indexů*. Formulář nabízí stejné rozhraní pro definici dotazu jako při editaci vyhledávání v detailu indexu. Mimo to je zde možné povolovat vyhledávání v jednotlivých skupinách, anebo využít akce *Výchozí nastavení hledání*. Rozhraní tohoto formuláře je znázorněno na obrázku 4.8.



Obrázek 4.8: Formulář s nastavením dotazu pro globální vyhledávání.

4.3 Filtrovací podmínka

Konkrétním využitím integrace vyhledávače Elasticsearch do systému K2 je nová filtrovací podmínka, kterou lze použít při vytváření filtru v datovém modulu. Cílem bylo zakomponovat tuto novou podmínku do stávajícího systému tak, aby navazovala na již hotové řešení vytváření filtrů a aby ji bylo možné kombinovat s již existujícími podmínkami. Stávající systém načítá data do filtrů přímo z SQL serveru, proto bylo třeba vymyslet způsob, kterým by se propojila služba Elasticsearch s databázovým serverem.

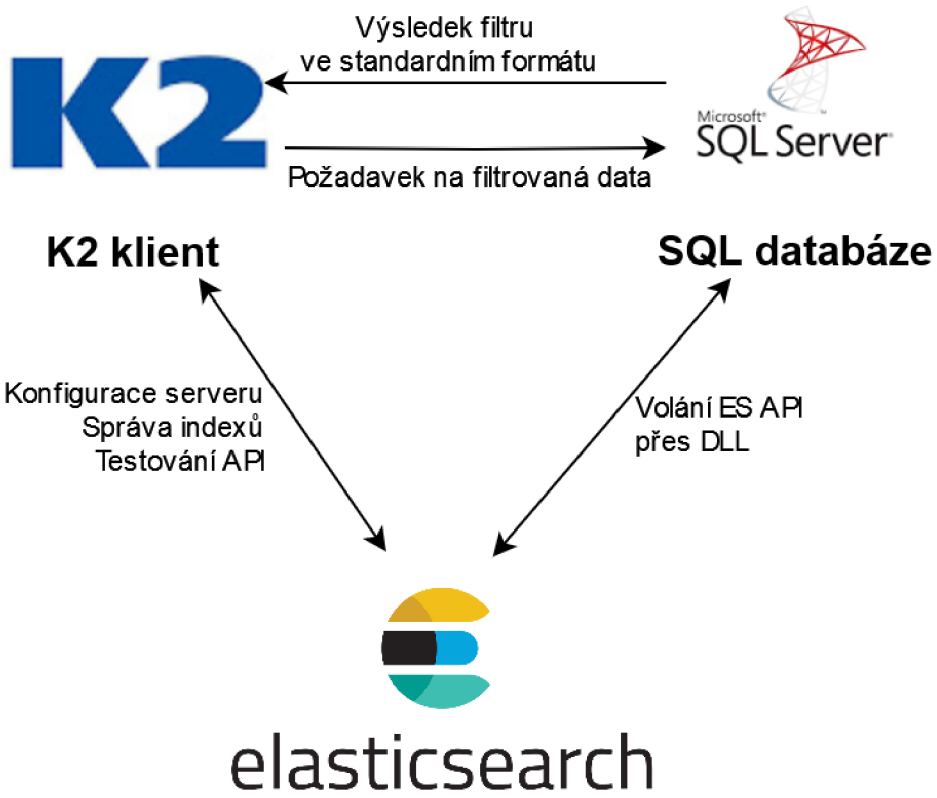
4.3.1 Propojení s SQL serverem

Funkcionalitu Microsoft SQL serveru, jež slouží jako hlavní úložiště dat systému K2, lze rozšířit pomocí tzv. *assemblies* – DDL souborů napsaných v jazyce podporovaném Microsoft .NET běhovým prostředím CLR.[12] Funkce a datové typy z těchto *assemblies* lze využívat skrze SQL funkce či uložené procedury. Pro účely tohoto projektu tedy vzniklo nové *assembly*, které obsahuje funkce pro volání Elasticsearch REST API, konkrétně přímo volání vyhledávacího dotazu. Systém K2 standardním způsobem kontaktuje SQL server skrze dotaz, který volá SQL funkci, jež realizuje funkci daného assembly. SQL server zpracuje data

[12] https://learn.microsoft.com/en-us/sql/relational-databases/clr-integration/assemblies-database-engine?view=sql-server-ver16

zaslaná službou Elasticsearch a vrací je zpět do systému K2 v podobě záznamů tabulky daného datového modulu tak, jak to systém standardně očekává.

Schéma tohoto procesu je graficky znázorněné na obrázku 4.9.



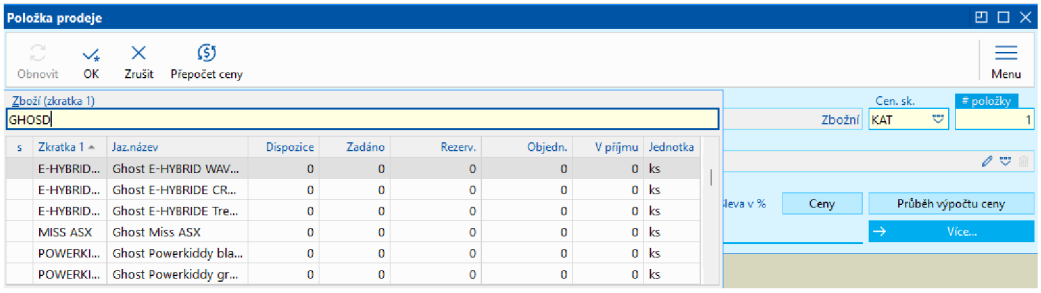
Obrázek 4.9: Diagram zobrazující schéma propojení systému K2, služby Elasticsearch a SQL databázového serveru při realizaci Elasticsearch filtrovací podmínky.

Pro realizaci tohoto propojení vznikl SQL skript, který na SQL serveru registruje jak potřebné *assembly* tak příslušné funkce.

4.3.2 Našeptávač

Novou Elasticsearch podmínku je také možné využít jako náhradu pro funkci *našeptávače*. Našeptávač lze v systému K2 využít při vyplňování vazebních polí, a to v případě, jsou-li v systému nastavené speciální fulltextové indexy. Vyvolává se klávesovou zkratkou `CTRL + MEZERNÍK` a výsledek zobrazuje jako seznam pod vyplňovaným polem. Tato funkce byla rozšířena o Elasticsearch podmínku, kdy místo stávajícího fulltextového indexu lze využít možnosti Elasticsearch. Tímto způsobem lze do našeptávače přidat například právě podporu překlepů či dalších specifických možností skrze vhodně nadefinovaný vyhledávací dotaz. Obrázek 4.10 demonstruje našeptávač s Elasticsearch podmínkou při zadávání nové položky zakázky a jeho schopnost tolerovat překlep. V rámci tohoto projektu jsou navíc im-

plementovány pomocné akce pro snadnou realizaci a nastavení Elasticsearch našeptávače. Konfigurace indexů obsahuje akce pro automatické přidání vhodných polí pro našeptávač a vytvoření základního dotazu.



Obrázek 4.10: Využití našeptávače s Elasticsearch podmínkou při vyhledávání názvu zboží na položku zakázky. Obrázek ukazuje toleranci překlepu v hledaném výrazu.

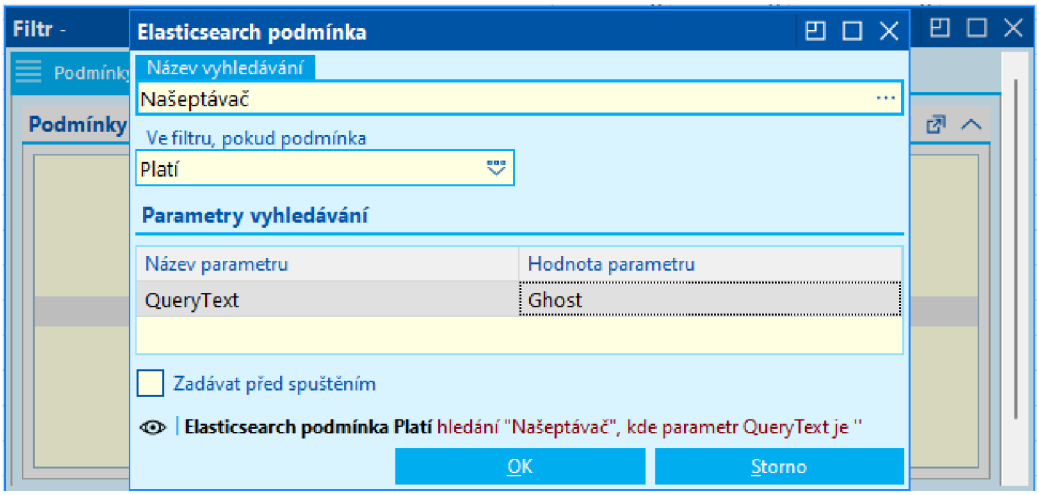
4.3.3 Implementace

Elasticsearch podmínka je implementována třídou `TxElasticsearchCondition`, která dědí od již existující třídy `TxParamCondition`, jež slouží jako základ pro parametrizovatelné filtrovací podmínky. Třída implementuje funkci `GetInternalSQL`, která skládá SQL dotaz, jenž se posílá na SQL server. V této funkci se načítá vyhledávací dotaz z konfigurace indexu příslušného datového modulu, který je zde doplněný o hledaný výraz. Kromě dalších pomocných metod obsahuje ještě třída funkce pro kontrolu podpory Elasticsearch v rámci systému K2 a existence dotazu pro našeptávač – `IsSupported` a `SuggestionsSupported`. Tyto funkce řídí možnost použití Elasticsearch podmínky.

Propojení SQL serveru a K2 je realizováno přes dva skripty. První řeší nastavení serveru na úrovni konfigurace celého systému, dochází zde k vytvoření asymetrického klíče a přihlášení s ním spojené. Druhý skript poté na úrovni mandanta načte požadované assembly ze souboru a vytvoří funkci `clr_https_request`, která provádí zabezpečené volání Elasticsearch API, a funkci `ES_Search_Int`, která je volána ze systému K2 při realizaci Elasticsearch podmínky. Skript také provádí kontrolu existence těchto objektů v databázi a jejich případné odstranění.

4.3.4 Uživatelské rozhraní

Uživatelské rozhraní umožňuje specifikovat Elasticsearch podmínku ve filtru. V rámci formuláře lze vybrat dotaz, jenž je specifikovaný u indexu příslušného datového modulu, který se má použít pro filtrování záznamů. Dále je pak třeba nastavit hledaný výraz, který se poté doplní na příslušná místa v dotazu. V případě, že uživatel vytvořil dotaz, který obsahuje více parametrů, lze je také na tomto místě specifikovat. Ukázka formuláře s Elasticsearch podmínkou a vybraným vyhledáváním *Našeptávač* je na obrázku 4.11. Formulář dále obsahuje další nastavitelné parametry podmínky, které jsou však již součástí standardu.



Obrázek 4.11: Formulář s nastavením Elasticsearch podmínky. Obsahuje vybrané vyhledávání Našeptávač a vyplněný hledaný výraz hodnotou Ghost.

4.4 Struktura zdrojových souborů

Výstupem tohoto projektu je kód napsaný v jazyce Delphi a `c#` a dva SQL skripty. Vzhledem k tomu, že se jedná o projekt, který zasahuje do aktivně vyvíjeného systému a který rozšiřuje jeho funkcionalitu, bylo v některých případech nutné zasáhnout do již existujících zdrojových souborů a provést v něm buďto změny, nebo přidat nové třídy či funkce. Vzniklo také několik nových souborů, do kterých zasáhli i jiní vývojáři. Tato část textu slouží k vymezení a specifikování vlastní autorské implementace.

Konfigurace připojení a indexů

Tato skupina souborů se zabývá konfigurací připojení Elasticsearch serveru a nastavením Elasticsearch indexů:

- `xFile.Elasticsearch.pas` – jedná se o převzatý soubor, který existoval již před tímto projektem a řešil základní konfiguraci Elasticsearch na e-shopu. Soubor byl prakticky kompletně předělán, stávající obsah byl aktualizován, aby byl podporovaný novým uživatelským rozhraním, a byl doplněn o nové funkce a třídy realizující připojení na Elasticsearch server. Nové připojení lze využívat i pro účely e-shopu.
- `xFile.Elasticsearch.Classes.pas` – zcela nový soubor, který implementuje veškerou logiku potřebnou pro nastavení a správu Elasticsearch indexů.
- `Elasticsearch.Classes.pas` – tento soubor implementuje některé funkce, které z technických důvodů nešlo implementovat v předchozím souboru.
- `Presenter.Elasticsearch.pas` – zde se nachází implementace tříd prezentérů pro objekty z předchozích souborů, tedy tento kód obsahuje uživatelské rozhraní pro jednotlivé konfigurace.

- `bb.Shortcut.Elasticsearch.Configurations.pas` – v rámci tohoto souboru jsou implementovány zkratky, přes které lze spouštět jednotlivé konfigurace. Soubor je také doplněn o funkci, která volá SQL skripty s navedením potřebného DLL souboru na databázovém serveru – tato funkce je dílem jiného vývojáře společnosti K2.
- `bb.Shortcut.Elasticsearch.SynchronizeData.pas` – podobně jako v předchozím souboru je zde definována zkratka, která vyvolává synchronizaci dat daných Elasticsearch indexů. Zkratku z tohoto souboru je možné využít v plánovači úloh.

Nastavení databázového serveru

Následující SQL skripty a zdrojový kód pro vytvoření DLL assembly řeší problematiku nastavení databázového serveru tak, aby umožňoval filtrování pomocí Elasticsearch podmínky:

- `K2SQL_Deploy_Conf.sql` – jedná se o zcela nový autorský skript, který realizuje nastavení společné konfigurace systému.
- `K2SQL_Deploy_Data.sql` – podobně jako předchozí soubor je i tento skript autorským dílem a zabývá se navedením DLL souboru a vytvořením potřebných funkcí pro realizaci samotného filtrování.
- `http_request.cs` – tento soubor obsahuje zdrojový kód pro sestavení DLL souboru, který umožňuje volat Elasticsearch API. Soubor již obsahoval metodu pro realizaci HTTP požadavku, a v rámci tohoto projektu do něj byla přidána metoda, která komunikuje s Elasticsearch přes zabezpečený protokol HTTPS.

Filtrování a vyhledávání

Následují soubory, které vznikly před tímto projektem. Jedná se o implementace dosavadního filtrování a vyhledávání a příslušného uživatelského rozhraní.

- `xConditions.pas` – tato jednotka se zabývá implementací filtrovacích podmínek. Ta byla doplněna o novou Elasticsearch podmínku, kterou tak lze nově využít při definici filtrů, a to přidáním třídy `TxElasticsearchCondition` a implementací jejich metod.
- `Model.UFConditions.pas` – tato jednotka slouží jako most mezi starým uživatelským rozhraním a novými modrými formuláři, na které se v době práce na projektu aktivně přecházelo. Přidanou hodnotou tohoto projektu je v tomto souboru třída implementující novou Elasticsearch podmínku tak, aby ji bylo možné používat v modrých formulářích. Přidaná třída se nazývá `TUFElasticSearchCondition`.
- `Presenter.TUFConditions.pas` – zde jsou definovány prezentéry pro jednotlivé filtrovací podmínky. Zde byla vytvořena nová metoda k existující třídě `TPresenterDetailTUFCondition`, která umožňuje výběr z existujících vyhledávacích dotazů daného indexu. Jedná se o metodu `ShowESSearchLookup`.
- `xFile.pas` – v tomto souboru byly upraveny funkce `ApplySearchConditions`, `InitSearchFilterConditions` a `InternalGetSuggestionsData` pro podporu Elasticsearch podmínky.

Přehled všech zdrojových souborů týkajících se tohoto projektu je k dispozici v příloze A.

Kapitola 5

Testování a vyhodnocení

V této kapitole je shrnuto testování a vyhodnocování nově vytvořeného řešení tohoto projektu.

Implementace tohoto projektu byla testována průběžně, a to zpravidla vždy po dokončení nějakého logického celku. Testování probíhalo v rámci oddělení QA[1] společnosti K2 Software s.r.o. Jednalo se o uživatelské testovaní formou přímé interakce člověka s novými moduly nad demo daty, které má společnost k dispozici právě pro tyto účely.

V prvé řadě se ověřovalo, zdali je řešení korektně naprogramované – veškeré funkce jsou dostupné a jejich vyvolání nezpůsobí pád programu, případně zareaguje na vzniklé chyby rozumnými chybovými hlášeními. Toto testování několikrát pomohlo identifikovat situace, které byly během implementace opomenuty a které způsobovaly nekorektní chování programu. Jednalo se například o situaci vícenásobného otevření formuláře *Nastavení Elasticsearch indexů* více různými uživateli, kteří si pak toto nastavení vzájemně přepisovali. Tento problém byl následně vyřešen přidáním omezení na editaci nastavení pouze jedním uživatelem v danou chvíli.

Dále se hodnotilo uživatelské rozhraní. Cílem tohoto testování bylo identifikovat místa, která nebyla uživatelsky přívětivá, nebo nebyla dokonce vůbec pochopitelná. Zde se v rámci testování opravovaly především popisky jednotlivých komponent formulářů či akcí. Na několika místech byla doplněna informační hlášení, která varují uživatele před následky provedení určitých akcí. Jedno z takových hlášení bylo například třeba přidat při ukládání a opouštění konfiguračních formulářů, kdy během editace došlo k provedení akce na Elasticsearch serveru. Uživatel je varován, že pokud neuloží provedené změny v rámci systému K2, může dojít k narušení konzistence mezi oběma místy, a tím pak k nedefinovanému chování jednotlivých funkcí.

V neposlední řadě se testovaly také nové funkce využívající technologie Elasticsearch. Bylo třeba především ověřit kompatibilitu se stávajícím systémem filtrů, a také kombinace nové Elasticsearch podmínky s ostatními podmínkami. Ověřovalo se také nastavení vyhledávacích dotazů, zdali nalezené dotazy odpovídají očekávanému výstupu. Také nová funkce globálního vyhledávání byla otestována v rámci webového klienta.

Hotové řešení bylo prvně nasazeno v rámci firemní instance systému K2, která řídí jednotlivé oblasti společnosti K2. Projekt tak byl otestován i na reálných datech v provozu. Aktuálně

[1] QA – quality assurance (česky zajištění kvality)

je projekt zveřejněn v produkční verzi systému K2 a řešení je k dispozici zákazníkům hned, jakmile aktualizují na požadovanou verzi.

Nově integrovaná technologie není a nejspíš ani nebude schopná plně nahradit stávající způsoby filtrování a vyhledávání, za to ale umožňuje provádět nové operace nad daty, které mohou konkrétním uživatelům usnadnit či zrychlit práci se systémem. Oproti SQL dotazům nad daty v relační databázi, kterými bylo filtrování doposud realizováno, je díky Elasticsearch možné provádět textovou analýzu a zpracování dat z datových modulů, a tím ovlivnit přesnost a relevanci hledání. Nová technologie umí ohodnocovat nalezené shody podle požadovaných kritérií a je tak možné zvýhodňovat některé části dotazu oproti ostatním. Indexované dokumenty jsou ve formátu JSON, což umožňuje indexovat i komplexní datové struktury. Uživatelsky velice přívětivou vlastností je také schopnost „tolerovat“ překlepy vhodným nastavením hledacího dotazu. Prostřednictvím vhodného výběru analyzátorů pro indexovaná pole jde jednoduše uzpůsobit způsob vyhledávání a filtrování podle potřeb firmy bez nutnosti většího zásahu do systému filtrování.

Na druhou stranu Elasticsearch je zaměřený primárně na vyhledávání a analýzu dat, zatímco SQL databáze jsou univerzálnější co se týče správy dat a umožňují lépe definovat vztahy mezi nimi.

S postupným nasazováním služby Elasticsearch u zákazníků se očekává ať už nalezení neobjevených nedostatků tohoto řešení, tak také spousta nových nápadů, jak by šlo vyhledávače využít i na dalších místech systému. Implementace nových rozšíření by díky tomuto projektu měla být již značně jednodušší.

Kapitola 6

Závěr

Tato práce se zabývala integrací vyhledávací technologie Elasticsearch do ERP systému společnosti K2 Software s.r.o. Hlavním cílem bylo zdokonalit a rozšířit stávající způsob filtrování a vyhledávání a také adresovat jejich rychlostní nedostatky. Tato technologie rovněž umožňuje nové způsoby dotazování se nad daty, díky kterým je možné definovat komplexní kritéria pro požadovanou množinu záznamů.

V práci je nejprve popsán samotný systém K2. Je zde přiblížena architektura systému a datová organizace, ve které je hlavním prvkem datový modul. Dále je popsána struktura těchto modulů, včetně možností, jak v nich lze filtrovat a vyhledávat. Poté je popsána technologie Elasticsearch. Jsou zde vyzdviženy její klíčové vlastnosti, dotazovací jazyk a rozhraní, přes které se s ní komunikuje. Část textu se také věnuje existujícím integracím této technologie s tím, že nebyly nalezeny žádné, které by řešily stejný problém jako tato práce, ale za to jich existuje spousta na webových aplikacích či e-shopech. Po seznámení se jak se systémem K2, tak s technologií Elasticsearch se práce věnuje samotnému návrhu a implementaci integrace. Implementace je logicky rozdělená na několik částí, které řeší dílčí problémy celého projektu. V prvé řade se řeší připojení ke službě Elasticsearch a základní nastavení, která jsou sdílena v rámci všech mandantů organizace. Za tímto účelem vznikl v K2 modul *Konfigurace Elasticsearch*. Další řešenou oblastí byla správa indexů, které představují data jednotlivých datových modulů. Nově vzniklý modul, přes který lze kromě správy indexů definovat i vyhledávací dotazy k filtrování a konfigurovat globální hledání, se nazývá *Nastavení Elasticsearch indexů*. Dále je v práci popsána nová filtrovací *Elasticsearch podmínka* a způsob, kterým je možné ji využít pro vytvoření filtru či využití funkce *našeptávače*. U každé nově vzniklé funkcionality je popsáno i její uživatelské rozhraní. Poslední kapitola popisuje uživatelské testování, které bylo provedeno pro validaci nově vzniklého řešení, a také je zde porovnání se standardním způsobem vyhledávání a filtrování.

Podařilo se tedy splnit veškeré požadavky, které byly pro tuto práci stanoveny, o čemž svědčí i fakt, že výstup této práce je již nasazen v produkční verzi systému K2 a nově implementované funkce jsou tak k dispozici uživatelům. Při práci na tomto úkolu se objevily i další možnosti, jak využít nově zabudovaného nástroje pro vyhledávání. Jednou z nich je například realizace našeptávače v rámci webového klienta, který by zobrazoval a aktualizoval nalezené shody hned po každém zadání jednoho znaku do vstupního pole datového modulu s vazbou do jiného datového modulu. Další možností by bylo využít Elasticsearch i indexování textových dokumentů typu Word či PDF, ve kterých by následně bylo možné

skrze tuto technologii vyhledávat. V neposlední řadě se nabízí využít tuto technologii pro účely inteligentní nápovědy využívající techniky RAG[1]. Elasticsearch by tak šlo využít jako databázi znalostí a přes textové, vektorové nebo sémantické vyhledávání z něj získávat relevantní kontext pro nějaký jazykový model, který by tak byl schopen odpovídat na otázky uživatelů ohledně systému K2 [4].

Integrací této technologie se otevírají nové možnosti, které pomáhají společnosti K2 Software s.r.o. nabízet stále lepší verze jejich produktu a jejím zákazníkům zase z nových funkcí těžit při každodenní práci. Práce byla také velkým přínosem i pro mě samotného, jelikož jsem měl možnost seznámit se s pracovním prostředím vetší české IT firmy a možnost obohatit se znalostmi svých zkušenějších kolegů. Pozitivně vnímám také fakt, že jsem schopen aplikovat své teoretické vědomosti i v praxi, a tím být přínosem ať už samotné společnosti, tak i jejím zákazníkům.

[1]RAG – Retrieval Augmented Generation

Literatura

[1] Asociace za lepší ICT řešení, o.p.s.. *ERP systémy | Největší katalog ICT řešení.* online. 2020. Dostupné z: https://lepsi-reseni.cz/prehledy/erp-systemy/. [cit. 2024-04-28].

[2] Elasticsearch B.V.. *Elasticsearch Guide [8.2] | Elastic.* online. 2024. Dostupné z: https://www.elastic.co/guide/en/elasticsearch/reference/current. [cit. 2024-04-29].

[3] Elasticsearch B.V.. *Elasticsearch: The Official Distributed Search & Analytics Engine.* online. 2024. Dostupné z: https://www.elastic.co/elasticsearch. [cit. 2024-05-05].

[4] Elasticsearch B.V.. *What is Retrieval Augmented Generation (RAG)? | A Comprehensive RAG Guide.* online. 2024. Dostupné z: https://www.elastic.co/what-is/retrieval-augmented-generation. [cit. 2024-04-29].

[5] Gartner. *Enterprise Resource Planning (ERP).* online. 2019. Dostupné z: https://www.gartner.com/en/information-technology/glossary/enterprise-resource-planning-erp. [cit. 2024-04-28].

[6] Gormley, C. a Tong, Z. *Elasticsearch: The Definitive Guide: A Distributed Real-Time Search and Analytics Engine.* O'Reilly Media, Inc., leden 2015. ISBN 978-144-93-5854-9.

[7] K2 atmitec s.r.o.. *K2.* online. 2024. Dostupné z: https://www.k2.cz. [cit. 2024-04-29].

[8] K2 atmitec s.r.o.. *Nápověda systému K2.* online. 2024. Dostupné z: https://help.k2.cz. [cit. 2024-04-29].

[9] K2 atmitec s.r.o.. *Z čeho se systém K2 skládá?.* online. 2024. Dostupné z: https://www.k2.cz/cs/z-ceho-se-system-k2-sklada. [cit. 2024-04-29].

[10] Nikita Kathare, V. P. A Comprehensive Study of Elasticsearch. *International Journal of Science and Research (IJSR)* online, Červen 2021, sv. 10, č. 6, s. 716–720. ISSN 2319-7064. Dostupné z: https://doi.org/10.21275/SR21529233126. [cit. 2024-04-20].

[11] SAP. *What is ERP | Enterprise resource planning definition | SAP Insights.* online. 2023. Dostupné z: https://www.sap.com/products/erp/what-is-erp.html. [cit. 2024-04-28].

[12] Sphinx Technologies Inc.. *Sphinx | Open Source Search Engine.* online. 2024. Dostupné z: http://sphinxsearch.com/. [cit. 2024-05-05].

[13] The Apache Software Foundation. *Apache Lucene.* online. 2024. Dostupné z: https://lucene.apache.org/. [cit. 2024-05-05].

[14] The Apache Software Foundation. *Solr Features.* online. 2024. Dostupné z: https://solr.apache.org/features.html. [cit. 2024-04-29].

Příloha A

Obsah přiloženého paměťového média

```
.
├── dokumentace
│   ├── obrazky-figures
│   │   └── ...
│   ├── xszymi00-Elasticsearch-ERP-01-kapitoly-chapters.tex
│   ├── xszymi00-Elasticsearch-ERP-20-literatura-bibliography.bib
│   ├── xszymi00-Elasticsearch-ERP-30-prilohy-appendices.tex
│   ├── xszymi00-Elasticsearch-ERP.tex
│   └── ...
├── src
│   ├── configuration
│   │   ├── Elasticsearch.Classes.pas
│   │   ├── Presenter.ElasticSearch.pas
│   │   ├── bb.Shortcut.Elasticsearch.Configurations.pas
│   │   ├── bb.Shortcut.Elasticsearch.SynchronizeData.pas
│   │   ├── xFile.Elasticsearch.Classes.pas
│   │   └── xFile.Elasticsearch.pas
│   ├── filtering
│   │   ├── Model.UFConditions.pas
│   │   ├── Presenter.TUFCondition.pas
│   │   ├── xConditions.pas
│   │   └── xFile.pas
│   └── sql
│       ├── K2SQL_Deploy_Conf.sql
│       ├── K2SQL_Deploy_Data.sql
│       └── http_request.cs
├── xszymi00-Elasticsearch-ERP.pdf
└── README.md
```

Ve složce `Dokumentace` se nachází veškeré zdrojové soubory potřebné k vytvoření textového dokumentu této práce. Složka `src` obsahuje zdrojové soubory systému K2, které vznikly nebo do kterých bylo v rámci této práce zasaženo. V souboru `README.md` je popsán postup, pomocí kterého lze řešení této práce zprovoznit.