Implementace propojení SAP CTS se systémem Jira

Jakub Doležal

Bakalářská práce
2024

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav informatiky a umělé inteligence

Akademický rok: 2023/2024

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Jakub Doležal**
Osobní číslo:	**A21048**
Studijní program:	**B0613A140020 Softwarové inženýrství**
Forma studia:	**Prezenční**
Téma práce:	**Implementace propojení SAP CTS se systémem Jira**
Téma práce anglicky:	**Implementation of SAP CTS and Jira Integration**

Zásady pro vypracování

1. Prostudujte systém SAP CTS a plugin Atlassian Jira Release Management.
2. Popište možnosti využití integrace pluginu Jira Release Management se SAP CTS.
3. Navrhněte třídu systému SAP pro komunikaci s API Jira Release Management.
4. Implementujte integraci obou systémů.
5. Vytvořte uživatelskou dokumentaci a vyhodnoťte přínosy propojení těchto dvou systémů.

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

1. BANDARI, Kiran. Complete ABAP. SAP Press, 2022. ISBN 978-1-4932-2305-3,
2. KÖSEGI, Armin a Rainer NERDING. *SAP Change and Transport Management*. 3rd ed. Boston: Rheinwerk Publishing Inc., 2009. ISBN 978-1-59229-529-6.
3. GUPTA, Raja a JHA, Sandip. Devops with SAP. S.l.: SAP PRESS, 2023. ISBN 978-1-4932-2420-3.
4. LI, Patrick. Jira software essentials: plan, track, and release great applications with Jira software. Second edition. Birmingham, UK: Packt Publishing, 2018. ISBN 978-1-78883-608-1.
5. BAUMBUSCH, Lutz, Matthias JÄGER a Michael LENSCH. *ABAP RESTful application programming model: the comprehensive guide*. 1. vyd. Boston: Rheinwerk Publishing, 2022. ISBN 978-1-4932-2379-4.

Vedoucí bakalářské práce: **Ing. Tomáš Dulík, Ph.D.**
Ústav informatiky a umělé inteligence

Datum zadání bakalářské práce: **5. listopadu 2023**
Termín odevzdání bakalářské práce: **13. května 2024**

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky
DĚKAN
L.S.

doc. Ing. Jiří Vojtěšek, Ph.D. v.r.
děkan

prof. Mgr. Roman Jašek, Ph.D., DBA v.r.
ředitel ústavu

Ve Zlíně dne 5. ledna 2024

Prohlašuji, že

- beru na vědomí, že odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že bakalářské práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky. Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – bakalářskou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně 9. 5. 2024

Jakub Doležal v.r.

podpis autora

ABSTRAKT

Tato bakalářská práce se zabývá vytvořením RESTful komunikace mezi systémem SAP Change and Transport System (CTS) a nástrojem Jira Release Management od společnosti Atlassian, který je využíván ve společnosti HP TRONIC Zlín pro řízení vývoje. Cílem práce je propojit tyto dvě klíčové technologie a umožnit synchronizaci informací mezi Jira a SAP CTS. Toto řešení přispívá k automatizaci procesů, zlepšuje sledovatelnost změn a usnadňuje správu verzí software. Práce popisuje návrh a implementaci RESTful služeb v SAP pomocí tříd ABAP, které jsou volány z „webhooks“ systému Jira při změně stavu verzí vývoje. Tyto třídy zpracovávají přijatá data, aktualizují stav projektu v SAP CTS a odesílají odpovědi zpět do Jiry.

Klíčová slova: CTS, Jira release management, RESTful, SAP, Webhook

ABSTRACT

This bachelor thesis deals with creating RESTful communication between SAP Change and Transport System (CTS) and Jira Release Management tool from Atlassian, which is used in HP TRONIC Zlín for development management. The aim of the work is to connect these two key technologies and enable synchronization of information between Jira and SAP CTS. This solution contributes to process automation, improves change traceability and facilitates software version management. This thesis describes the design and implementation of RESTful services in SAP using ABAP classes that respond to webhooks triggered in Jira when development versions change state. These classes process the received data, update the project status in SAP CTS and send the responses back to Jira.

Keywords: CTS, Jira release management, RESTful, SAP, Webhooks

Tímto bych chtěl poděkovat svému vedoucímu Ing. Tomáši Dulíkovi, Ph.D. za odborné vedení a podporu během práce na této bakalářské práci a Ing. Jakubu Verbovskému za skvělé nápady, rady a cenné připomínky.

Prohlašuji, že odevzdaná verze bakalářské/diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

OBSAH

ÚVOD

Tato bakalářská práce se zaměřuje na integraci dvou významných nástrojů používaných v oblasti softwarového inženýrství a správy projektů: SAP Change and Transport System (CTS) a Atlassian Jira Release Management pluginu. Tyto systémy jsou základními stavebními kameny pro správu verzí a změn v mnoha organizacích. Přestože každý z nich slouží odlišným účelům, jejich synergické propojení může přinést mnohé výhody pro efektivnější správu release cyklů a lepší sledování změn ve velkých softwarových projektech.

Prvním krokem této práce je podrobná analýza systému SAP CTS a pluginu Atlassian Jira Release Management. Cílem je pochopit jejich základní funkce, architekturu a rozhraní API, která umožňují jejich vzájemnou komunikaci.

Hlavním praktickým výstupem práce je návrh a implementace třídy v systému SAP pro komunikaci s API Jira Release Management. Tato část zahrnuje vytvoření tříd v programovacím jazyku ABAP, generování Json web tokenů, vytvoření webhooku a samotné automatizace systému Jira, která propojuje oba systémy a umožňuje jejich vzájemnou interakci.

V závěrečné části se práce soustředí na vytvoření uživatelské dokumentace pro budoucí vývojáře a programátory. Dokumentace popisuje potřebné kroky pro úspěšné použití automatizace.

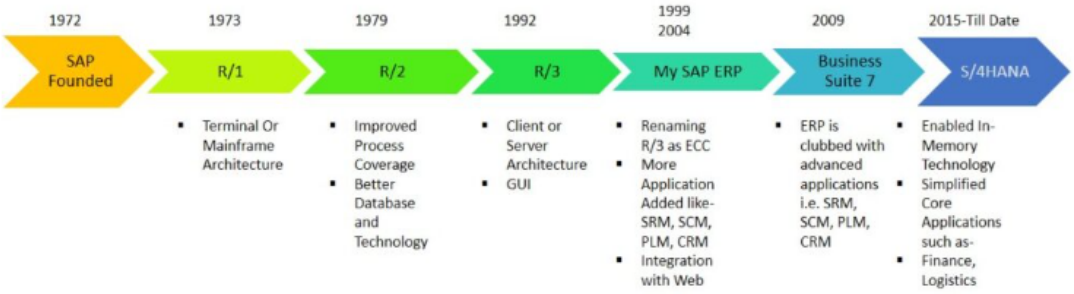
I. TEORETICKÁ ČÁST

1 Úvod do SAP

Systém SAP je komplexní soubor integrovaných podnikových aplikací a řešení navržených k zefektivnění a automatizaci různých obchodních procesů v organizacích různých velikostí a odvětví. SAP, stojící za "Systémy, Aplikace a Produkty v zpracování dat", nabízí širokou škálu softwarových řešení, která pokrývají všechny aspekty podnikání, od účetnictví a řízení financí, přes výrobu, dodavatelský řetězec, prodej a marketing, až po řízení lidských zdrojů a zákaznické vztahy. [1]

Obr. 1.1 Sap logo [2]

1.1 Historie a vývoj SAP



Obr. 1.2 Historie verzí produktu SAP [3]

Společnost SAP, byla založena v roce 1972 pěti bývalými zaměstnanci společnosti IBM v Mannheimu v Německu. Zakladateli byli Dietmar Hopp, Hasso Plattner, Claus Wellenreuther, Klaus Tschira a Hans-Werner Hector.

Cílem SAP bylo vytvořit standardní aplikační software pro zpracování dat v reálném čase, což byl v té době revoluční přístup. První produkt společnosti, SAP R/1, byl vydán v roce 1973. Byl to finanční systém, který představoval základ pro další vývoj řady produktů. V roce 1979 následoval významnější produkt SAP R/2, který umožňoval integraci různých obchodních procesů v jednom systému.

SAP R/2 byl hlavně určen pro velké společnosti a byl schopen zpracovávat různé jazyky a měny, což umožňovalo internacionalizaci společnosti.

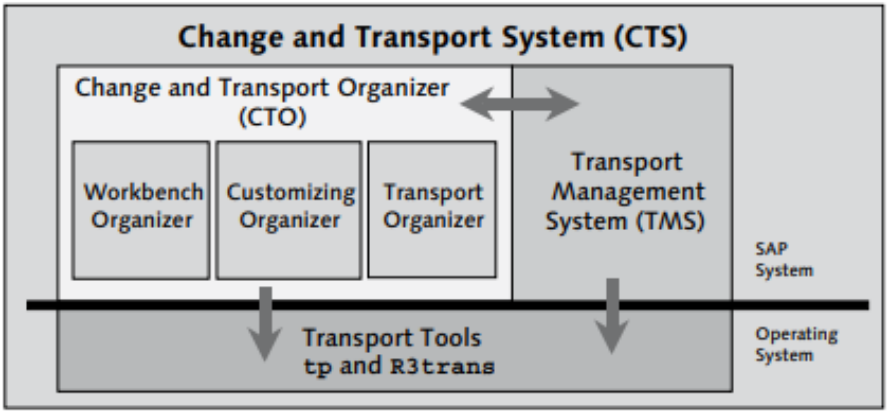
V 90. letech SAP představila SAP R/3, klient-server architekturu, která umožňovala firmám snadněji přizpůsobit software svým potřebám. SAP R/3 zaznamenal obrovský úspěch a stal se základem pro další rozvoj společnosti. SAP začala expandovat do mezinárodních trhů a stala se jedním z největších poskytovatelů podnikového softwaru na světě.

V novém tisíciletí SAP pokračuje v inovacích a rozšiřuje své portfolio produktů o cloudové služby, mobilní aplikace, platformy pro zpracování velkých dat a umělou inteligenci. Mezi klíčové produkty patří SAP S/4HANA, cloudová platforma SAP Cloud Platform a různé řešení pro analýzu dat, lidské zdroje a řízení vztahů se zákazníky. [4]

1.2 Základy SAP CTS

Je to nástroj určený pro transport dat mezi různými prostředími, který usnadňuje správu změn a transportů v rámci systému SAP.

Díky tomuhle nástroji je možné v systému SAP lépe sledovat a přenášet změny z vývojových prostředí do testovacích a produkčních prostředí. Tento systém zajišťuje kontrolu verzí a udržuje konzistenci dat a objektů napříč různými systémy a prostředími SAP. Je to klíčový nástroj pro správu a zabezpečení vývoje a implementace změn v prostředí SAP. [5]



Obr. 1.3 Komponenty CTS [5]

1.3 Komponenty SAP CTS

Skládá se z:

- Organizátor změn a transportu (CTO)

- Systém řízení transportu (TMS)
- Programy TP a R3trans

1.3.1 Organizátor změn a transportu (CTO)

Customizing Organizer *Umožňuje vytváření, dokumentaci a vydání požadavku na změny vytvořených během přizpůsobení. Umožňuje také lidem implementujícím systém SAP ERP sledovat své změny v požadavcích na změnu, zobrazit požadavky na změnu, u kterých jsou zodpovědní a následně zpřístupnit změny ostatním uživatelům systémů uvolněním požadavku na změny.* (Přeloženo autorem bakalářské práce[5] str.123) [1)]

Workbench Organizer *Datový modelář je propojen s organizérem Workbench. To znamená, že změny modelovacích objektů jsou zaznamenávány a mohou být přenášeny do jiných systémů. Workbench Organizer zajišťuje, že existuje pouze jedna původní verze objektu. Pouze tuto původní verzi objektu lze měnit (v systému, kde se nachází).* (Přeloženo autorem bakalářské práce) [6] [2)]

Transport Organizer Zajišťuje správu a koordinaci transportních požadavků a objektů mezi různými systémy. Jedná se o klíčovou komponentu pro správu změn v systémech SAP, která umožňuje uživatelům vytvářet, spravovat a sledovat transporty objektů, jako jsou programy, tabulky a konfigurační nastavení, mezi vývojovým, testovacím a produkčním prostředím. [7]

Jakmile jsou objekty úložiště zahrnuty do požadavku na změnu, je možné je upravovat pouze v tomto požadavku. To znamená, že dokud není požadavek na změnu uvolněn, jsou uzamčeny proti vývojovým pracím nebo údržbě jinými vývojáři, kteří na tomto požadavku na změnu nepracují. Tito vývojáři mohou objekty pouze zobrazovat. (Přeloženo autorem bakalářské práce) [7] [3)]

[1)] It enables the creation, documentation, and release of change requests generated during Customizing. It also enables the people implementing SAP ERP to track their changes to change requests, view the change requests for which they are responsible, and then make the changes available to other systems by releasing the change requests.

[2)] The Data Modeler is linked to the Workbench Organizer. This means that changes to modeling objects are logged and can be transported to other systems. The Workbench Organizer ensures that there is only one original version of an object. Only this original version of an object can be changed (in the system where it is located).

[3)] Once you have included Repository objects in a change request, you can edit them in this request only. This means that until the change request has been released, they are locked against development work or maintenance by other developers not working on this change request. These developers are only allowed to display the objects.

Tímto způsobem Organizátor transportu zabraňuje nekoordinovaným, paralelním změnám objektů.

1.3.2 Systém řízení transportu (TMS)

Pomocí systému TMS (Transport Management System) můžete organizovat, provádět a monitorovat transporty mezi systémy SAP. (Přeloženo autorem bakalářské práce[8])[4] Mezi hlavní funkce TMS patří konfigurace cest transportu pomocí grafického editoru, zobrazení importních front pro všechny SAP systémy v transportní doméně, importování všech požadavků ve frontě importu, projektu, nebo import jednotlivých požadavků.

1.3.3 Programy TP a R3trans

V systému SAP jsou programy TP (Transport Control Program) a R3trans klíčové nástroje pro správu a realizaci transportů (transportů) mezi různými systémy SAP. Tyto nástroje umožňují bezpečné přenášení dat a konfigurací v rámci prostředí SAP, což je zásadní pro správu změn, aktualizací a udržování konzistence dat mezi vývojovým, testovacím a produkčním prostředím.

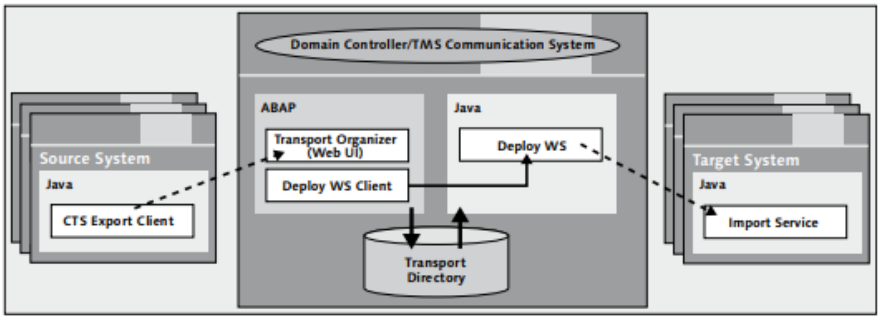
TP (Transport Control Program) TP je vysokoúrovňový nástroj používaný k řízení transportních procesů v SAP. Slouží jako rozhraní pro správu transportních požadavků a transportních objektů TP koordinuje procesy transportu, jako je balení objektů do transportních balíků, spouštění samotných transportů a monitorování jejich stavu. Většinou je volán prostřednictvím SAP GUI.[5]

R3trans je nižší úroveň a specializovanější nástroj, který se přímo stará o transport dat mezi systémy. Tento program provádí skutečnou migraci dat a objektů na úrovni databáze a je schopen pracovat s transportními soubory, které definují, jaká data mají být přenesena. R3trans se používá pro migrace mezi různými verzemi SAP, opravy v případech, kdy standardní metody selhávají, nebo pro komplexní manipulaci s daty v databázi. R3trans se používá jen ve výjimečných situacích, protože k jeho použití jsou zapotřebí pokročilé znalosti SAP systémů a databází. [9]

[4] You can organize, perform, and monitor transports between your SAP Systems using the Transport Management System (TMS).

1.3.4 CTS plus

Je vhodné také zmínit, že pro přenášení non-ABAP objektů slouží CTS+, společně v rámci jednoho požadavku na transport. Pro použití vylepšeného systému změn a přepravy je potřeba alespoň SAP NetWeaver 7.0 ABAP a jeden SAP NetWeaver 7.0.



Obr. 1.4 Transport přes všechna systémová prostředí [5]

Důležité komponenty jsou:

- *Domain Controller (ABAP) je systém, ve kterém jsou nastaveny přepravní trasy TMS.* (Přeloženo autorem bakalářské práce[5] str.620) [5)]
- *TMS komunikační systém (ABAP) je systém ABAP, ve kterém je spuštěn transportní program tp, který provádí kroky importu. Obvykle je doménový řadič stejný systém, jako komunikační systém.* (Přeloženo autorem bakalářské práce[5] str.620) [6)]
- *Deploy Web Service (Java) je webová služba v jazyce Java, která komunikuje s nástroji pro nasazení systémů, které nejsou součástí systému ABAP. Transportní program tp komunikuje s webovou službou CTS Deploy Web Service za účelem nasazení objektů, které nejsou součástí systému ABAP.* (Přeloženo autorem bakalářské práce[5] str.620) [7)]
- *Deploy Web Service Client Povolení programu pro řízení transportu tp v systému SAP NetWeaver AS ABAP komunikovat se službou Deploy Web Service*

[5)] The Domain Controller is the system in which the TMS transport routes are set up.

[6)] TMS communication system (ABAP) is the ABAP system in which the transport program tp is triggered to perform the import steps. Usually, the Domain Controller is the same system as the communication system

[7)] The Deploy Web Service is a Java web service that communicates with the deployment tools of non-ABAP systems. The transport program tp communicates with the CTS Deploy Web Service to deploy non-ABAP objects.

v systému SAP NetWeaver AS Java, je klient webové služby Deploy (Deploy Proxy). pro ABAP. (Přeloženo autorem bakalářské práce[5] str.620) [8)]

- *Transport Organizer Web UI (ABAP) je aplikace ABAP Web Dynpro sloužící k vytváření a úpravám transportních požadavků pro systémy, které nejsou založeny na ABAP.* (Přeloženo autorem bakalářské práce[5] str.620) [9)]

1.4 Implementace REST API v SAP

V SAP systémech se REST API používá k integraci s externími aplikacemi a taky poskytování přístupu k funkcionalitám SAP prostřednictvím webových služeb. V SAP existuje několik způsobů, jak implementovat a vystavit REST API.

1.4.1 SAP Gateway

SAP Gateway je efektivní způsob, jak vytvořit OData služby (které jsou založené na REST principu), umožňující integraci se SAP i non-SAP aplikacemi. Gateway umožňuje vývojářům vytvářet, publikovat a spravovat RESTful webové služby. OData služby poskytované přes SAP Gateway jsou dobře strukturované a podporují operace (Create, Read, Update, Delete). [10]

1.4.2 SAP Cloud Platform API Management

Tento nástroj umožňuje správu, zabezpečení a monitorování API vystavených ze SAP a non-SAP systémů. Umožňuje vývojářům konfigurovat API Gateway, který spravuje volání API a zabezpečuje je pomocí protokolů jako OAuth. SAP Cloud Platform API Management podporuje vývoj REST API a jejich integraci s dalšími cloudovými nebo on-premise systémy. [11]

1.4.3 Custom RESTful Web Service

V ABAP prostředí lze také vytvářet vlastní RESTful služby pomocí klasických ABAP tříd. ABAP třídy mohou být využity pro manipulaci s HTTP požadavky a odpověďmi přímo, umožňující vývojářům plnou kontrolu nad funkcionalitou API. Tento přístup a zároveň tahle bakalářská práce zahrnuje práci s třídami jako je cl_rest_http_handler pro naslouchání na HTTP požadavcích a if_rest_ application pro zpracování požadavků. [12]

[8)] To enable the transport control program tp on SAP NetWeaver AS ABAP to communicate with the Deploy Web Service on SAP NetWeaver AS Java, a Deploy Web Service Client (Deploy Proxy) is required for ABAP.

[9)] The Transport Organizer Web UI is an ABAP Web Dynpro application used to create and modify transport requests for non-ABAP systems.

1.5 Architektura SAP CTS

Hlavní cíl Architektury v systému SAP CTS je, aby zabezpečil efektivní správu, sledování a implementaci změn v konfigraci systémového softwaru a aplikací. Tento systém zajišťuje, že všechny změny mohou být bezpečně přeneseny z vývojového prostředí do testovacího a následně do produkčního prostředí. Důraz je kladen na integritu dat a konzistenci systémových nastavení v rámci různých prostředí. [5]

1.5.1 Vývojový systém

V tomto prostředí vývojáři vytvářejí a testují své změny. Vývojový systém je výchozím bodem pro všechny změny, které mají být transportovány do ostatních systémů. Každá změna je zabalena do transportního požadavku, který obsahuje všechny relevantní objekty a informace potřebné pro transport. [5]

1.5.2 Transportní adresář

Transportní adresář je sdílený úložný prostor na operačním systému, který obsahuje všechny transportní požadavky a jejich data. Tento adresář je přístupný ze všech systémů v transportní krajině a slouží jako centrální bod pro ukládání a správu transportních souborů. [13]

1.5.3 Kontroler transportní domény

Kontroler transportní domény koordinuje transportní procesy napříč všemi systémy v SAP krajině. Udržuje informace o konfiguraci transportního systému, včetně definice systémů a tras, kterými mohou být změny transportovány. Všechny systémy v transportní krajině jsou registrovány u kontroloru domény, což umožňuje jejich vzájemnou komunikaci a správu transportů. [5]

1.5.4 Transportní trasy

Transportní trasy definují cesty, kterými mohou být změny transportovány z vývojového systému do ostatních cílových systémů, jako jsou testovací, kvalifikační a produkční prostředí. Tyto trasy určují, jak změny postupují skrze systémovou krajinu a zajišťují, že změny jsou aplikovány ve správném pořadí a kontrolovaně.[14]

1.5.5 Typy transportních požadavků

- *Typ K - se změnou integrovaného systému na konsolidovaný systém.* (Přeloženo autorem bakalářské práce[15]) [10)]

[10)] K type – with change in integrated system to consolidated system

- *Typ C - bez změny integrovaného systému na konsolidovaný systém* (Přeloženo autorem bakalářské práce[15]) [11)]
- *Typ T - transport jednoho systému do jiného systému* (Přeloženo autorem bakalářské práce[15]) [12)]

1.6 Bezpečnost a kontrola v CTS

CTS využívá komplexní systém pro autentizaci uživatelů, aby zajistil, že k transportu změn jsou oprávněni pouze autentizovaní uživatelé.

1.6.1 Autentizace uživatelů

- Přihlašovací údaje: Uživatelé musí zadat své uživatelské jméno a heslo, které jsou ověřeny proti databázi uživatelů SAP.
- Single Sign-On (SSO): V mnoha případech je CTS integrován s řešeními Single Sign-On, což umožňuje uživatelům přistupovat k CTS bez nutnosti opakované autentizace, pokud jsou již autentizováni v rámci podnikové sítě.
- Certifikáty a šifrování: Pro přístup k CTS přes síť mohou být použity digitální certifikáty a šifrování spojení, což zvyšuje bezpečnost autentizace uživatelů.[16]

1.6.2 Autorizace uživatelů

Po autentizaci je dalším krokem zajištění, že uživatelé mají správné oprávnění k vykonávání specifických akcí v CTS. Toto je realizováno pomocí systému založeného na rolích a oprávněních:

- Role a oprávnění: uživatelům jsou přiřazeny role, které určují, které akce mohou v systému vykonávat. Například někteří uživatelé mohou mít oprávnění k vytváření transportních požadavků, zatímco jiní mohou mít oprávnění pouze k prohlížení.
- Správa oprávnění: oprávnění jsou spravována centrálně, což umožňuje administrátorům SAP snadno kontrolovat přístup a zajišťovat, že uživatelé mají pouze taková oprávnění, která potřebují pro svou práci.
- Auditování přístupu: CTS zaznamenává logy přístupu a akcí, což umožňuje sledovat, kdo provedl jaké změny a kdy. To je klíčové pro revizi bezpečnosti a vyšetřování případných bezpečnostních incidentů. [16]

[11)] C type – without change in integrated system to consolidated system
[12)] T type – move the one system to another system

1.6.3 Zabezpečení transportu

Samotný transport změn je zabezpečen několika mechanismy:

- transportní cesty a domény: CTS používá definované transportní cesty a domény k zajištění, že změny jsou transportovány pouze mezi autorizovanými systémy v rámci kontrolovaného procesu.
- Šifrování transportů: data transportovaná mezi systémy mohou být šifrována, což zabraňuje jejich odposlechu nebo manipulaci během transportu.[16]

1.7 ABAP

ABAP (Advanced Business Application Programming) je programovací jazyk vyvinutý společností SAP pro vývoj aplikací v rámci jejich systémů Enterprise Resource Planning (ERP). Tento jazyk je používán primárně pro vývoj aplikací integrovaných do platformy SAP, což zahrnuje aplikace pro zpracování transakcí, reportování a konfiguraci business procesů.

ABAP se vyznačuje tím, že je silně integrován s datovými strukturami systému SAP, což umožňuje efektivní manipulaci s daty a jejich zpracování. Je to jazyk vysoce specializovaný na potřeby SAP systémů a jeho použití je rozšířené hlavně mezi SAP vývojáři a konzultanty. ABAP umožňuje jak procedurální, tak objektově orientované programování a je klíčový pro implementaci uživatelsky specifických požadavků a funkcionalit v rámci standardního software SAP. [17]

2 JIRA

Jira Software společnosti Atlassian je předním řešením pro agilní správu projektu. Používají jej vývojářské týmy po celém světě.

Obr. 2.1 Jira Software logo [18]

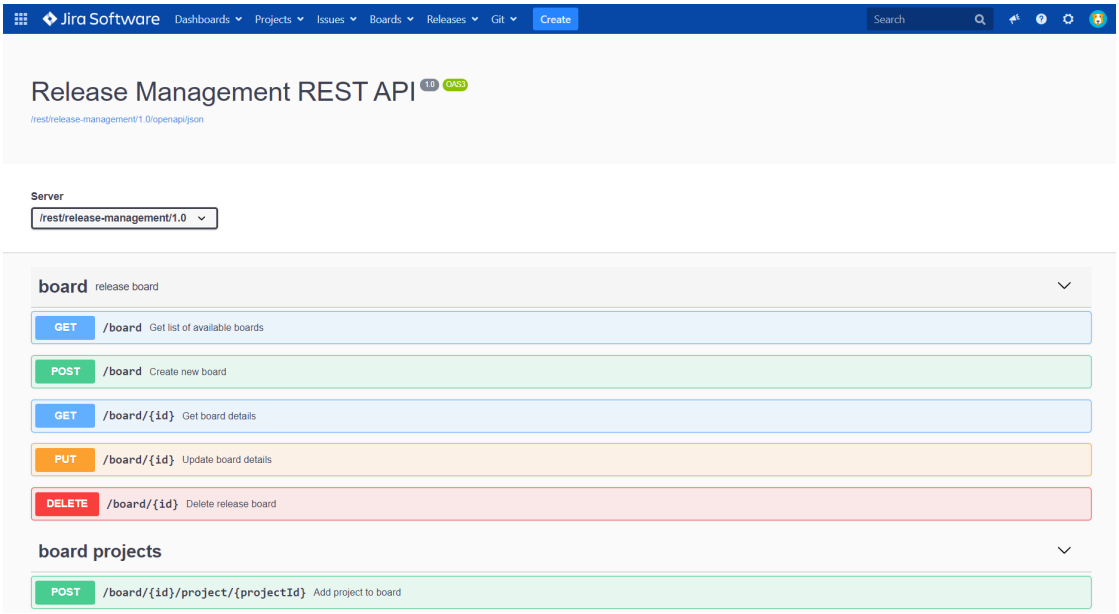
Slouží především jako ucelené centrum pro plánování, sledování a podporu softwarových projektů. Hlavní myšlenka Jiry je zefektivnit postup úkolů, podporuje soudržnost a umožňuje souvislou komunikaci mezi členy týmu. Poskytuje autonomním týmům základní souvislosti pro rychlý pokrok a hodí se od organizování jednoduchých projektů nebo podporuje úsilí DevOps. JIRA podporuje mnoho dalších funkcí, jako jsou statistiky, přehledy, možnosti pokročilého vyhledávání. Důležité je taky zmínit, že Jira stejně jako ostatní produkty Atlassian podporuje vývoj pluginů, Atlassian má bohatou dokumentaci, jak vytvářet doplňky a obohacovat svůj produkt Jira o tzv. další různé funkce a procesy. [19]

2.1 Release Management pro Jiru

Release Management neboli správa verzí je proces plánování, testování a nasazování verzí. Celý tento proces začíná v okamžiku, kdy je naplánováno vydání nové verze do budoucna. Tato situace je vyvolána oznámením chyby nebo když se očekávají nové funkce softwaru. Tyto problémy jsou seskupeny pro novou připravovanou verzi. Poté pokračuje integrační fáze - tzv. vývoj a testování. Software je třeba řádně otestovat. Jedná se o fázi, ve kterém se testuje vydání této verze. Pokud testy selžou, vše se řeší v rámci kontinuálního integračního kruhu. Testování může probíhat automaticky nebo ručně vystavením softwaru testerům. Fáze nasazení se skládá z archivace finální verze, testování a nakonec vydaní verze pro zákazníky. [20]

2.2 Release Management REST API

Rozhraní REST API Jira se používá ke vzdálené interakci s aplikacemi datového centra Jira a umožňuje vývojářům a týmům automatizovat a spravovat procesy uvolňování softwarových verzí v aplikaci Jira. [21]



Obr. 2.2 REST API [19]

2.3 webhook

Webhook v aplikaci Jira Release Management je nástroj, který umožňuje aplikacím Jira komunikovat s externími službami v reálném čase. Kdykoli v systému Jira dojde k nějaké události (například k aktualizaci úkolu, přidání komentáře, změně stavu úkolu atd.), může webhook automaticky odeslat požadavek HTTP na předem definovanou adresu URL externí služby. Tento požadavek může obsahovat podrobnosti o události, která vyvolala odeslání webhooku. [22]

2.4 JSON web token

JSON web token (dále jen „JWT“) je otevřený standard pro bezpečné sdílení dat JSON mezi stranami. Data jsou zakódována a digitálně podepsána, což zajišťuje jejich pravost. JWT se široce používá v pracovních postupech ověřování a autorizace API a také pro transport dat mezi klienty a servery. [23]

II. PRAKTICKÁ ČÁST

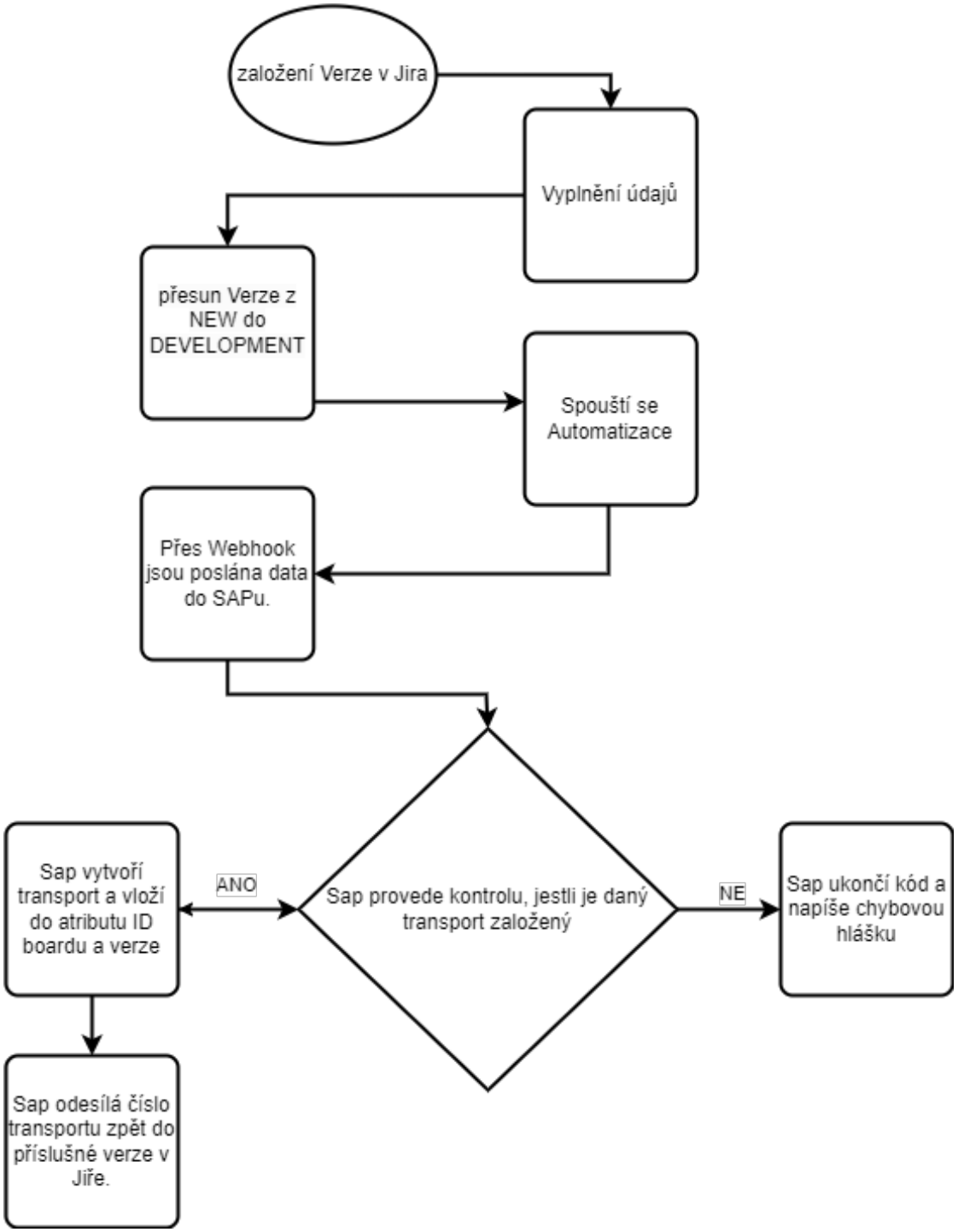
3 Využití integrace Jira Release management se systémeme SAP CTS

Tato praktická část bakalářské práce se zabývá vývojem a implementací systému pro automatizaci procesů mezi JIRA Release Managementem a systémem SAP Change and Transport System (CTS). Cílem bylo vytvořit řešení, které by umožnilo automatické zakládání transportních požadavků v SAP CTS při změně stavu verze v Jira z "NEW"na "DEVELOPMENT". Tato integrace je navržena tak, aby zefektivnila pracovní procesy vývojářů tím, že eliminuje potřebu manuálního zakládání transportních požadavků, čímž šetří čas a snižuje možnost lidských chyb.

Systém byl implementován v ABAPu, a to prostřednictvím třídy zcl_jd_jira_ws_resource, která dědí z abstraktní třídy cl_rest_resource. Tato třída je navržena tak, aby poskytovala RESTful API služby, které komunikují s Jirou a SAP CTS. V rámci této třídy byly implementovány metody pro odesílání dat do Jiry, získání JWT tokenů z URL a kontrolu existence transportního požadavku v Jira.

Hlavní metoda if_rest_resource post zajišťuje zachycení a zpracování požadavků z Jiry. Jira jakmile detekuje změnu při transportu verze, posílá informace na danou URL a kód v SAPu automaticky vytváří nový transportní požadavek pomocí funkčního modulu TR_INSERT_REQUEST_WITH_TASKS, přičemž dynamicky přiděluje atributy transportu na základě informací získaných z Jiry. Jakmile je transportní požadavek úspěšně vytvořen, informace jsou zpětně synchronizovány do Jiry, čímž se aktualizuje stav transportního požadavku.

Tento přístup výrazně zjednodušuje workflow vývojářů a správců systému SAP tím, že automatizuje opakující se úkoly a poskytuje uživatelům okamžité aktualizace o stavu jejich požadavků. V následujících kapitolách se budu podrobněji věnovat technickým detailům implementace, včetně návrhu tříd, metod a integrace s pluginem Jira release management.



Obr. 3.1 Diagram procesu

4 Praktické kroky v SAP

1. Vytvoření dvou tříd v ABAP:

- Třída zcl_jd_jira_ws_handler: Tato třída dědí z cl_rest_http_handler a slouží jako HTTP handler pro RESTful komunikaci se systémem Jira. Implementace obsahuje metodu pro získání root handleru pomocí routeru, který připojuje třídu zcl_jd_jira_ws_resource.
- Třída zcl_jd_jira_ws_resource: Dědí z cl_rest_resource a poskytuje metody pro interakci s Jira API, včetně odesílání dat a kontroly existence transportních požadavků v Jira. Třída také obsahuje definici struktury dat a logovací mechanismus.

2. Vytvoření uživatele v SAP:

- Byla provedena konfigurace systému SAP pro vytvoření nových uživatelských účtů, které jsou nezbytné pro testování a implementaci navržených řešení.

3. Vytvoření služby v SICF pro komunikaci RESTful:

- Byla zřízena služba v SICF (SAP Internet Communication Framework) pro zajištění komunikace mezi SAP a externími systémy pomocí RESTful API, což je klíčové pro integraci se systémem Jira.

4.1 Třída zcl_jd_jira_ws_handler

Kód 1. Třída zcl_jd_jira_ws_handler

```
CLASS zcl_jd_jira_ws_handler DEFINITION
  PUBLIC
  INHERITING FROM cl_rest_http_handler
  FINAL
  CREATE PUBLIC.
  PUBLIC SECTION.
    METHODS:
      if_rest_application~get_root_handler REDEFINITION.
  PROTECTED SECTION.
    METHODS:
      handle_csrf_token REDEFINITION.
  PRIVATE SECTION.
ENDCLASS.
CLASS zcl_jd_jira_ws_handler IMPLEMENTATION.
  METHOD handle_csrf_token.
  ENDMETHOD.
  METHOD if_rest_application~get_root_handler.
```

Kód 2. Třída zcl_jd_jira_ws_handler

```
    DATA(router) = NEW cl_rest_router( ).
    router->attach( iv_template = '' iv_handler_class = '
   ZCL_JD_JIRA_WS_RESOURCE' ).
    ro_root_handler = router.
  ENDMETHOD.
ENDCLASS.
```

Třída zcl_jd_jira_ws_handler je definována jako veřejná (PUBLIC) a konečná (FINAL), to znamená, že od ní nelze dědit. Třída dědí z_cl_rest_http_handler, což je standardní SAP třída používaná pro obsluhu HTTP požadavků v RESTful službách.

- PUBLIC SECTION: Zde je redefinovaná metoda if_rest_application, která umožňuje třídě nastavit kořenový (root) handler pro příchozí HTTP požadavky.
- PROTECTED SECTION: Obsahuje redefinici metody handle_csrf_token, která by byla použita pro zpracování CSRF (Cross-Site Request Forgery) tokenů, což je bezpečnostní prvek chránící před nežádoucími požadavky.
- Metoda if_rest_application get_root_handler: Vytvoří instanci cl_rest_router, která se používá pro směrování příchozích HTTP požadavků. Metoda nastaví, aby třída ZCL_JD_JIRA_WS_RESOURCE byla handlerem pro kořenovou cestu, takže všechny příchozí požadavky na základní URL API budou zpracovány touto třídou.

4.2 Třída zcl_jd_jira_ws_resource

Kód 3. Třída zcl_jd_jira_ws_resource DEFINITION

```
CLASS zcl_jd_jira_ws_resource DEFINITION
      PUBLIC
      INHERITING FROM cl_rest_resource
      FINAL
      CREATE PUBLIC.
  PUBLIC SECTION.
    METHODS:
      if_rest_resource~post REDEFINITION,
      send_data_to_jira,
      get_jwt_from_url
        RETURNING
          VALUE(lv_contextjwt) TYPE string,
      check_transport_in_jira
        RETURNING

          VALUE(rv_continue) TYPE abap_bool.
```

Kód 4. Třída zcl_jd_jira_ws_resource DEFINITION

```
  PROTECTED SECTION.
  PRIVATE SECTION.

    TYPES: BEGIN OF ty_reg,
             owner         TYPE as4user,
             task          TYPE as4text,
             name          TYPE as4text,
             system        TYPE c2,
             idboard       TYPE trvalue,
             versionid     TYPE trvalue,
             transport_id  TYPE trkorr,
           END OF ty_reg.
    DATA: ms_data       TYPE ty_reg,
          ls_header     TYPE trwbo_request_header,
          lv_continue   TYPE abap_bool,
          app_log       TYPE REF TO zcl_jv_log,
          lv_mess       TYPE string,
          lv_jwt_token  TYPE string.
ENDCLASS.
```

V PUBLIC sekci jsou definovány metody, které mohou být volány externě:

- if_rest_resource post - předefinovaná metoda pro zpracování POST požadavků. Zahrnuje deserializaci JSON dat, kontrolu existence transportních požadavků v JIRA a nakonec odesílání dat do JIRA.
- send_data_to_jira - metoda pro odeslání dat do JIRA, specificky pro vytvoření nového záznamů.
- get_jwt_from_url - získává JWT (JSON Web Token) z URL, což je klíčové pro autentizaci požadavků.
- check_transport_in_jira - kontroluje, zda existuje záznam transportu v Jira, což pomáhá rozhodnout, zda pokračovat v procesu.

V PRIVATE sekci jsou definovány typy a proměnné pro interní použití třídy:

- ty_reg - struktura pro ukládání informací o úloze.
- ms_data - instance struktury ty_reg pro uchování dat.
- ls_header - hlavička požadavku.
- lv_continue a lv_mess - proměnné pro logické rozhodnutí a zprávy.
- app_log - reference na třídu logování.

- lv_jwt_token - proměnná pro uchování JWT.

Třída zcl_jd_jira_ws_resource v ABAP je navržena pro interakci s JIRA webovými službami a je odvozena od základní třídy cl_rest_resource tím pádem se jedná o RESTful webovou službu. Třída je označena jako FINAL a CREATE PUBLIC, což znamená, že již nemůže být dále děděna a může být vytvářena veřejně.

4.3 Třída zcl_jd_jira_ws_resource IMPLEMENTATION

V implementaci se nacházejí 4 metody, které pro lepší přehlednost byly rozděleny a popsány odděleně, pro použití kódu je potřeba dodržet ABAP syntax a dát tyhle metody do třídy zcl_jd_jira_ws_resource IMPLEMENTATION.

4.3.1 Metoda if_rest_resource post

Kód 5. Metoda if_rest_resource post

```
METHOD if_rest_resource~post.
      TRY.
          DATA(lv_request_body) = mo_request->get_entity( )
   ->get_string_data( ).
          zcl_jv_json3=>deserialize( EXPORTING json =
   lv_request_body CHANGING data = ms_data ).
          me->app_log = zcl_jv_log=>create( p_object = '
   Z_JIRA' p_subobject = 'CHYBASPOJENI' ).
          lv_jwt_token = get_jwt_from_url( ).
          lv_continue = check_transport_in_jira( ).
          IF lv_continue = abap_false.
            mo_response->set_status( iv_status = 400
   iv_reason_phrase = 'Transport already exists, cannot
   create new one.' ).
            app_log->add_string( 'Transport already exists,
   cannot create new one.' ).
            app_log->save_log( ).
            RETURN.
          ENDIF.
          CALL FUNCTION 'TR_INSERT_REQUEST_WITH_TASKS'
            EXPORTING
              iv_type           = 'K'
              iv_target         = '/TEST/'
              iv_text           = CONV as4text( |WB: {
   ms_data-name }| )
              iv_owner          = ms_data-owner
              it_attributes     = VALUE scts_attrs( (
   attribute = 'SAP_TMS_BS_PLG_REQ_SC' value = ms_data-
   idboard ) ( attribute = 'SAP_TMS_BS_PLG_REQ_SC' value =
   ms_data-versionid ) )
```

Kód 6. Metoda if_rest_resource post

```
            IMPORTING

              es_request_header = ls_header.
          ms_data-transport_id = ls_header-trkorr.
          send_data_to_jira( ).
          IF sy-subrc = 0.
            mo_response->set_status( iv_status = 200
  iv_reason_phrase = 'Success.' ).
          ELSE.
            mo_response->set_status( iv_status = 500
  iv_reason_phrase = 'Failed to send data to Jira.' ).
          ENDIF.
        CATCH cx_root INTO DATA(lx_exception).
          mo_response->set_status( iv_status = 500
  iv_reason_phrase = 'Internal Server Error.' ).
     ENDTRY.
   ENDMETHOD.
```

Metoda if_rest_resource post v třídě zcl_jd_jira_ws_resource zpracovává požadavky POST pro integraci s externím systémem Jira. Tato metoda je klíčovou součástí webové služby a je navržena pro zpracování datových požadavků zaslaných uživatelem nebo jiným systémem. Provádí následující kroky:

1. Příjem a deserializace dat: Na začátku metody se získává tělo požadavku ve formátu řetězce z HTTP entity, které je deserializováno do interní datové struktury ms_data pomocí třídy zcl_jv_json3. Tento krok přeměňuje JSON data na strukturovaný ABAP datový typ, který lze dále zpracovávat.

2. Logování a autentizace: Dále metoda inicializuje logování pro tento proces, což umožňuje zachytit důležité informace nebo chyby. Po inicializaci logování se volá metoda get_jwt_from_url pro získání JWT tokenu, který je nezbytný pro autentizaci následujících požadavků na externí systém.

3. Kontrola existence transportu: Před pokračováním se ověřuje, zda již v Jira existuje příslušný transportní požadavek. Toto se děje pomocí voláním metody check_transport_in_jira. Pokud takový záznam existuje (vrací abap_false), metoda nastaví HTTP stavový kód 400 a vrátí chybovou zprávu, že nelze vytvořit nový transport, protože již existuje.

4. Pokud neexistuje žádný transport v dané Verzi, metoda pokračuje voláním funkce TR_INSERT_REQUEST_WITH_TASKS. Tato SAP funkce vytváří nový transportní požadavek s atributy specifikovanými v datové struktuře ms_data. Nastaví se typ, cíl, text, vlastník a další atributy transportního požadavku.

5. Odeslání dat do JIRA a odpověď: Po úspěšném vytvoření transportního požadavku se pokračuje voláním metody send_data_to_jira, která zodpovídá za odeslání těchto dat do JIRA. Úspěch nebo neúspěch tohoto volání je zachycen a podle výsledku se nastaví odpovídající HTTP stavový kód (200 pro úspěch, 500 pro chybu).

6. Zachycení a zpracování výjimek: Celá logika metody je obalena v bloku TRY-CATCH, který zachytává všechny výjimky vyvolané během vykonávání metody. V případě chyby se nastaví stavový kód 500 a vrací se zpráva o interní serverové chybě.

4.3.2 Metoda check_transport_in_jira

Kód 7. Metoda check_transport_in_jira

```
METHOD check_transport_in_jira.
    DATA: lo_http_client  TYPE REF TO if_http_client,
          lv_url          TYPE string,
          lv_response     TYPE string,
          lv_xstring_json TYPE xstring,
          lt_customfields TYPE string,
          lv_customfields TYPE string.
    TYPES: BEGIN OF ty_s_json,
             customfields TYPE string,
           END OF ty_s_json.
    DATA: lr_json TYPE REF TO ty_s_json.
    FIELD-SYMBOLS: <ls_json> TYPE ty_s_json.
    FIELD-SYMBOLS: <custonfield_json> TYPE any.
    lv_url = |https://rmcloud-prd.releasemanagement.app/
  api/1/board/{ ms_data-idboard }/version-details/{ ms_data
  -versionid }|.
    TRY.
        cl_http_client=>create_by_url(
          EXPORTING
            url                = lv_url
          IMPORTING
            client             = lo_http_client
          EXCEPTIONS
            argument_not_found = 1
            plugin_not_active  = 2
            internal_error     = 3
            OTHERS             = 4 ).
        lo_http_client->request->set_method( 'GET' ).
        CALL METHOD lo_http_client->request->
  set_header_field
          EXPORTING
            name  = 'Authorization'
            value = |JWT { lv_jwt_token }|.
```

Kód 8. Metoda check_transport_in_jira

```
        lo_http_client->send( ).
        lo_http_client->receive( ).
        lv_response = lo_http_client->response->get_cdata(
 ).
        lv_xstring_json = cl_abap_codepage=>convert_to(
lv_response ).
        DATA: data TYPE REF TO data.
        CREATE DATA lr_json.
        ASSIGN lr_json->* TO <ls_json>.
        /ui2/cl_json=>deserialize(
          EXPORTING
            json = lv_response
          CHANGING
            data = data ).
        FIELD-SYMBOLS: <line>         TYPE data,
                       <customfields> TYPE any,
                       <customfield2> TYPE any,
                       <name>         TYPE any,
                       <name2>        TYPE any.
        ASSIGN  data->* TO <line>.
        ASSIGN COMPONENT 'CUSTOMFIELDS' OF STRUCTURE <line
> TO <customfields>.
        ASSIGN <customfields>->* TO <customfield2>.
        ASSIGN COMPONENT '660e5dc73d71fa46b68216ec' OF
STRUCTURE <customfield2> TO <name>.
        ASSIGN <name>->* TO <name2>.
        IF <name2> IS NOT ASSIGNED.
          rv_continue = abap_true.
        ELSE.
          rv_continue = abap_false.
        ENDIF.
      CATCH cx_sy_message_in_plugin_mode INTO DATA(
lx_exception).
        MESSAGE ID lx_exception->msgid TYPE lx_exception->
msgty NUMBER lx_exception->msgno WITH lx_exception->msgv1
 lx_exception->msgv2 lx_exception->msgv3 lx_exception->
msgv4 INTO lv_mess.
        BREAK dolezalja.
        app_log->add_bapiret2( /mibcon/cl_wms_utils=>
msg_to_bapiret_line( ) ).
        app_log->save_log( ).
    ENDTRY.
  ENDMETHOD.
```

Metoda check_transport_in_jira je zodpovědná za ověření existence transportního požadavku v externím systému JIRA před jeho potenciálním vytvořením. Tato metoda je klíčová pro zamezení duplikace záznamů a zajištění integritního toku dat mezi ABAP systémem a JIRA.

1. Inicializace HTTP klienta: Nejprve se vytváří instance HTTP klienta pomocí cl_http_client=>create_by_url, kde je jako parametr předáno URL cílící na konkrétní API endpoint JIRA. Toto URL obsahuje identifikátory pro specifickou nástěnku a verzi, které jsou dynamicky získány z datové struktury ms_data.

2. Nastavení HTTP metody a hlaviček: HTTP klient je nastaven na metodu 'GET', to znamená, že se odesílá požadavek pro čtení dat. Dále se nastavuje hlavička 'Authorization' s JWT tokenem získaným z předchozí autentizační metody, což je nezbytné pro autorizovaný přístup k datům.

3. Odeslání a příjem dat: Po nastavení hlaviček a dalších potřebných parametrů se odesílá HTTP požadavek a následně se přijímají data zpět od serveru. Odpověď serveru je zpracována a převedena z formátu 'cdata' na 'xstring', což umožňuje další manipulaci s daty.

4. Deserializace a analýza dat: Přijatá data jsou deserializována pomocí knihovny '/ui2/cl_json', která převede JSON formát zpět na ABAP struktury a objekty. Zde se specificky hledají informace ve struktuře 'CUSTOMFIELDS' odpovědi JIRA, aby se zjistilo, zda již existuje transportní požadavek s daným identifikátorem.

5. Rozhodnutí o existenci požadavku: Po deserializaci a analýze se kontroluje, zda klíčový atribut, jenž indikuje existenci transportního požadavku, je přítomen a přiřazen. Pokud není atribut nalezen (<name2> není přiřazen), metoda nastaví rv_continue na abap_true, což znamená, že nový transport může být vytvořen. V opačném případě se nastaví abap_false, signalizující, že transport již existuje.

6. Zachycení a zpracování výjimek: Všechny kroky jsou provedeny v bloku TRY-CATCH, který zachytí jakékoliv výjimky vyvolané během HTTP komunikace nebo při zpracování dat. V případě výjimky se zaznamená chybová zpráva a metoda přidá relevantní informace do logu pomocí app_log->add_bapiret2.

4.3.3 Metoda get_jwt_from_url

Metoda get_jwt_from_url v třídě zcl_jd_jira_ws_resource je určena pro získání JWT (JSON Web Token), který je nezbytný pro autentizaci požadavků na webové služby JIRA. Metoda provadí integraci ABAP systému s moderními webovými autentizačními protokoly, což je klíčové pro bezpečný přístup k externím API.

Kód 9. Metoda get_jwt_from_url

```
METHOD get_jwt_from_url.
    TYPES: BEGIN OF ty_s_json,
             contextjwt TYPE string,
           END OF ty_s_json.
    DATA: lo_http_client  TYPE REF TO if_http_client,
          lv_url          TYPE string,
          lv_user         TYPE string,
          lv_password     TYPE string,
          lv_auth_string  TYPE string,
          lv_response     TYPE string,
          lv_xstring_json TYPE xstring.
    DATA: lr_json TYPE REF TO ty_s_json.
    FIELD-SYMBOLS: <ls_json> TYPE ty_s_json.
    lv_url = 'https://hptronic-sandbox-269.atlassian.net/
  plugins/servlet/ac/app.releasemanagement.rmcloud/board?
  classifier=json'.
    lv_user = 'emailová adresa'.
    lv_password = 'API_TOKEN'.
    lv_auth_string = cl_http_utility=>if_http_utility~
  encode_base64( lv_user && ':' && lv_password ).
    TRY.
        cl_http_client=>create_by_url(
          EXPORTING
            url                = lv_url
          IMPORTING
            client             = lo_http_client
          EXCEPTIONS
            argument_not_found = 1
            plugin_not_active  = 2
            internal_error     = 3
            OTHERS             = 4 ).
        CALL METHOD lo_http_client->request->
  set_header_field
          EXPORTING
            name  = 'Authorization'
            value = |Basic { lv_auth_string }|.
        lo_http_client->send( ).
        lo_http_client->receive( ).
        mo_response->set_status( iv_status = 200
  iv_reason_phrase = 'Success.' ).
   lv_response = lo_http_client->response->get_cdata( ).
        lv_xstring_json = cl_abap_codepage=>convert_to(
  lv_response ).
        CREATE DATA lr_json.
        ASSIGN lr_json->* TO <ls_json>.
        /ui2/cl_json=>deserialize(
          EXPORTING
            json = lv_response
          CHANGING
```

Kód 10. Metoda get_jwt_from_url

```
            data = lr_json ).
        lv_contextjwt = <ls_json>-contextjwt.
      CATCH cx_sy_message_in_plugin_mode INTO DATA(
lx_exception).
        MESSAGE ID lx_exception->msgid TYPE lx_exception->
msgty NUMBER lx_exception->msgno WITH lx_exception->msgv1
 lx_exception->msgv2 lx_exception->msgv3 lx_exception->
msgv4 INTO lv_mess.
        BREAK dolezalja.
        app_log->add_bapiret2( /mibcon/cl_wms_utils=>
msg_to_bapiret_line( ) ).
        app_log->save_log( ).
   ENDTRY.
 ENDMETHOD.
```

1. Definice datové struktury: Nejprve je definována datová struktura

   ty_s_json, která obsahuje jediné pole contextjwt typu string. Tato struktura slouží k uchování deserializovaného JSON objektu, který obsahuje JWT.

2. Konfigurace HTTP klienta: Metoda inicializuje HTTP klienta pomocí třídy cl_http_client s URL, které specifikuje cílový endpoint pro získání JWT. Autentizační údaje (uživatelské jméno a heslo) jsou zakódovány do base64 formátu a použity v hlavičce Authorization pro základní HTTP autentizaci.

3. Odeslání a příjem HTTP požadavku: Po konfiguraci hlaviček se HTTP požadavek odesílá a následně se přijímají data. Metoda receive získá odpověď od serveru, která je transformována na xstring pro další zpracování.

4. Deserializace odpovědi: Přijatá data jsou deserializována pomocí standardní ABAP JSON knihovny (/ui2/cl_json). JSON odpověď, obsahující JWT, je převedena do interní datové struktury ABAP. Tento krok je klíčový pro extrakci JWT z odpovědi.

5. Extrakce a uložení JWT: Po deserializaci jsou data přiřazena k dynamickým datovým objektům (<ls_json>) a JWT je extrahován z odpovídajícího pole JSON struktury. Tato hodnota je poté uložena do proměnné lv_contextjwt, která je následně využita v dalších metodách třídy pro autentizované HTTP požadavky.

6. Zpracování výjimek: Celý proces je obalen v bloku TRY-CATCH, který zachytává jakékoli výjimky vyvolané během HTTP komunikace nebo zpracování dat. V případě chyby se loguje příslušná chybová zpráva a mohou

být provedeny další kroky pro zaznamenání chyby pomocí logovací třídy app_log.

4.3.4 Metoda send_data_to_jira

Metoda send_data_to_jira je designována pro odesílání nových dat do Jiry prostřednictvím REST API. Tato metoda demonstruje interakci ABAP systému s externím cloudovým API a použití moderních autentizačních mechanismů pro zabezpečení komunikace.

Kód 11. Metoda send_data_to_jira

```
METHOD send_data_to_jira.
   DATA: lo_http_client TYPE REF TO if_http_client,
         lv_json        TYPE string,
         lv_url         TYPE string,
         lv_status_code TYPE i,
         lv_jwt_token   TYPE string,
         jwt_value      TYPE string.
   TRY.
       DATA(lv_request_body) = mo_request->get_entity( )
->get_string_data( ).
       zcl_jv_json3=>deserialize( EXPORTING json =
lv_request_body CHANGING data = ms_data ).
     CATCH cx_root.
   ENDTRY.
   lv_jwt_token = get_jwt_from_url( ).
   lv_url = |https://rmcloud-prd.releasemanagement.app/
api/1/board/{ ms_data-idboard }/version/{ ms_data-
versionid }/customField|.
   lv_json = '{ "660e5dc73d71fa46b68216ec": "' && ms_data
-transport_id && '" }'.
   TRY.
       cl_http_client=>create_by_url(
         EXPORTING
           url                = lv_url
         IMPORTING
           client             = lo_http_client
         EXCEPTIONS
           argument_not_found = 1
           plugin_not_active  = 2
           internal_error     = 3
           OTHERS             = 4 ).
       lo_http_client->request->set_content_type( '
application/json' ).
       CALL METHOD lo_http_client->request->
set_header_field
         EXPORTING
           name  = 'Authorization'
```

Kód 12. Metoda send_data_to_jira

```
            value = |JWT { lv_jwt_token }|.
        lo_http_client->request->set_method( 'PUT' ).
        lo_http_client->request->set_cdata( lv_json ).
        lo_http_client->send( ).
        lo_http_client->receive( ).
        mo_response->set_status( iv_status = 200
iv_reason_phrase = 'Success.' ).
     CATCH cx_sy_message_in_plugin_mode INTO DATA(
lx_exception).
        MESSAGE ID lx_exception->msgid TYPE lx_exception->
msgty NUMBER lx_exception->msgno WITH lx_exception->msgv1
 lx_exception->msgv2 lx_exception->msgv3 lx_exception->
msgv4 INTO lv_mess.
        BREAK dolezalja.
        app_log->add_bapiret2( /mibcon/cl_wms_utils=>
msg_to_bapiret_line( ) ).
        app_log->save_log( ).
   ENDTRY.
 ENDMETHOD.
```

1. Deserializace vstupních dat: Metoda začíná pokusem o deserializaci JSON dat, která jsou přijata v těle HTTP požadavku. Tato data jsou transformována do interní datové struktury ms_data, což umožňuje jejich další zpracování.

2. Získání JWT: Pro autentizaci požadavků na JIRA API je zapotřebí JWT (JSON Web Token), který je získán z volání metody get_jwt_from_url. Tento krok je klíčový pro zabezpečení a autorizaci následujících HTTP požadavků.

3. Konfigurace HTTP klienta: Po získání JWT se vytvoří instance HTTP klienta pomocí třídy cl_http_client s URL specifikující cílové API endpointy pro aktualizaci nebo vytvoření nového záznamu v JIRA. URL je dynamicky sestaveno z datové struktury ms_data.

4. Nastavení HTTP požadavku: HTTP klient je nakonfigurován pro odeslání požadavku typu 'PUT', což je běžně používaná metoda pro aktualizaci dat na serveru. Do hlavičky požadavku je přidán JWT pro autentizaci a jako typ obsahu je nastaveno 'application/json', což odpovídá formátu odesílaných dat.

5. Serializace a odeslání dat: Data pro odeslání jsou serializována do JSON formátu, který reprezentuje aktualizaci nebo nový záznam a jsou vložena do

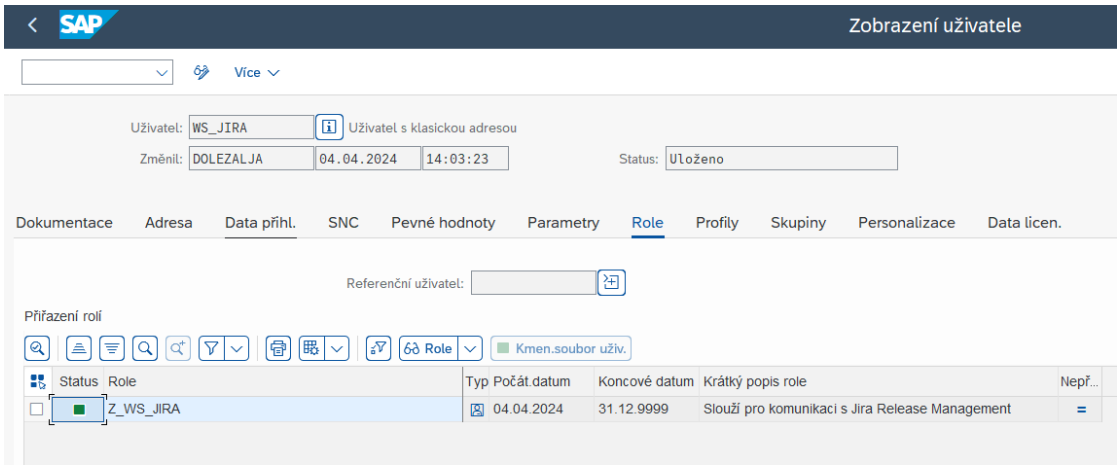
těla požadavku. Po konfiguraci jsou data odeslána na server JIRA.

6. Příjem odpovědi a zpracování výsledků: Po odeslání požadavku se čeká na odpověď serveru. Pokud je odpověď serveru úspěšná (HTTP status 200), metoda nastaví odpovídající stavový kód na 'Success'. V případě chyby (např. serverová chyba nebo problém s autorizací) se zpracuje výjimka a metoda reaguje nastavením chybového stavu a logováním informací o chybě.

7. Zachycení a zpracování výjimek: Operace je chráněna TRY-CATCH, který zachytává jakékoliv výjimky vzniklé během HTTP komunikace nebo zpracování dat. V případě chyby se zaznamenávají detaily chyby a pomocí třídy app_log se přidávají informace do systémového logu pro diagnostiku a audit.

4.4 Vytvoření uživatele JIRA v SAP

Dále bylo potřeba vytvořit uživatele WS_JIRA pro komunikaci se SAPem, uživatel se vytváří v transakci su01, zvolíme vhodné jméno, v záložce Adresa vyplníme Jméno: Jira a Příjmení: Testovací, v záložce Data přihl. dáme typ uživatele: Služba a nastavíme v záložce Role, vytvořenou Roli pro tuhle komunikaci.



Obr. 4.1 Transakce SU01

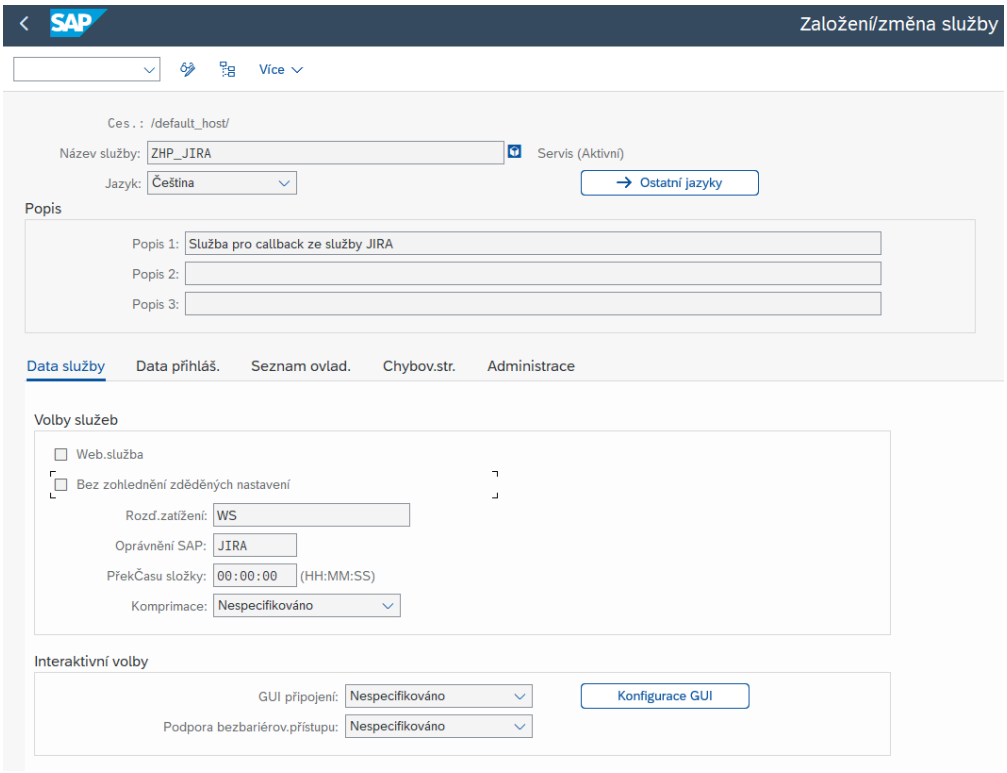
4.5 Vytvoření oprávnění JIRA

V transakci PFCG bylo potřeba vytvořit roli Z_WS_JIRA, která obsahuje oprávnění pro zabezpečení komunikace s Jirou.

4.6 Vytvoření služby v SICF pro RESTful komunikaci

Pro začlenění RESTful webové služby do SAP prostředí je nejprve nutné vytvořit službu v transakci SICF (SAP Internet Communication Framework), která je

zásadní pro konfiguraci a správu HTTP služeb v SAP. Je potřeba se přihlásit do SAP GUI a spustit transakci SICF.



Obr. 4.2 Transakce SICF

Zde byl vytvořen nový podprvek v default host a byl pojmenovaný ZHP_JIRA, do ovladače se nastavila vytvořená třída zcl_jd_jira_ws_handler a přiřadilo se oprávnění SAP na Jira, které bylo taky vytvořené. Dále bylo nastaveno jako bezpečnostní prvek na SSL a autentizaci na Uživatel internetu. Po dokončení konfigurace se služba aktivovala. To umožní službě přijímat příchozí požadavky. V okamžiku, kdy se služba otestuje, získá se tím potřebná URL adresa.

5 Praktické kroky v systému Jira

Pro zajištění bezpečného připojení mezi Jira a SAP CTS je prvním krokem vytvoření API tokenu. Tento token je nezbytný pro autentizaci a autorizaci požadavků mezi systémy. V Jira se token generuje v nastaveních uživatelského účtu pod sekci API tokeny.

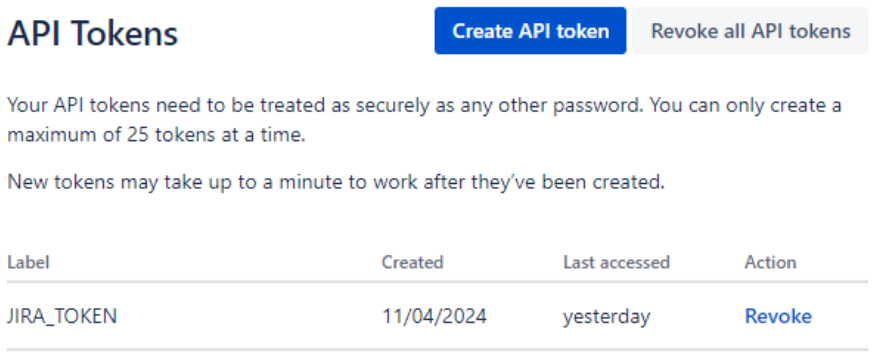
Dále je potřeba v Jira vytvořit nový board, který bude sloužit k sledování a správě release cyklů. Board může být konfigurován jako Scrum nebo Kanban podle metodiky, která je ve společnosti preferována. Board je základem pro organizaci úkolů a sprintů.

Pro efektivní řízení procesů v rámci release managementu je klíčové nastavit workflow, který odpovídá procesům uvolňování v SAP CTS. Toto workflow musí zahrnovat všechny potřebné kroky, jako jsou revize kódu, testování, schválení a samotné uvolnění. Každý krok může mít definované podmínky pro přechod na další fázi.

S využitím funkcí automatizace v Jira lze nastavit pravidla pro automatizované spouštění úloh, například automatické přiřazení úkolů, odesílání upozornění nebo aktualizace statusů. Automatizace pomáhá zvýšit efektivitu procesů a snížit manuální zásahy.

Konečným krokem je nastavení webhooku, který umožňuje Jira systému komunikovat s externími aplikacemi v reálném čase. Webhook je konfigurován tak, aby odesílal notifikace do SAP CTS při událostech definovaných v Jira, například po schválení release. Tímto způsobem je zajištěno, že informace mezi systémy jsou synchronizovány a aktualizovány automaticky.

5.1 Vytvoření Api Tokenu

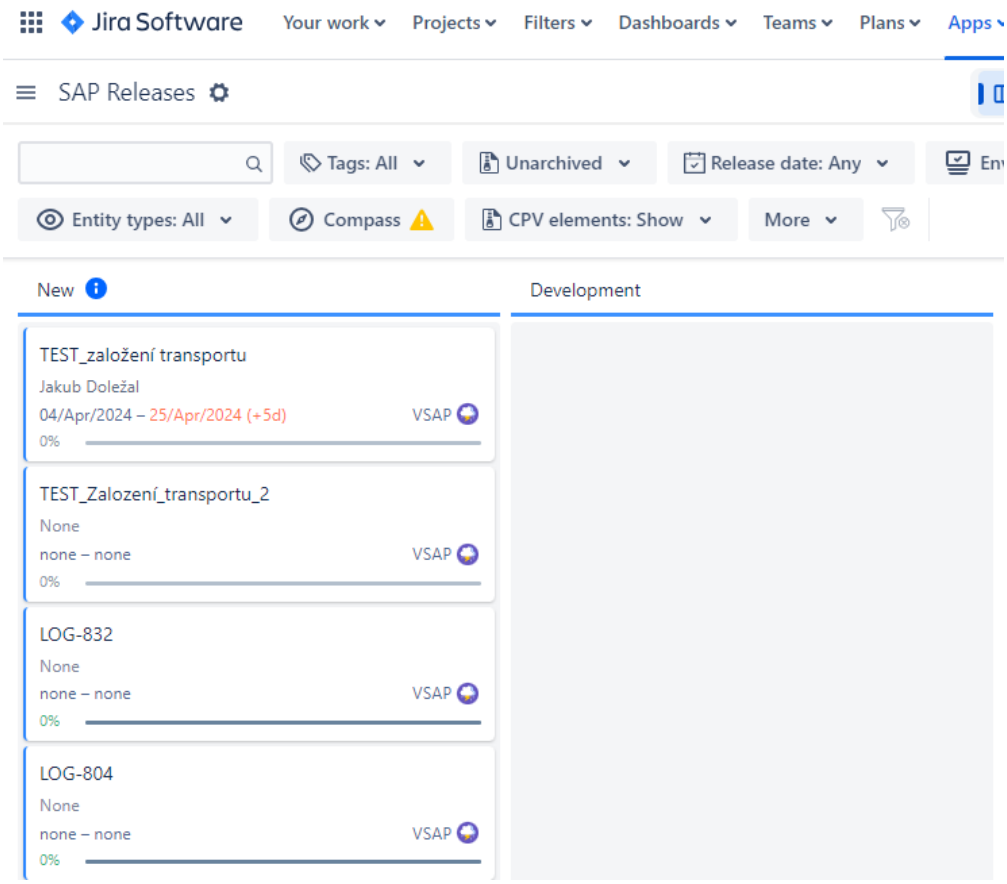


Obr. 5.1 Vytvoření Api Tokenu

Ná stránce Atlassian Account v záložce security se vytváří API token, který je důležitý při vytváření JWT v moment, kdy si ho dožaduje SAP v metodě get_jwt_from_url.

5.2 Vytvoření Boardu

V testovací instanci Jira byl vytvořený nový projekt. Projekt může být specificky nastaven pro potřeby správy vydání. V rámci projektu byl vytvořen board, který slouží k vizualizaci procesu správy vydání. Board je typu Kanban a sloupce jsou nastaveny, aby odpovídali různým fázím vydání, jako jsou 'New', 'Development', 'To Test', 'To Deploy' a 'Deployed' a následně se definovalo workflow, které určuje, kam se mají jednotlivé verze dále posouvat.

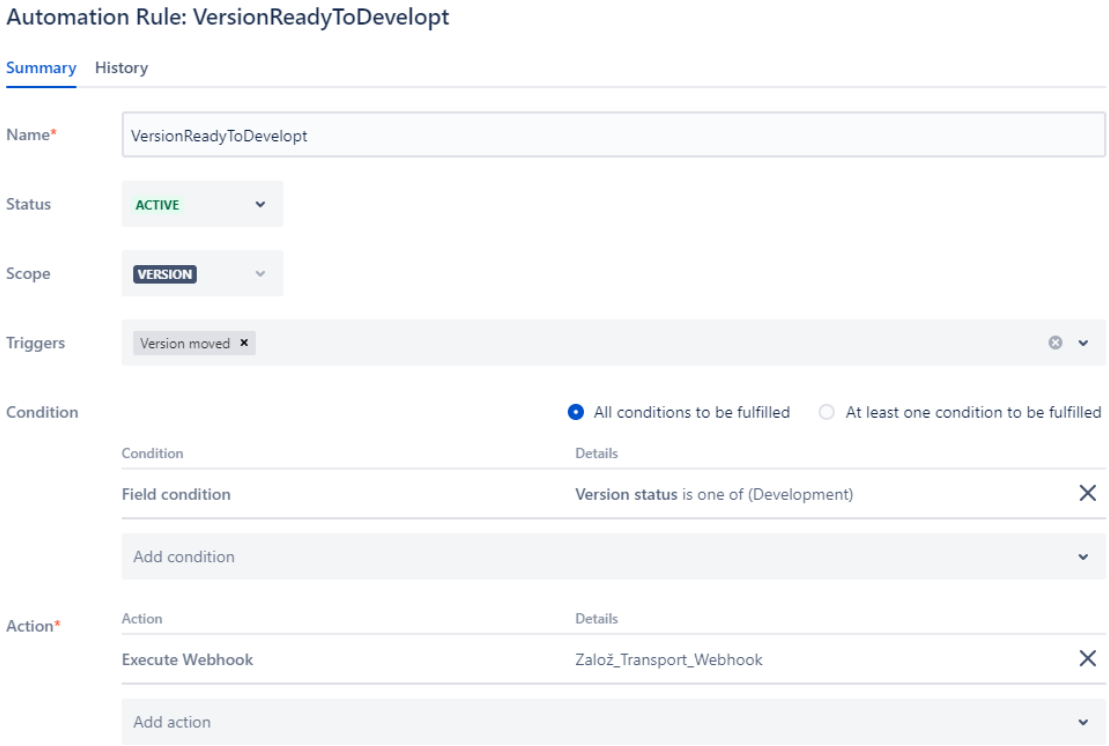


Obr. 5.2 Nastavení postupu verzí

5.3 Vytvoření Automatizace

Při vytváření automatizace bude příslušné jméno nastaveno, status bude nastaven na 'Active', dále budou nastaveny Triggers, neboli spouštěče, které vyvolají požadovanou automatizaci. Bude zvoleno 'Version moved', což v praxi znamená, že při

přetažení verze se spustí automatizace. Bude zaškrtnuto, aby musely být splněny všechny podmínky a následně bude zvolena podmínka, kam je potřeba verzi přetáhnout. Po kliknutí na 'add condition' bude kliknuto na 'Field condition', condition bude vyplněna na 'one of' a jako value bude zvoleno 'Development'. Takže při každém přetažení verze do Development se automatizace spustí. Do kolonky 'Action' bude vybráno 'Execute Webhook' a bude pojmenováno. Následným proklikem na Webhook se dostane do bližšího nastavení Webhooku.

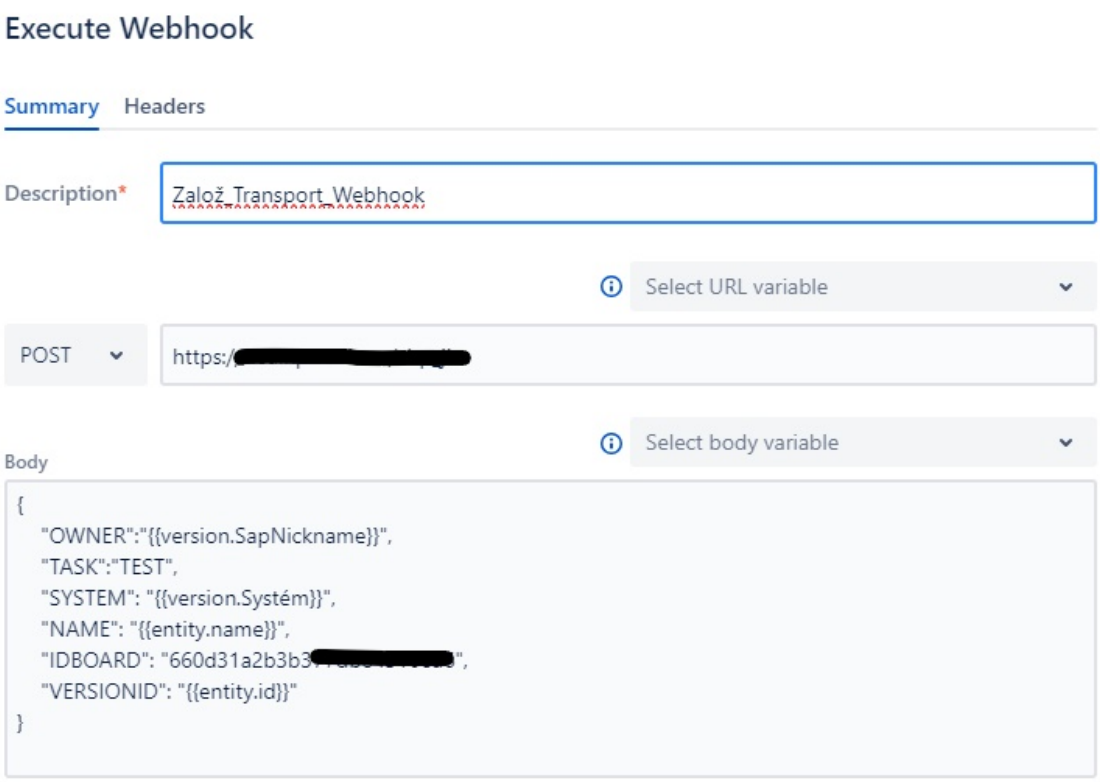


Obr. 5.3 Tvorba automatizace

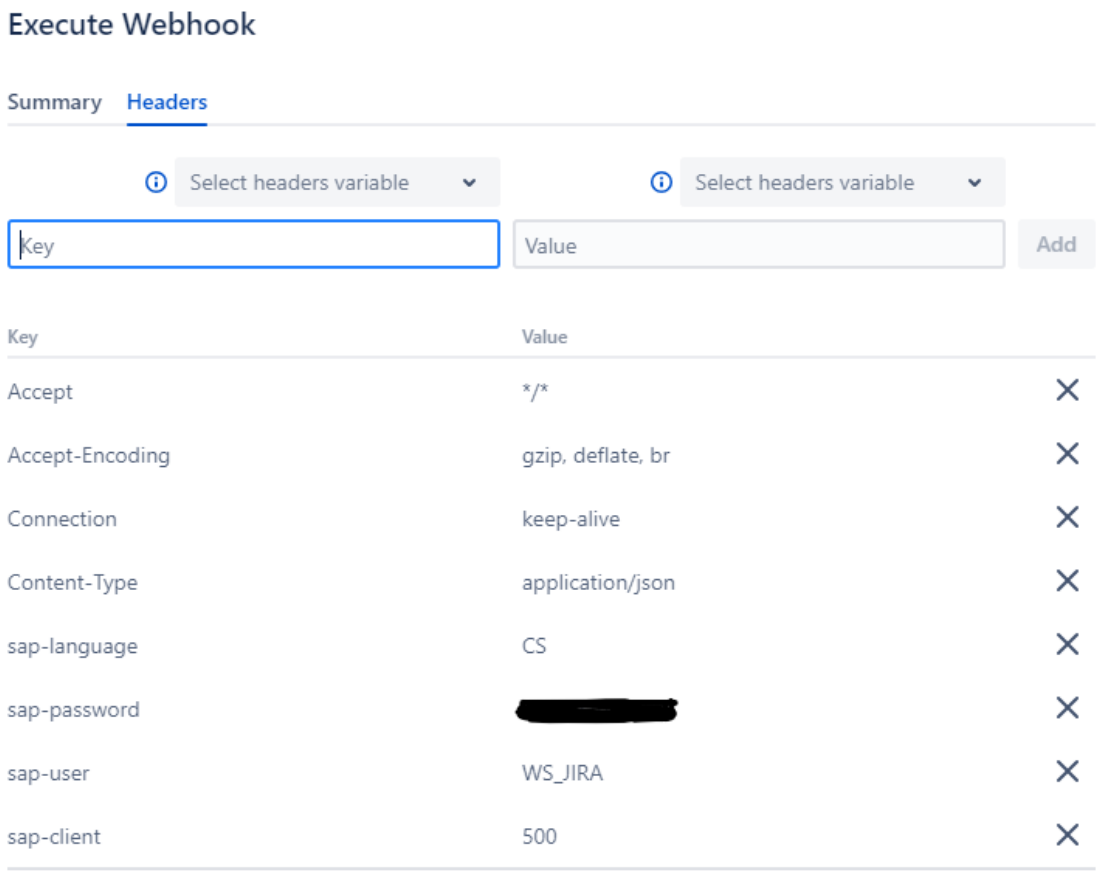
5.3.1 Nastavení Webhooku

Při nastavování Webhooku musí být vyplněn název, musí být nastavena metoda REST API, kterou chceme využít. URL adresa musí být vyplněna a do body ve tvaru JSON musí být napsána data, která mají být do SAPu poslána.

V záložce Headers je důležité, aby byly vyplněny parametry pro autorizaci do SAPu. Do key musí být vloženy názvy: sap-user, sap-password, sap-client a sap-language a do value musí být vloženy požadované hodnoty.



Obr. 5.4 Nastavení Webhook



Obr. 5.5 Nastavení Headers

6 Uživatelská dokumentace

Základním předpokladem je, že verze obsahuje maximalne jeden transport od každého druhu (WB, DICT, CU).

Popis procesu: Vyvojar zaklada verzi v Jira ve stavu NEW.

- Ideálně ji pojmenuje, zadává Release Managera, systém a požadované typy transportu (WB, DICT, CU).
- Při přechodu ze stavu NEW do stavu DEVELOPMENT se spouští automatizace, která na základě systému zakládá požadované transporty v daném systému. Název transportu se přebírá z názvu verze. Do atributu transportu se pak uloží id boardu a verze.
- Po založení transportu pak odesíláme Release Managementu číslo transportu do odpovídajícího pole ve verzi, podle typu transportu.

6.1 Vyhodnocení přínosů propojení těchto dvou systémů.

Integrace využívá RESTful webové služby k propojení SAP CTS a Jira. To znamená, že aktualizace a tok dat probíhají automaticky mezi oběma systémy bez manuálního zásahu, což šetří čas a zabraňuje případným chybám způsobené lidským faktorem.

Propojením transportů SAP CTS s úkoly Jira se vytvoří přímé spojení mezi změnami softwaru a požadavky na transport. To zlepšuje sledovatelnost, umožňuje členům týmu snadno vidět, které úkoly Jira jsou spojeny s konkrétními transporty a naopak. To může být klíčové během auditů nebo přezkumů.

ZÁVĚR

Závěr mojí bakalářské práce se věnuje zhodnocení integrace systémů SAP CTS a Jira Release Management pluginu. Cílem práce bylo vytvořit automatizované řešení, které by při změně stavu verze v Jira z NEW na DEVELOPMENT automaticky generovalo transportní požadavky v SAP CTS. Tento přístup byl implementován prostřednictvím dvou tříd v ABAPu, které interagují s Jira REST API a umožňuje výrazné zefektivnění pracovních procesů vývojářů a správců systému SAP.

Výsledky implementace ukázaly, že integrace funguje spolehlivě a splňuje všechny stanovené cíle. Bylo potvrzeno, že automatizace procesů mezi Jirou a SAP CTS může značně přispět k efektivnější správě verzí a transportních požadavků. Tato bakalářská práce představuje důležitý krok k plné automatizaci procesů mezi Jira a SAP CTS a nabízí robustní základ pro další vývoj v této oblasti.

Teoretických možností využití propojení těchto dvou systémů do budoucna je mnoho, například pro lepší viditelnost stavu projektu pro všechny zúčastněné týmy by šlo provést takové automatizované sledování změn, že když vývojář dokončí úpravy v SAP a změna je schválena k transportu, automaticky se aktualizuje příslušný JIRA ticket na stav "Ready for Testing".

SEZNAM POUŽITÉ LITERATURY

[1] Sap. [online]. [cit.2024-02-17]. Dostupný z: https://www.sap.com/cz/about/what-is-sap.html.

[2] SapLogo. [online]. [cit. 2024-02-17]. Dostupné z: https://logos-download.com/wp-content/uploads/2016/08/SAP_logo.png.

[3] What is sap. [online]. [cit. 2024-02-17]. Dostupné z: https://skillstek.com/what-is-sap/.

[4] Simomonds Jon, fifty years and counting. [online] [cit.2024-02-18]. Dostupný z: https://erp.today/fifty-years-and-counting/.

[5] Armin Kösegi, Rainer Nerding. SAP Change and Transport Management. SAP Press, 2009. ISBN 978-1-59229-529-6.

[6] Workbench Organizer. [online]. [cit.2024-02-19]. Dostupný z: https://help.sap.com/doc/saphelp_nw74/7.4.16/enUS/4e/00f3053198001de1000000a42189c/content.htm?no_cache=true

[7] Transport Organizer. [online]. [cit.2024-02-25]. Dostupný z: https://help.sap.com/docs/SOFTWARE_LOGISTICS_TOOLSET_CTS_PLUG-IN/05c12df5b54849c49940a14bc089d8b4/5738dd924eb711d182bf0000e829fbfe.html.

[8] Transport Management System [online]. [cit.2024-02-25]. Dostupný z: https://help.sap.com/doc/saphelp_autoid2007/2007/en-US/44/b4a0137acc11d1899e0000e829fbbd/frameset.htm.

[9] What is the difference between tp and r3 trans [online]. [cit.2024-03-03]. Dostupný z: https://community.sap.com/t5/sap-for-consumer-products-blogs/what-is-the-difference-between-tp-and-r3-trans/ba-p/13291860.

[10] UnderstandingSapNetwaverGateway [online]. [cit.2024-03-15]. Dostupný z: https://itpfed.com/understanding-sap-netweaver-gateway/.

[11] SapApiManagement [online]. [cit.2024-03-15]. Dostupný z: https://help.sap.com/docs/sap-api-management/sap-api-management/what-is-api-management?version=Cloud.

[12] SapCustomRWS [online]. [cit.2024-3-15]. Dostupný z: https://community.sap.com/t5/enterprise-resource-planning-blogs-by-members/creation-of-rest-api-get-post-method-call/ba-p/13534724.

[13] SAP Directory [online]. [cit.2024-03-14]. Dostupný z: https://help.sap.com/doc/saphelp_nw73ehp1/7.31.19/en-us/c4/6045377b52253de10000009b38f889/frameset.htm.

[14] SAP Transport Layers and Transport Routes [online]. [cit.2024-03-26]. Dostupný z: https://help.sap.com/docs/ABAP_PLATFORM_NEW/4a368c163b08418890a406d413933ba7/5738de0c4eb711d182bf0000e829fbfe.html.

[15] TypesTR [online]. [cit.2024-03-25]. Dostupný z: https://community.sap.com/t5/technology-blogs-by-members/all-about-transport-request/ba-p/13287831.

[16] MARKGRAF, Joe a Alessandro BANZER. SAP Systém Security Guide. SAP Press, 2018. ISBN 978-1-7932-1481-5.

[17] Abap [online]. [cit.20024-03-25]. Dostupný z: https://learning.sap-press.com/abap.

[18] JiraLogo [online]. [cit.2024-04-05]. Dostupný z:https://www.atlassian.com/download.

[19] ReleaseManagementJira [online]. [cit.2024-04-05]. Dostupný z: https://community.atlassian.com/t5/App-Central/Release-Management-is-the-next-big-thing-for-Atlassian-Jira/ba-p/1348276.

[20] JiraReleaseManagement [online]. [cit.2024-04-05]. Dostupný z: https://releasemanagement.atlassian.net/wiki/spaces/RMC/overview?homepageId=92078213.

[21] RAPIJIRA [online]. [cit.2024-04-05]. Dostupný z: https://developer.atlassian.com/server/jira/platform/rest-apis/

[22] JiraWebHook [online]. [cit.2024-04-08]. Dostupný z: https://developer.atlassian.com/server/jira/platform/webhooks/.

[23] JSON web token [online]. [cit.2024-04-08]. Dostupný z: https://jwt.io/introduction.

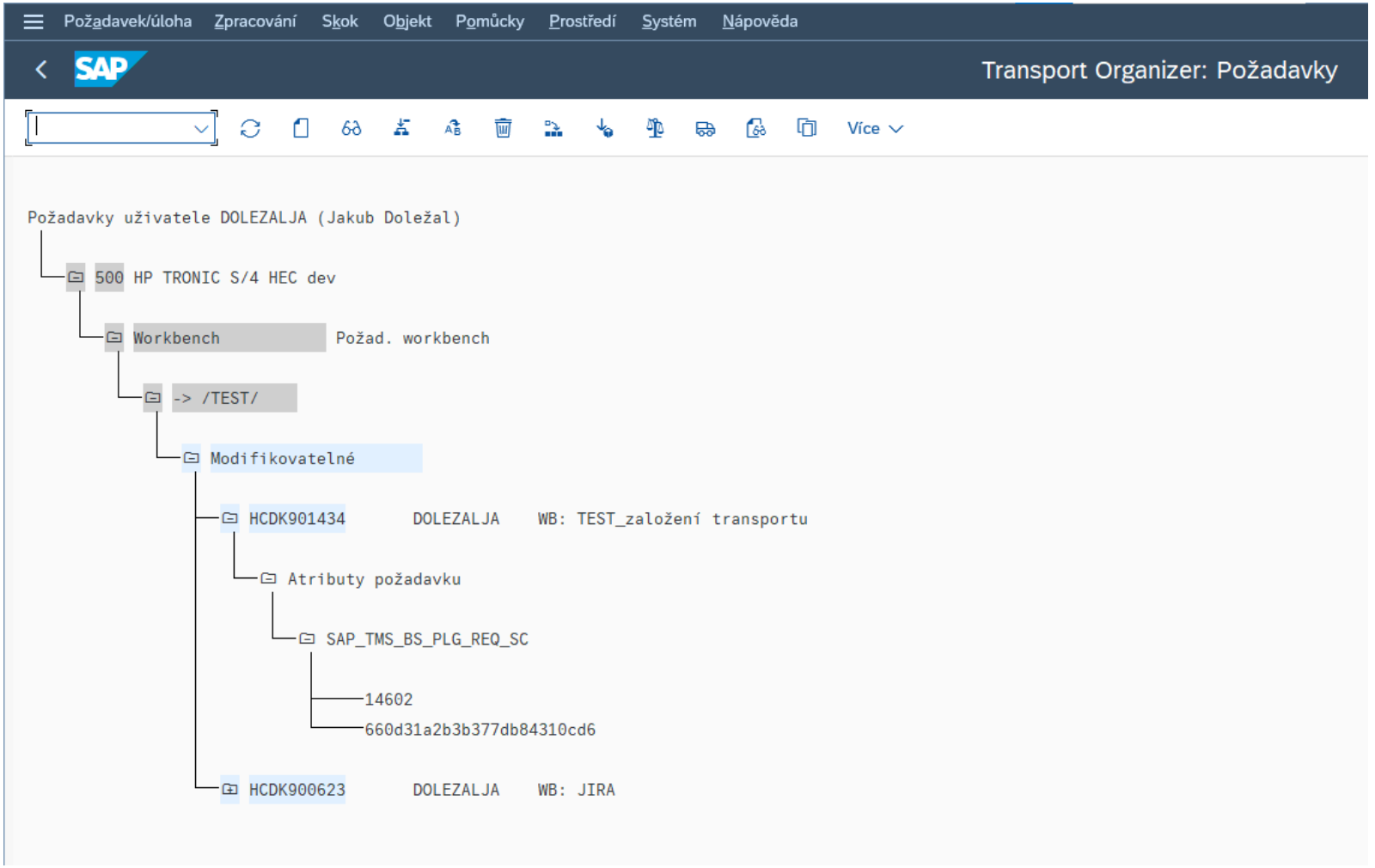
SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

SAP	System Analysis Program Development
CTS	Change and Transport System
CTO	Change and Transport Organizer
TMS	Transport Management System
TP	transport control program
ABAP	Advanced Business Application Programming
REST	Representational State Transfer
API	Application Programming Interface
HTTP	Hypertext Transfer Protocol
SSO	Single Sign-On
URL	Uniform Resource Locator
JWT	JSON Web Token
JSON	JavaScript Object Notation
CSRF	ross-Site Request Forgery
CRUD	Create, Read, Update, and Delete
GUI	Graphical user interface
WB	Workbench
DICT	Dictionary
CU	Customizing

SEZNAM OBRÁZKŮ

SEZNAM PŘÍLOH

PŘÍLOHA P I. ZALOŽENÝ TRANSPORT



PŘÍLOHA P II. UPDATE V JIRE