VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

INFORMAČNÍ SYSTÉM PRO ODVÁDĚNÍ VÝROBY A MATERIÁLU

INFORMATION SYSTEM FOR PRODUCTION AND MATERIAL MANAGEMENT

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. MIROSLAVA ŠKUTOVÁ**
AUTHOR

VEDOUCÍ PRÁCE — **RNDr. MAREK RYCHLÝ, Ph.D.**
SUPERVISOR

BRNO 2024

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Zadání diplomové práce

152626

Ústav:	Ústav informačních systémů (UIFS)
Studentka:	**Škutová Miroslava, Bc.**
Program:	Informační technologie a umělá inteligence
Specializace:	Informační systémy a databáze
Název:	**Informační systém pro odvádění výroby a materiálu**
Kategorie:	Informační systémy
Akademický rok:	2023/24

Zadání:

1. Seznamte se s uživatelskými požadavky na systém pro podporu odvádění výroby a materiálu rozšiřující stávající ERP systém HELIOS. Zaměřte se zejména na možnosti měření a zjištění efektivity výroby a analyzujte dostupná řešení v této oblasti. Prozkoumejte aplikační a databázové rozhraní systému HELIOS pro související agendy a jejich data.
2. Navrhněte vlastní informační systém pro podporu odvádění výroby a materiálu, který bude napojen na datové či aplikační rozhraní systému HELIOS. Navržený systém bude umožňovat zadávání a analýzy dat dle požadavků (např. měření efektivity strojů a výrobních oblastí).
3. Po konzultaci s vedoucím navržený systém implementujte. Navrhněte a implementujte také akceptační, integrační, případně i jednotkové testy systému či jeho komponent, a pomocí těchto testů výsledný systém ověřte.
4. Systém vyzkoušejte v testovacím prostředí či při zkušebním provozu, zhodnoťte výsledky a navrhněte možná rozšíření.

Literatura:

- T. Zheng, M. Ardolino, A. Bacchetti, M. Perona. The applications of Industry 4.0 technologies in manufacturing context: a systematic literature review, International Journal of Production Research, 59:6, 1922-1954, 2021. https://doi.org/10.1080/00207543.2020.1824085
- J. Björkdahl. Strategies for Digitalization in Manufacturing Firms. California Management Review, 62(4), 17–36, 2020. https://doi.org/10.1177/0008125620920349
- J. T. Arnold. Introduction to materials management. 8th edition, 2021. ISBN 9780137517299.

Při obhajobě semestrální části projektu je požadováno:
Body 1 a 2.

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

Vedoucí práce:	**Rychlý Marek, RNDr., 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

Cílem této práce je provést digitalizaci klíčových aspektů výrobních a personálních procesů ve strojírenské firmě a současně optimalizovat využití existujícího informačního systému HELIOS iNuvio. Vyvinutý software, založený na webových technologiích PHP s využitím frameworku Symfony, React a REST, obsahuje moduly, které nejen nahrazují terminálovou aplikaci pro odvádění výroby a správu materiálů, ale také umožňují digitalizaci a zjednodušení dalších firemních procesů, včetně podepisování důležitých dokumentů a správy strojů. Analytický modul tohoto řešení poskytuje pomocí vizualizačního nástroje Grafana vedení firmy podrobné informace o výkonnosti zaměstnanců, efektivitě výrobních procesů a finančních aspektech výroby. Docházkový modul nahrazuje manuální správu docházky zaměstnanců. Komplexní řešení zvyšuje celkovou efektivitu podnikových procesů a zlepšuje uživatelskou zkušenost, aniž by vyžadovala další investice do hardwaru nebo systémů.

Abstract

The aim of this work is to digitalise key aspects of the production and personnel processes in an engineering company and at the same time to optimise the use of the existing HELIOS iNuvio information system. The software developed, based on PHP web technologies using the Symfony, React and REST frameworks, includes modules that not only replace the terminal application for production control and materials management, but also enable the digitalisation and simplification of other company processes, including the signing of important documents and machine management. The solution's analytics module provides management with detailed information on employee performance, production process efficiency and financial aspects of production using the Grafana visualisation tool. The attendance module replaces manual attendance management. This complex solution increases overall efficiency of business processes and improves the user experience without requiring additional investment in hardware or systems.

Klíčová slova

Informační systém, HELIOS, iNuvio, digitalizace, PHP, React, Symfony, Grafana, REST, výroba

Keywords

Information system, HELIOS, iNuvio, digitalization, PHP, React, Symfony, Grafana, REST, production

Citace

ŠKUTOVÁ, Miroslava. *Informační systém pro odvádění výroby a materiálu.* Brno, 2024. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce RNDr. Marek Rychlý, Ph.D.

Informační systém pro odvádění výroby a materiálu

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracovala samostatně pod vedením pana RNDr. Marka Rychlého, PhD. Uvedla jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpala.

.....
Miroslava Škutová
5. května 2024

Poděkování

Děkuji panu RNDr. Marku Rychlému, PhD., za odborné vedení a vstřícnost během tvorby diplomové práce. Rovněž bych ráda vyjádřila vděk panu Vladimíru Valáškovi za příležitost nahlédnout do výroby a za možnost získat praktické dovednosti. Dále děkuji i panu Janu Skybovi za asistenci při integraci systému HELIOS a za sdílení svých zkušeností s výrobními procesy.

Obsah

Seznam obrázků

Kapitola 1

Úvod

Informační systémy jsou nedílnou součástí moderní a efektivní výroby. Jejich použití v podnicích přináší zlepšení takřka ve všech oblastech. Umožňují vést přehlednou evidenci materiálů i výrobků na skladě, díky čemuž je možné optimalizovat plán výroby. Automatizované plánování i sledování firemních procesů snižuje riziko lidských chyb a zároveň umožňuje rychlejší reakci na nehody, v některých případech dokonce i jejich předcházení. Do správy a vedení podniku přináší informační systémy další úroveň transparentnosti, například v podobě záznamů o údržbě strojů. Všechny tyto aspekty se pojí se snížením nákladů, větší kontrolou nad výrobními procesy a zejména větší konkurenceschopností.

Ačkoliv přínos informačních systémů v oblasti výroby nelze nijak zpochybnit, kamenem úrazu bývá jejich samotné použití. Často se můžeme setkat s polovičním krokem firmy směrem k automatizaci, kdy sice vlastní dobrý informační systém a investovala nemalé peníze do jeho nasazení, a přestože obsahuje prvky, které by výrobu zefektivnily, jeho použití je natolik obtížné nebo nepohodlné, že se část procesů realizuje v původní podobě před automatizací.

Diplomová práce se zabývá konkrétním případem plně nevyužitého potenciálu informačního systému ve strojírenské firmě. Klade si za cíl nejen zefektivnit samotný proces výroby, ale také zpříjemnit použití systému zaměstnancům. Řešení je navrhováno tak, aby využilo co největšího možného počtu zdrojů ve firmě, bez nutnosti dalších investic do modulů, jiného systému nebo technického vybavení. Zaměřuje na oblasti, kde automatizace zaostává, nebo je k její implementaci přistupováno nedokonale a je zaměstnanci hodnocena jako nepohodlná.

V rámci diplomové práce je navržen a implementován systém pro podporu odvádění výroby a správy materiálů. Díky systému mohou zaměstnanci odvádět výrobu pomocí terminálů na hale, odepisovat materiál, naskladňovat zboží a mimo jiné například evidovat zmetky. Součástí je i modul pro evidenci docházky zaměstnanců. Pro vedení firmy je k dispozici modul analýzy dat, který zobrazuje informace o výdělečnosti a zmetkovitosti jednotlivých výrobků, efektivitě jednotlivých zaměstnanců a v neposlední řadě zobrazuje aktuální finanční situaci ve firmě. Hlavní předpokládané využití systému je zejména podpora výroby na terminálech, ale díky využití webových technologií lze tento systém vedením firmy používat také na počítačích nebo telefonech.

Nový systém je navržen tak, aby znovu neimplementoval funkční firemní procesy, pouze vylepšoval jejich provedení. Pro odvádění výroby, správu materiálů a analýzu dat využívá informační systém HELIOS iNuvio a nahrazuje existující nedokonalou terminálovou aplikaci. Evidence docházky pak plně automatizuje a nahrazuje papírové píchací hodiny.

Kapitola 2

Analýza situace a požadavků

V této kapitole bude detailně rozebrána současná situace v zadavatelské firmě. Bude zhodnocen aktuální výrobní proces, řešení, která jsou ve firmě nasazena, jejich nedostatky z technického hlediska i z pohledu samotné firmy a zaměstnanců

Zadavatelská firma je středně velký podnik o zhruba padesáti zaměstnancích. Zaměřuje se na lisování kovů, komplexní zpracování a tváření plechů, obrábění, laserové řezání a výrobu a prodej pantů. Pro automatizaci procesů využívá informační systém HELIOS iNuvio. Tento systém je podrobněji popsán v kapitole 4. Ve firmě jsou nyní pro zaměstnance na dílnách k dispozici tři průmyslové terminály s dotykovým displejem.

2.1 Autentizace a autorizace uživatelů

Současný stav Jedním z hlavních problémů stávající aplikace na odvádění výroby je absence zabezpečení. Zaměstnanci se přihlašují pouze čipem, nebo velmi krátkým osobním číslem (jedno až dvou místné číslo). Zaměstnanec se tak může přihlásit pod jménem kohokoliv jiného, včetně svých nadřízených.

Očekávaný stav Firma požaduje implementaci následujících typů přístupů k systému:

- Základní přístup pomocí čipového kódu, bez hesla (pouze z terminálů)
- Privilegovaný přístup s heslem (kdekoliv)

Zároveň je potřeba rozdělit uživatele do několika skupin:

- **Typ 1**: Uživatel má přístup pouze z terminálů na dílně, nemůže se přihlásit pomocí hesla a získat vyšší práva
- **Typ 2**: Uživatel se může verifikovat heslem a tím získat vyšší práva
- **Typ 3**: Uživatel se musí verifikovat heslem, a to z jakéhokoliv stroje

Pokud zaměstnanec přistupuje k systému z jednoho z výrobních terminálů, stačí pouze přiložit čip ke čtečce, aby mu byl umožněn základní přístup. Pokud je zaměstnanec uživatelem typu 2, v jakékoliv fázi základního přístupu se může dále verifikovat heslem, čímž zpřístupní další funkce systému. Naopak při přístupu z jiného zařízení bude systém už při prvotním přihlášení vyžadovat heslo. Zaměstnanec typu 1 má k systému přístup pouze z terminálů, z jakéhokoliv jiného stroje bude přístup zamítnut, a to i v případě zadání

správného hesla. Zaměstnanec typu 3 se musí vždy verifikovat heslem, a to zejména z toho důvodu, aby nebylo možné provádět ani základní operace jménem nadřízených, pokud by bylo jakýmkoliv způsobem zjištěno jejich číslo čipu.

2.2 Evidence docházky

Současný stav V oblasti evidence docházky nebyla dosud zavedena žádná automatizace. Zaměstnanci evidují své příchody a odchody pomocí píchacích hodin při příchodu do práce, poznámky a případná odůvodnění píšou ručně k datu příchodu a odchodu. Na konci měsíce se všechny příchody a odchody manuálně procházejí, zaokrouhlují se hodiny a počítá se odpracovaná doba a adekvátní mzda zaměstnance.

Problémy Vyhodnocení jednoho měsíce nyní zabere zhruba osm hodin čistého času a dochází k chybám při manuálních výpočtech.

Očekávaný stav Evidence docházky by měla být plně automatizována a digitalizována. Zaměstnanci by měli evidovat své příchody a odchody pomocí naskenování čipu na terminálech ve výrobních halách. Očekávají se tyto funkcionality:

- Automatická předvolba příchodu/odchodu s možností úpravy
- Výpočet odpracované doby v aktuálním dni v reálném čase
- Zobrazení pracovního kalendáře, kde v rámci dne budou vypsány všechny příchody, odchody a počet odpracovaných hodin
- Možnost přidat poznámku ke konkrétnimu dni
- Konfigurovatelné zaokrouhlení příchodů a odchodů
- Možnost přednastavení svátků a firemních dovolených
- Export měsíčních výpisů podle předem zadaného formátu

2.3 Odvádění výroby

Současný stav S podporou odvádění výroby se zadavatelská firma potýká již delší dobu. Na zakázku byla vyvinuta terminálová aplikace, která ale nesplňuje očekávání. Další spolupráce mezi firmou, která zakázku realizovala, a klientem (v kontextu diplomové práce zadavatelskou firmou) už neprobíhá, a tak neexistuje ani žádná podpora, údržba nebo další vývoj dodané aplikace. V současné době tak odvádění výroby probíhá napůl ručně a napůl pomocí již změněné aplikace. Výrobní příkazy se tisknou, po prvním odvedeném kusu musí kontrolor kvality výrobek podpisem schválit. V terminálové aplikaci pak na konci výroby zadají zaměstnanci celkový odpracovaný čas a počet vyrobených kusů. Někteří zaměstnanci také musí provádět kontrolu přidělených strojů, která je v současné situaci řešena podpisy na kontrolních listech v papírové podobě.

Problémy Při tištěných výrobních příkazech se občas stává, že se ztratí anebo poničí. V terminálové aplikaci je pak řada chyb, na které si stěžují jak zaměstnanci, tak vedení, a sice:

- **Chybovost**: Zaměstnanci si stěžují na vysokou chybovost aplikace, kdy nelze dokončit výrobní operace, nebo se tyto operace dokončí, ale s chybovou hláškou, která vzápětí zmizí anebo se zdánlivě vůbec netýká prováděné operace. To způsobuje problémy, protože zaměstnanci neví, jestli je to chyba aplikace, nebo jejich, a jestli byla operace skutečně dokončena.
- **Neintuivní rozhraní**: Aplikace se velmi těžce ovládá, rozhraní je nekonzistentní a neintuitivní. Chybové hlášky často problikávají tak rychle, že je zaměstnanci nestíhají přečíst.
- **Bezpečnost**: Chybí autentizace a i následná autorizace zaměstnanců, kteří se tak mohou přihlásit například jako správce systému, pokud znají jeho identifikační číslo. Tento problém byl podrobně popsán v sekci 2.1.
- **Absence datového výstupu**: Aplikace vůbec neřeší propojení s daty dostupnými od strojů a zároveň neposkytuje žádné rozhraní pro analýzu. Kvůli už zmíněné chybovosti zaměstnanci často nepíšou pravdivý čas výroby, chybí jeho automatické sledování, a to znemožňuje správné odhady ceny a další analýzu.
- **Nemožnost úpravy**: Vývoj aplikace se zastavil. Jedná se o uzavřený a nedostupný kód, takže ani správci IT nemají možnost cokoliv upravit nebo přizpůsobit.
- **Nepřenositelnost na jinou platformu**: Aplikace je spuštěna pouze na terminálech a je jim přizpůsobená. Není možné ji použít například z telefonu anebo z počítače.

Očekávaný stav Firma si přeje zcela nahradit současnou terminálovou aplikaci. Nová aplikace by měla být přístupná ze všech zařízení ve firmě, ale s ohledem na bezpečnost. Je nutné přidat novou vrstvu autentizace a autorizace zaměstnanců, aby nemohlo dojít ke zneužití systému, ale zároveň zachovat jednoduchost a rychlost přihlašování čipem. Aplikace bude v omezeném rozsahu přístupná po přihlášení pomocí čipu (identifikačního čísla). Pro vyšší přístup a autorizované operace bude systém vyžadovat heslo, stejně tak, pokud se k aplikaci někdo pokusí přistupovat z jiných strojů než z terminálů. Nová aplikace pro podporu odvádění výroby by měla umožňovat:

- Vyhledávání výrobních příkazů jak pomocí čtečky čárových kódů, tak i manuálním zadáním vyhledávacího řetězce. Tyto případy budou rozlišeny a budou vyhledávat v jiných atributech. Pomocí čtečky čárových kódů se ve většině případů skenuje čárový kód, kdežto zaměstnanci vyhledávají podle názvu, firemního označení a dalších označení výrobků.
- Sledovat čas výrobní operace s možností přerušení aktivní výroby. Modul musí spolupracovat s modulem docházky, a sice po odchodu zaměstnance z práce se prováděná výroba automaticky pozastaví.
- Evidovat odvedené kusy výrobků včetně zmetků v systému HELIOS
- Automaticky generovat doklad o přijetí zboží na sklad po dokončení celého procesu výroby

- Automaticky počítat množství spotřebovaného materiálu při výrobě a odepisovat ho ze skladu
- Umožnit zobrazení PDF výkresů a balicích předpisů, pokud jsou pro daný výrobek k dispozici
- Přihlášenému uživateli zobrazit jeho poslední vyhledávání
- Přihlášenému uživateli zobrazit rozpracované výrobní příkazy
- Umožnit uživatelům evidovat kontrolu strojů

Zároveň je nutné modernizovat uživatelské rozhraní, ale s ohledem na použitelnost aplikace ve výrobní praxi a ne na úkor funkčnosti. Je nutné upřednostnit přehlednost před moderním a vizuálně pěkným rozhraním podle posledních trendů.

2.4 Analýza dat

Současný stav Firma má pouze málo prostředků pro analýzu dat. Informační systém HELIOS sice obsahuje prostředky pro výpočet celkové ceny produktu, ale neposkytuje možnost větší analýzy. Zaměstnanci tak musí ručně vyhledat konkrétní výrobek a konkrétní data. Tato data jsou zároveň nespolehlivá, protože zaměstnanci kvůli špatné terminálové aplikaci (viz sekce 2.3) neuvádějí pravdivý čas výroby, nebo omylem zadají špatné údaje.

Očekávaný stav Firma požaduje automatické zpracování dostupných dat. Měla by být vytvořena přehledná nástěnka, která zobrazí následující informace:

- Celkovou výdělečnost výrobků, správnost jejich nacenění
- Zmetkovitost výrobků
- Přehled stavu materiálů a výrobků na skladě s ohledem na probíhající výrobní operace. Musí být zobrazen vývoj stavu skladu a chybějící materiál nutný pro dokončení právě rozpracovaných výrovních operací.
- Využití materiálů, který materiál z dané skupiny se nejvíce používá
- Efektivitu jednotlivých zaměstnanců s důrazem na velké odchylky. Cílem není přesné sledování jednotlivých zaměstnanců, ale odhalování například špatného používání systému nebo dlouhodobě odlišných časů výroby od průměru.

2.5 Sklady a materiály

Současný stav Informace o materiálech a výrobcích jsou vedeny v informačním systému. Naskladnění probíhá pomocí manuálního vytváření příjemek v systému HELIOS pracovníky v expedici. Odepisování materiálu je prováděno buď taktéž v systému HELIOS, nebo pomocí stávající terminálové aplikace pro odvádění výroby, která materiál vyhledává pomocí přesného výrobního čísla (čárový kód).

Problémy Materiál je možné odepisovat na terminálech pouze pomocí přesného čárového kódu. Chybí možnost vyhledávání podle parametrů a současné řešení neumožňuje odepisovat materiál, který nemá přiřazené výrobní číslo. Zároveň je možné materiál pouze odepisovat, nikoliv naskladňovat.

Očekávaný stav Možnost komplexního vyhledávání jak pomocí čtečky čárových kódů, tak manuálního zadání názvu a dalších parametrů. Musí být možné materiál naskladnit i vyskladnit, a to i v případě, že mu nebylo přiděleno žádné výrobní číslo.

2.6 Měřidla a výrobky

Současný stav Každý zaměstnanec je zodpovědný za sledování stavu přiřazených měřidel. Je třeba udržovat přehled o těchto měřidlech, která jsou zaznamenána v informačním systému HELIOS. Stejný případ jsou i výrobky, výrobní příkazy a výkresy, které jsou evidovány v systému.

Problémy Ne všichni zaměstnanci mají přístup do systému, a proto musí o výpis měřidel, výrobní výkresy a balicí předpisy žádat své nadřízené.

Očekávaný stav Zaměstnanci budou vidět seznam měřidel ve své správě na terminálu a budou moct vyhledávat výrobky a zobrazovat si výkresy související s těmito výrobky.

2.7 Personální činnost

Současný stav Veškerá činnost týkající se zaměstnanců probíhá podpisem v papírové podobě, včetně různých bezpečnostních školení nebo upozornění.

Problémy Všechny záznamy s podpisy zaměstnanců se musí uchovávat v archivech, což ztěžuje dohledávání informací. Zároveň existuje riziko jejich ztráty nebo zničení. Pověřený pracovník musí osobně kontaktovat každého zaměstnance.

Očekávaný stav Převedení záznamů do digitální podoby. Každý zaměstnanec by měl po přihlášení do aplikace vidět, s čím udělil souhlas. Aplikace by také měla sloužit jako jednotný zdroj informací o budoucích školeních a dalších událostech týkajících se chodu firmy.

Kapitola 3

Dostupná řešení

Obecně lze říct, že firma chce co nejvíce proces výroby automatizovat a odstranit papírovou evidenci. Od digitalizace si slibuje více transparentnosti, možnost lépe analyzovat data a díky tomu včas odhalovat problémy. Digitalizace je pro firmu důležitá také z hlediska kvality, protože umožňuje lepší monitorování zmetků, přesnější evidenci údržby a mnoho dalšího. Firma musí správně dodržovat dokumentaci, což se kontroluje při pravidelných auditech, a digitalizace většiny záznamů tento proces značně urychlí a zpřesní.

Evidence docházky Systémů pro evidenci docházky zaměstnanců existuje hodně. Jsou k dispozici řešení přímo integrovaná se systémem HELIOS, například GIRITON[1]. Jedná se o docházkový systém určený pro HELIOS iNuvio a kromě terminálového řešení poskytuje také cloudovou aplikaci pro další správu zaměstnanců a procesů. Nevýhodou je měsíční poplatek. Podobným řešením je například systém ANeT[2].

Další možností jsou docházkové systémy bez návaznosti na HELIOS. Příklad může být Aktion[3] nebo PREMIER[4] a množství dalších. Kromě finanční náročnosti mají všechna tato řešení, společně se systémy navázanými na HELIOS, jeden shodný nedostatek, a sice to, že kromě software dodávají i vlastní terminály. Firma by tedy nevyužila technologie, které již má, a zaměstnanci by museli používat více zařízení.

Odvádění výroby Samotný systém HELIOS nabízí modul pro odvádění výroby[5].

Problém s tímto modulem je ale v tom, že nelze žádným způsobem zpřístupnit rozhraní ve zjednodušené formě zaměstnancům. Zaměstnanec jako takový není plným uživatelem sytému a ani nemá žádné heslo pro zabezpečený přístup. Pracovníci na dílně nesmí mít přímý přístup k databázi HELIOSu a dalším prvkům, který by v tomto modulu získali, pokud bychom z nich uživatele udělali. V současnosti tedy tento modul slouží pouze jako podpůrný nástroj pro řízení firmy.

Další možností je oslovení partnerských firem Assecco Solutions, autora systému HELIOS. Příkladem takové nabídky je Gatema MES[6]. Typicky mají partneři společnosti Assecco Solutions již existující řešení, která mohou za další poplatek upravovat a přizpůsobovat potřebám firem, ovšem s omezením v rozsahu úprav.

[1] https://www.helios.eu/dochazka-giriton
[2] https://www.helios.eu/dochazka-anet
[3] https://www.aktion.cz/produkty/dochazkovy-system.html
[4] https://www.premier.cz/produkty/moduly/mzdy_a_personalistika/dochazkove-systemy/
[5] https://www.helios.eu/files/al-inuvio-vyroba.pdf
[6] https://www.gatemait.cz/gatema-mes/

Ani toto řešení není pro zadavatelskou firmu vyhovující. Protože se jedná o předpřipravená řešení, která vychází z obecného používání systému HELIOS, bylo by nutné je přizpůsobit na míru konkrétní firmě. Toto není vždy možné udělat v plném rozsahu, a je často nutné přistoupit ke kompromisům, například v podobě uživatelského rozhraní, které nemusí dané firmě vyhovovat.

Analýza dat Systém HELIOS nabízí pro verzi iNuvio analytický modul[7]. Problémem ale je, že v současném procesu výroby a správy skladu v zadavatelské firmě není v systému HELIOS žádná informace o návaznosti použitých materiálů a výrobních operací. To je zapříčiněno způsobem, jakým firma integrovala nově nasazený systém HELIOS se zavedeným procesem výroby. Materiály se v současnosti odvádí hromadně, bez návaznosti na výrobní příkaz, a to i když tuto návaznost systém HELIOS umožňuje evidovat. Z toho důvodu by nebylo možné ani s tímto analytickým modulem, vytvořený přímo autory systému HELIOS, zjistit, kolik stojí výsledný výrobek včetně ceny materiálu. Kvůli tomu by bylo nutné stávající řešení firmě přizpůsobovat, anebo změnit proces odvádění výroby, což souvisí s nutností nové aplikace pro podporu výroby.

Shrnutí dostupných řešení V kapitole 2 byl popsán jak současný stav a technologie použité ve firmě, tak i očekávaný výsledek po implementaci nového systému. Z rozsahu analýzy je zřejmé, že se bude jen obtížně hledat již existující řešení, které by splnilo všechny požadavky a efektivně využívalo dostupné zdroje. Existují řešení dílčí, jejichž příklady byly uvedeny v této kapitole, ale jejich nasazení typicky představuje velkou investici do každé oblasti zvlášť a použití několika systémů najednou, což není pro středně velkou firmu žádoucí a ani finančně výhodné. I když je HELIOS propracovaným systémem, který poskytuje řadu modulů uspokojujících část požadavků, samotným problémem je i způsob používání systému ve firmách, a to včetně té zadavatelské, který se často neslučuje s očekávaným použitím. Proto i kdyby byla tato řešení využita, musela by být velmi přizpůsobována. Z toho důvodu bude v rámci této práce implementován systém zcela nový.

[7] https://www.helios.eu/business-intelligence

Kapitola 4

Informační systém HELIOS

HELIOS označuje skupinu ERP systémů[1] určených pro automatizaci firemních procesů od firmy Asseco Solutions. Na trhu je od roku 1990. Jednotlivé varianty systému jsou přizpůsobeny odvětví a velikosti firem, do kterých jsou implementovány, ale obecně jsou určeny zejména pro malé a střední podniky.[9].

4.1 HELIOS iNuvio

Zadavatelská firma využívá softwarovou variantu HELIOS iNuvio, dříve označovanou jako HELIOS Orange. Tato varianta je obvykle doporučována středně velkým podnikům a využívá ji více než 4500 firem v Česku a na Slovensku.[10]

Systém je založen na modulárním principu. K nejdůležitějšímu jádru se přidávají moduly podle potřeby konkrétní firmy. Výhodou informačního systému HELIOS iNuvio je, že těchto modulů obsahuje poměrně hodně, všechny mají společné uživatelské rozhraní a interagují spolu. Modul výroby přijímá a odvádí výrobky ze skladu, to zase umožňuje generovat faktury a tak dále. Díky tomu není nutné mít pro chod firmy mnoho dalších programů, ale je možné všechny oblasti řídit pomocí jednoho systému. Příklady nodulů, které jsou zároveň nejdůležitější pro tuto práci, jsou:

Výroba Modul výroby je rozdělen na dvě části, a to řízení výroby a technologická příprava výroby. Důležitým prvkem přípravy výroby jsou technologické postupy, díky kterým je možné rozčlenit výrobu na jednotlivé operace. Modul podporuje i verzování technologických postupů. V řízení výroby se určuje, jaké výrobky se mají vyrobit. Obsahuje kompletní evidenci výrobků, a to včetně data jejich výroby a návaznosti na objednávky. Je možné přesně dohledat, kdo daný výrobek vyráběl a jak dlouho operace trvala.[11]

Sklady Modul skladů umožňuje evidovat nejen konkrétní výrobky a materiály, ale i jejich šarži. Díky návaznosti na výrobní čisla je možné zpětně dohledat, jaká šarže nebo tavba materiálu byla pro daný výrobek použita. Tento modul také eviduje pohyby zboží a materiálů na skladě, včetně měrných jednotek, skladových cen a nákladů. Pomocí něj je možné zboží naskladňovat a generovat tak příjemky, neboli doklady o příjmu zboží na sklad, anebo vyskladňovat jak obyčejným výdejem, tak s návazností na objednávku či výrobu. I v tomto případě se generují výdejky, neboli doklady o výdeji zboží ze skladu.[12]

[1] ERP je podnikový informační systém, který umožňuje efektivně řídit a automatizovat firemní procesy[2, s. 77]

Mezi další nabízené moduly patří například obchod pro podporu obchodního procesu, ekonomika zajišťující účetnictví nebo personalistika pro evidenci zaměstnanců.

4.2 Aplikační a databázové rozhraní

Pro variantu systému iNuvio existuje komunita HELIOS Space. Akceptací podmínek a členstvím v této komunitě je možné získat soubor nástrojů, příkladů a dokumentací určených pro vývoj externích modulů a pluginů. Členové mají zdarma přístup k rozhrani HELIOS Interface, základní typové knihovně, a také k sadě instalačních nástrojů a pomocných knihoven HELIOS iNuvio Core, které značně usnadňují vývoj dalšího software.[13] Tyto nástroje jsou založeny na technologii COM (Component Object Model, neboli objektový model komponent). COM je platformově nezávislý a objektově orientovaný systém pro vytváření softwarových komponent. Obecně se používá k vytváření objektů pro meziprocesovou komunikaci.[8] Nespornou výhodou technologie COM je, že se jedná o přenositelnou reprezentaci objektů mezi různými jazyky a platformami, a díky tomu je možné implementovat komunikaci mezi aplikacemi napsanými v různých programovacích jazycích. Problémem ale je, že registrace a členství v komunitě HELIOS Space je placené, stejně jako registrace samotného vyvinutého pluginu.[13]

Další možností je komunikace se systémem pomocí databázového rozhraní. Součástí systému je řada procedur pro podporu odvádění výroby i dalších operací a pro provedení určité akce není většinou nutné spouštět řadu individuálních SQL[2] příkazů. Nevýhodou je absence dokumentace (ta je přístupná pouze pro registrované členy). Systém HELIOS ale umožňuje spustit mód tzv. odposlouchávání, který trasuje každý SQL příkaz, který aplikace spustila. Díky tomu je možné při provádění jednotlivých operací zjistit, které příkazy a s jakými parametry byly spuštěny. Zřejmou nevýhodou je zpětná kompatibilita systému v případě aktualizace, se kterou je potřeba při návrhu aplikace počítat.

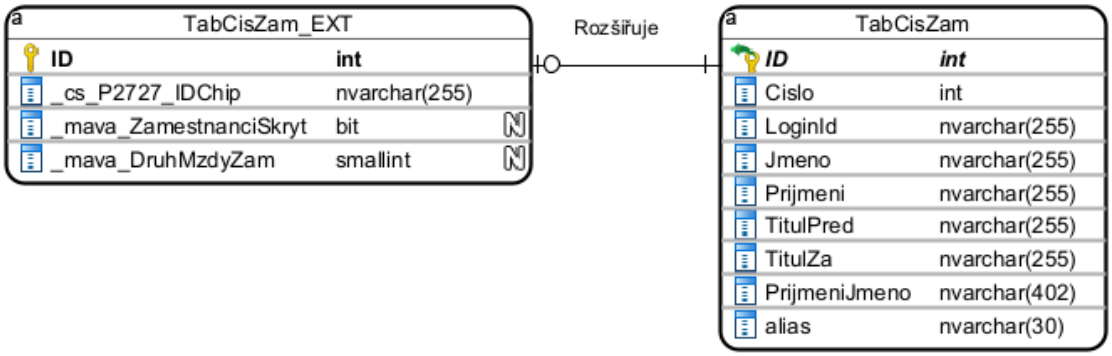
4.3 Databázová struktura

Aby bylo možné využít databázové rozhraní informačního systému, je nejdříve nutné prozkoumat a pochopit jeho strukturu, a to i pokud využijeme již připravené procedury. S ohledem na jiné firmení aplikace a potenciální budoucí rozšíření systému není možné měnit databázovou strukturu za žádných podmínek, a ani ji rozšiřovat nekorektním způsobem. HELIOS při každém spuštění kontroluje všechny procedury a tabulky, a pokud něco není v pořádku, například byl změněn typ sloupce, obnoví systém do výchozího korektního stavu. V této sekci budou popsány ty nejdůležitější entity, se kterými bude aplikace pracovat, a vztahy a návaznosti mezi tabulkami představujícími tyto entity.

Modifikace a rozšíření databáze

Systém umožňuje nativní modifikaci existujících tabulek a vytváření nových pomocí uživatelského rozhraní. Pokud chceme k existující tabulce přidat sloupec, interně se vytvoří tabulka nová, se stejným názvem a příponou "EXT", která s původní tabulkou sdílí primární klíč. Tato situace je znázorněna na obrázku 4.1, kde je tabulka zaměstnanců rozšířena o firemní tabulku s přídavnými informacemi, jako je například číslo čipu zaměstnance.

[2]SQL, celým názvem Structured Query Language, je strukturovaný dotazovací jazyk využívaný k manipulaci s daty v relačních databázích



Obrázek 4.1: Rozšíření tabulky zaměstnanců v databázi systému HELIOS

Také je možné vytvořit tabulku úplně novou. V takovém případě v databázi vzniknou dva záznamy, a sice samotná tabulka označená doporučeným prefixem "Tabx", a pohled, označený prefixem "hvw". Pohled určuje vzhled tabulky zobrazený v uživatelském rozhraní systému HELIOS. Tyto tabulky byly vytvořemy zadavatelskou firmou pro potřeby terminálové aplikace a digitalizace firemních procesů. Příkladem tohoto rozšíření je diagram na obrázku 4.6.

Diagramy důležitých entity

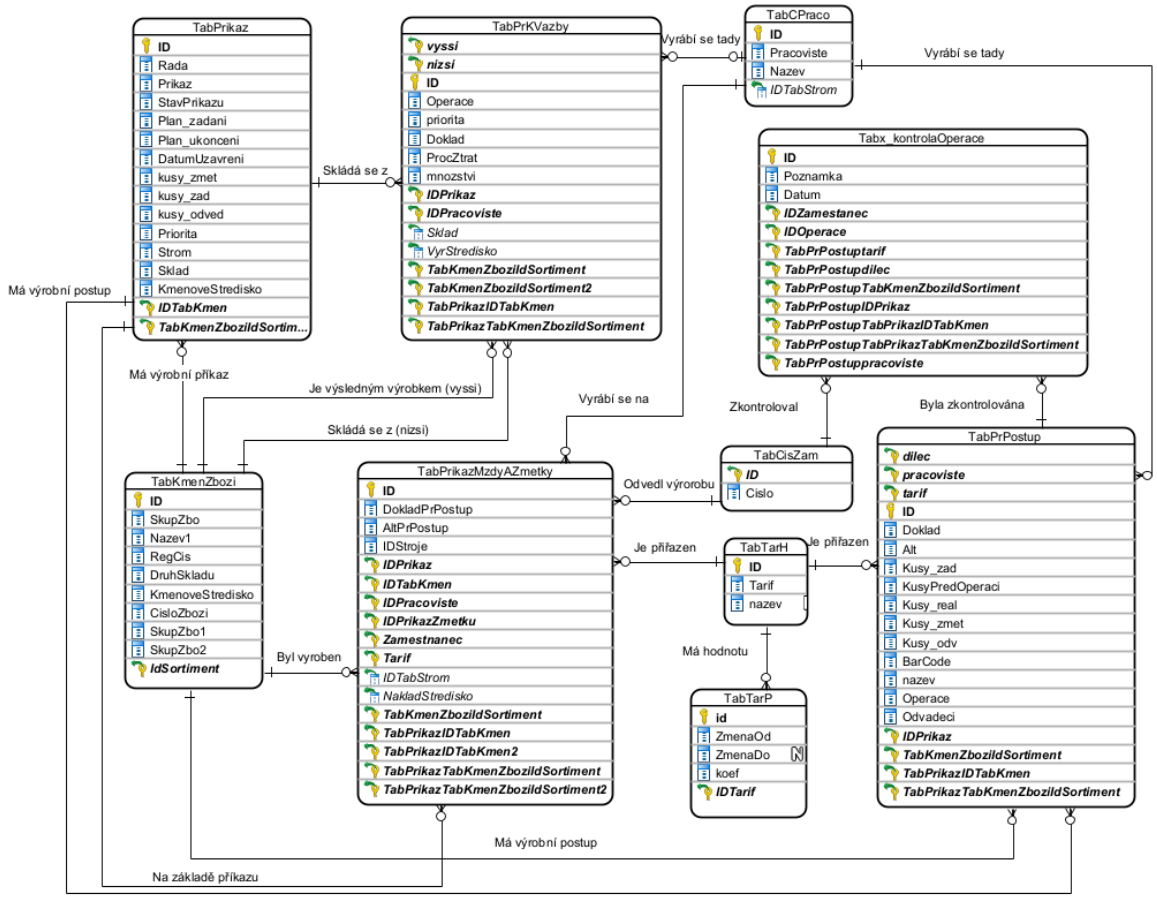
Diagramy na obrázcích 4.1 až 4.6 popisují důležité vztahy mezi entitami v systému HELIOS.

Odvádění výroby Diagramy entit nutných pro zajištění poaždavků na odvádění výroby podle sekce 2.3 jsou zobrazeny na obrázku 4.2. Kvůli velikosti entit byly pro přehlednost uvedeny pouze vazební atributy, entity na diagramech nejsou úplné.

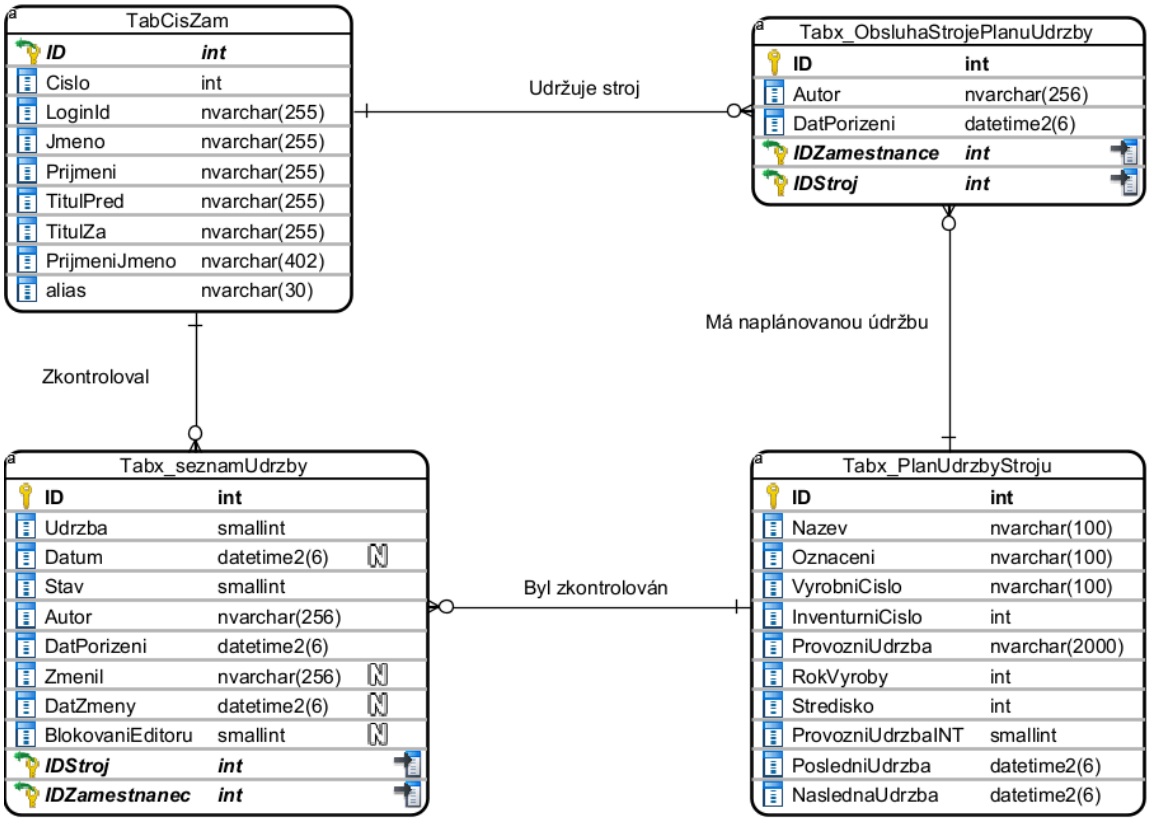
Entita TabKmenZbozi představuje jednotlivé výrobky, materiály a jejich součásti. Výrobky jsou vyráběny na základě výrobních příkazů, ty jsou reprezentovány entitou TabPrikaz. Výrobní příkazy se skládají z předem definovaného sledu operací navázaných na konkrétní výrobek. Jednu operaci představuje entita TabPrPostup. Z jakých materiálů a částí se skládají výsledné výrobky určuje entita TabPrKVazby. Operace nemusí být provedeny celé najednou. Entita TabPrikazMzdyAZmetky obsahuje informace o tom, kdy, kde a kolik se čeho vyrobilo, případně kolik bylo zmetků neboli nepovedených kusů. TabCPraco označuje výrobní pracoviště, například laser nebo soustruh. V tabulce Tabx_kontrolaOperace se ukládají záznamy o kontrole průběhu jednotlivých operací. Tabulky TabTarH a TabTarP popisují, kolik stojí operace za jednotku času (tarif), a umožňují tak vypočítat očekávané náklady.

Evidenci kontroly strojů zajišťují entity zobrazené na obrázku 4.3. Tabx_PlanUdrzbyStroju obsahuje informace o jednotlivých strojích a také o tom, s jakou periodou by měla být prováděna údržba. Tabulka Tabx_ObsluhaStrojePlanuUdrzby přiřazuje zaměstnancům stroje ke kontrole, udává, který zaměstnanec má ten daný stroj na starosti. Entita Tabx_seznamUdrzby uchovává historii jednotlivých kontrol a údržeb strojů.

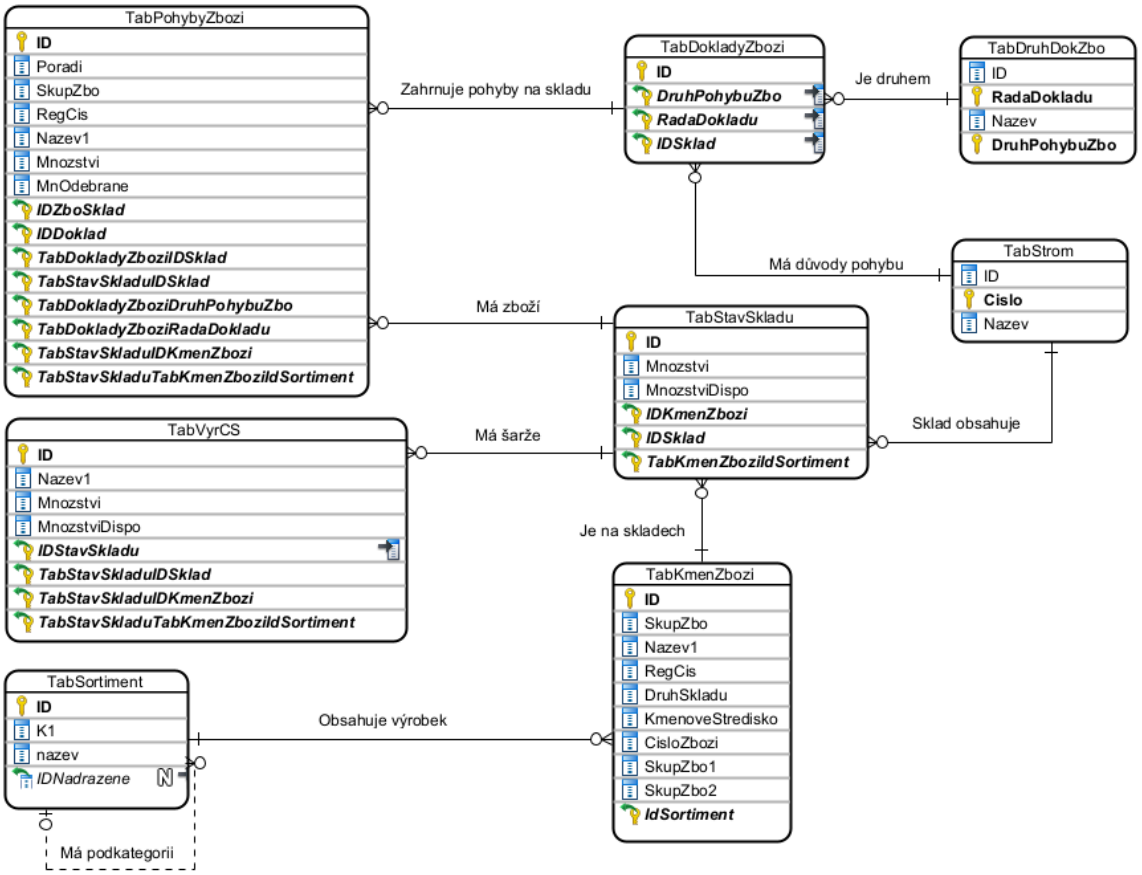
Sklady a materiály Entity zobrazené na obrázku 4.4 slouží k zajištění požadavků ze sekce 2.5. Entita TabKmenZbozi je shodná s entitou používanou v odvádění výroby a ozna-



Obrázek 4.2: ER diagram pro část odvádění výroby



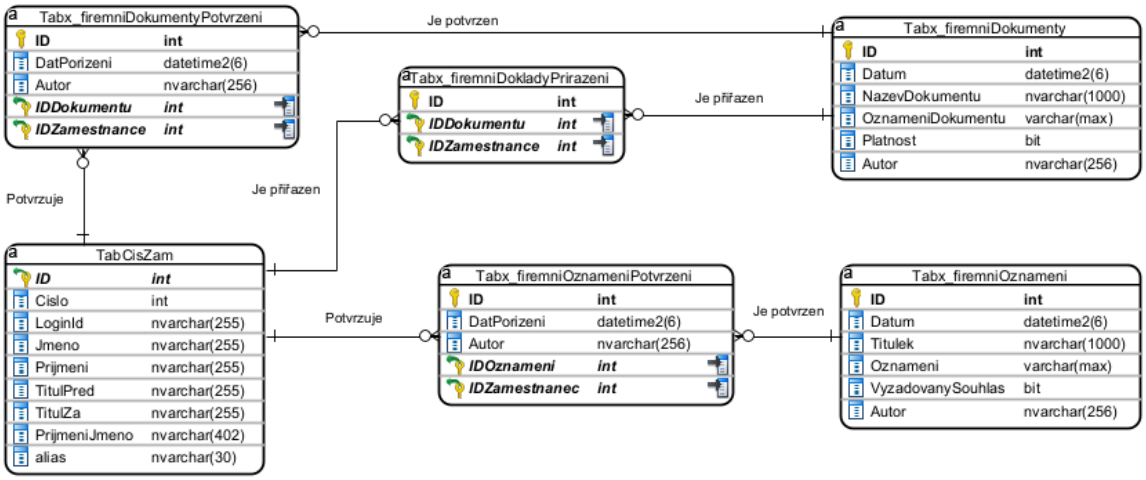
Obrázek 4.3: ER diagram pro údržbu a kontrolu strojů



Obrázek 4.4: ER diagram pro správu skladů a materiálů

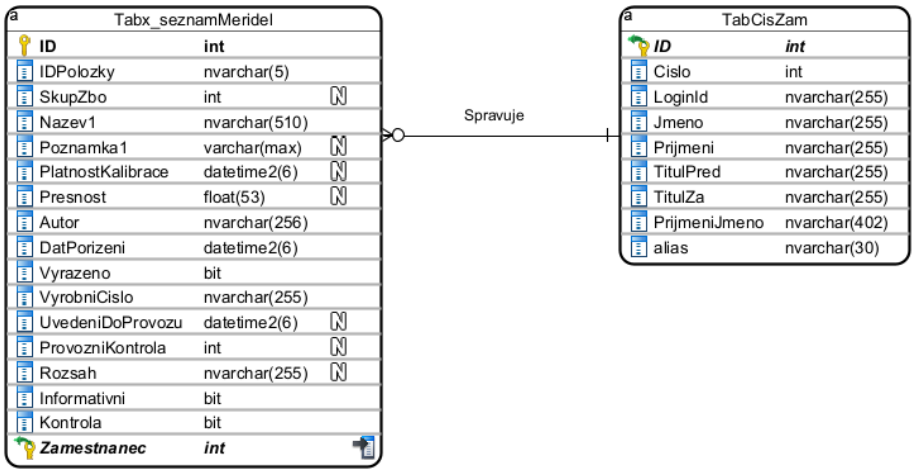
čuje jednotlivé výrobky a materiály. Každý výrobek může patřit do nějakého sortimentu neboli kategorie, k tomu slouží entita TabSortiment. Počet kusů na skladě, definovaný entitou TabStrom, eviduje entita TabStavSkladu. Materiály mohou, ale nemusí, mít několik šarží. U některých výrobků je důležité vědět, z kterého konkrétního materiálu byl vyráběn. Šarže obsahuje tabulka TabVyrCS. Pohyby na skladě, tedy naskladňování i vyskladňování, je zaznamenáno entitou TabPohybyZbozi. TabDokladyZbozi a TabDruhDokZbo určují typ a důvod pohybu, příkladem může být výdejka do výroby anebo vydaná objednávka.

Personální činnost Digitalizace personální činnosti podle části 2.7 je zajištěna pomocí entit na obrázku 4.5. Jsou rozlišeny dva typy dokumentů, a sice obecná oznámení, jako například ohlášení nadcházejícího auditu (entita Tabx_firemniOznameni), a dokumenty vyžadující potvrzení zaměstnanců, například oznámení o kamerovém systému (entita Tabx_firemniDokumenty). U obyčejných oznámení se uchovává pouze informace o tom, zda jej zaměstnanec přečetl (Tabx_firemniOznameniPotvrzeni), u dokumentů je přidána vazba přiřazení. Pokud má firemní dokument navázanou entitu Tabx_firemniDokladyPrirazeni, jedná se o oznámení přiřazené a zobrazené pouze některým zaměstnancům. V opačném případě je dokument zobrazen všem. Skutečnost, jestli zaměstnanec dokument podepsal, je uchována v entitě Tabx_firemniDokumentyPotvrzeni.



Obrázek 4.5: ER diagram pro modul personální činnosti

Měřidla a výrobky Na obrázku 4.6 je přiřazení měřidel jednotlivým zaměstnancům. Jako výrobek označujeme entity TabKmenZbozi z diagramu 4.2, jejichž atribut SkupZbo nabývá hodnoty 400.



Obrázek 4.6: ER diagram pro správu měřidel

Kapitola 5

Návrh řešení

Protože firma požaduje, aby aplikaci bylo možné využívat zejména na výrobních terminálech, ale také na mobilních zařízeních a počítačích bez nutnosti instalace dalšího software, jako implementační řešení byly zvoleny webové technologie. Výběr konkrétních technologií a jazyků je zdůvodněn v jednotlivých kapitolách a vychází zejména z faktu, že aplikace je webového charakteru.

Systém byl navrhován za použití třívrstvé architektury. V tomto typu architektury jsou komponenty organizovány do horizontálních vrstev, kdy každá z nich plní určitý vymezený účel a má zodpovědnost za omezený typ akcí. V případě třívrstvé architektury je systém rozdělen na prezentační, aplikační a datovou část. Prezentační vrstva zobrazuje data uživateli, je zodpovědná za vykreslování uživatelského rozhraní a interakci s webovým prohlížečem. Aplikační vrstva vykonává samotnou logiku aplikace, obstarává data pro prezentační vrstvu a naopak odesílá zpracovaná data datové části. Datová část představuje úložiště entit a dalších typů dat potřebných pro chod aplikace. Díky tomu je aplikace rozdělena do tří nezávislých celků, které jsou od sebe navzájem abstrahovány. [5, s. 1-5]

V průběhu návrhu aplikace bylo vytvořeno několik prototypů, kde byl ověřen jak výběr jednotlivých technologií, tak i dostatečnost způsobu řešení problémů uvedených v analýze aktuální situace ve firmě popsaných v kapitole 2. Návrh v následujících podkapitolách vychází z prototypu aplikace v poslední iteraci.

5.1 Technická omezení

Řešení je vytvářeno na míru zadavatelské firmě, a proto musí fungovat co nejefektivněji s dostupnými zdroji bez nutnosti dokupovat další vybavení nebo licence speciálně kvůli systému. Před samotným návrhem je třeba zohlednit následující požadavky:

- **Server**: Aplikace bude hostovaná ve firmě na vlastním serveru (Microsoft Windows Server 2019).
- **Uživatelské rozhraní**: Zaměstnanci budou k aplikaci přistupovat přes terminály na dílnách. Vždy je k dispozici pouze jeden terminál na dílnu, odvádění výroby tedy musí být rychlé a jednoduché, aby na sebe pracovníci zbytečně nečekali. Terminály jsou dotykové a hlavní výpočetní jednotku tvoří Raspberry Pi 4B s operačním systémem Raspbian, který je možné zaměnit za jiný linuxového typu.
- **Připojení**: Zařízení mohou vzájemně komunikovat pouze na vnitřní síti. Ani terminály, ani serverová část aplikace nebude mít přístup k internetu nebo jiným sítím a

naopak se nebude možné připojit k jakékoliv části systému jinak než prostřednictvím interní sítě firmy.

5.2 Přihlašování uživatelů

Protože navrhovaná aplikace bude hostovaná na firemním serveru a budou využity webové technologie, byla jako řešení zvolena autentizace pomocí certifikátů, konkrétně X.509[1]. Certifikátem se budou autentifikovat stroje, v tomto případě terminály, nikoliv uživatelé. Každému terminálu bude vydán jeden certifikát. Certifikační autoritu představuje firemní server, kde bude aplikace hostována.

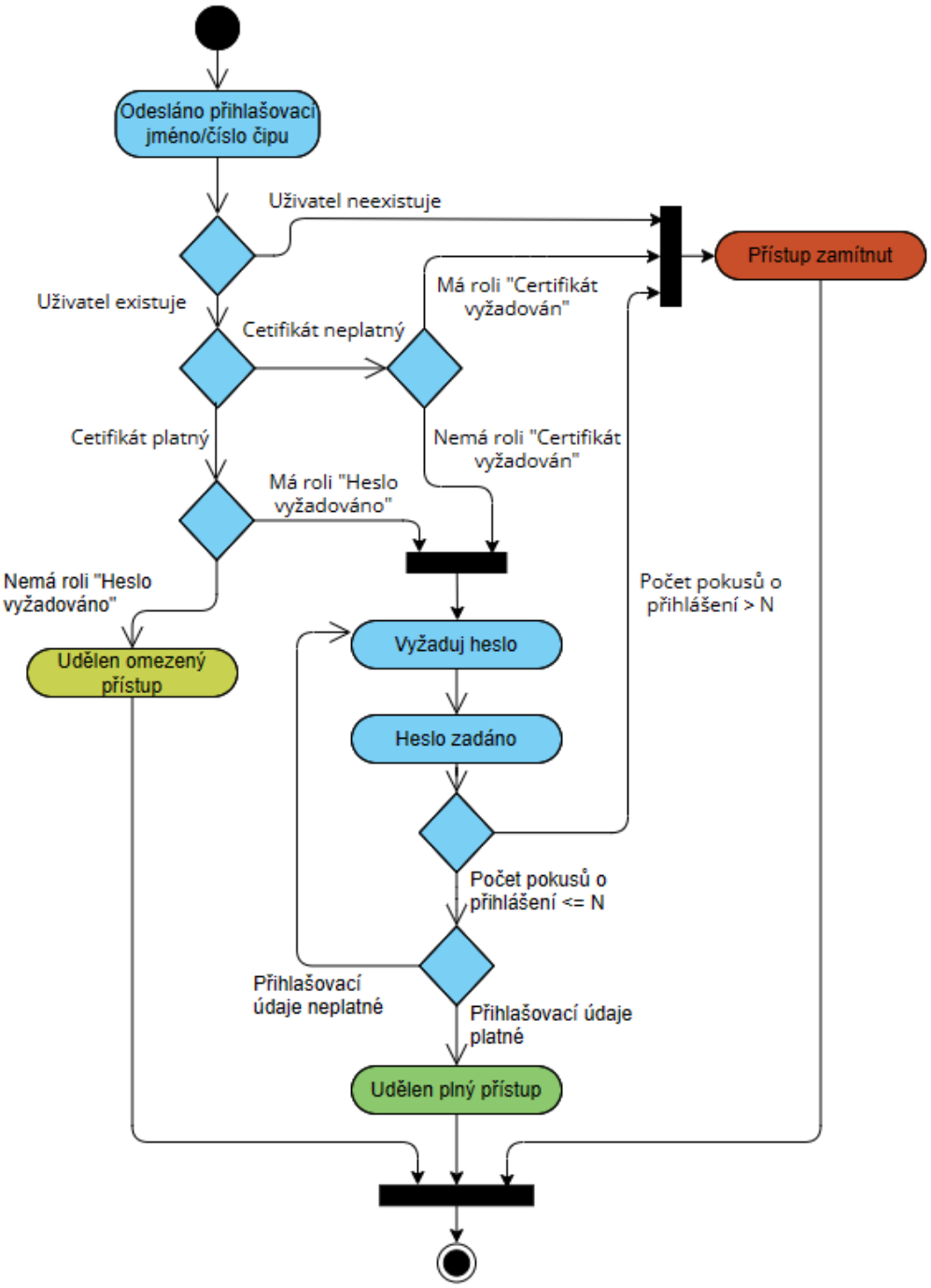
Přístup do systému se určí podle toho, jestli se uživatel přihlašuje pouze čipem, nebo i heslem, a také podle úrovně oprávnění uživatele a (ne)přítomnosti certifikátu. V systému byl kladen důraz jak na zajištění požadované bezpečnosti, tak i jednoduchosti a rychlosti přihlašování pro zaměstnance na dílně, kteří podřebují svou práci co nejvíce urychlit.

Celý proces přihlašování je podrobně znázorněn na obrázku 5.1. Je rozdělen na dva kroky. Nejdříve se uživatel prokáže buď číslem čipu, nebo svým uživatelskm jménem. Systém vyhodnotí oprávnění uživatele. Pokud se uživatel přihlašuje ze zařízení, které má nainstalováno platný certifikát, a nemá roli
textitheslo vyžadováno, je mu udělen omezený přístup. Pokud zařízení neposkytlo platný certifikát, a uživatel má přidělen roli *vyžaduj certifikát*, přístup je zamítnut. V opačném případě je vyžadováno heslo. Pokud se uživatel prokáže platným heslem, je mu udělen plný přístup.

Samotné přihlášení je zajištěno pomocí JWT. JWT, neboli JSON Web Token, je standardizovaný způsob reprezentace informací mezi stranami pomocí kompaktního objektu ve formátu JSON. Token je reprezentován jako posloupnost částí bezpečných pro přenášení pomocí URL oddělených tečkou, a sice hlavička (header), která obsahuje informace o použitém kódování, tělo (payload), ve kterém jsou přenášena samotná data, například role přidělené uživateli, a podpis (signature), pomocí nějž ověřujeme platnost tokenu.[6]

Token se ukládá v HTTP-only cookie, což je druh cookie, který je dostupný pouze serveru a nelze ho přečíst například v JavaScriptu. Tento typ cookie se odesílá s každým dotazem na server, a tak se uživatel s každou žádostí autentizuje, a zároveň je sníženo riziko zneužití například injektovaným kódem na straně klienta.

[1] X.509 je norma pro jednoduché podepisování založená na veřejném kliči popsaná v RFC 3280 [7]



Obrázek 5.1: Proces přihlášení uživatele do terminálové aplikace

5.3 Systémové role

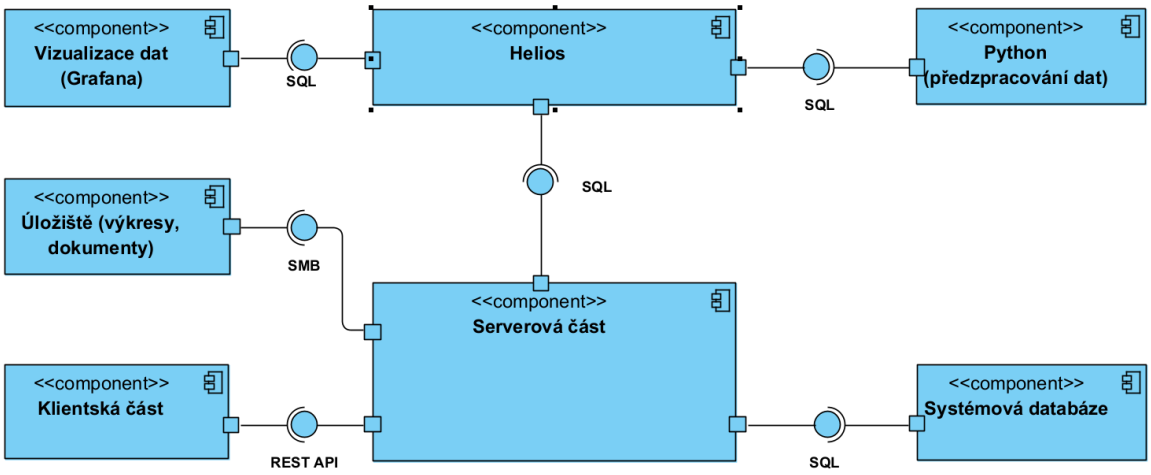
Role uživatelů v systému jsou hierarchicky členěny na několik úrovní. Typicky lze rozdělit uživatele na několik málo rolí, která mají různá oprávnění, například administrátor nebo skladník. V případě potřeby je možné uživateli přidělit nebo odebrat libovolnou kombinaci rolí nebo oprávnění. Seznam nejdůležitějších systémových rolí je následující:

- Superadmin (*ROLE_SUPERADMIN*)
  - Administrátor (*ROLE_ADMIN*)
    - Úprava uživatelů (*ROLE_USER_EDIT*)
    - Úprava rolí uživatelů (*ROLE_USER_EDIT_ROLES*)
    - Úprava docházky jiných uživatelů (*ROLE_USER_EDIT_ATTENDANCE*)
    - Export docházky (*ROLE_ADMIN_ATTENDANCE*)
    - Správa firemního kalendáře (*ROLE_ADMIN_CALENDAR*)
    - Role *Skladník* (*ROLE_STOREMAN*)
    - Vyžadováno heslo (*ROLE_PASSWORD_REQUIRED*)
- Kontrolor (*ROLE_INSPECTOR*)
  - Kontrola výrobní operace (*ROLE_INSPECT_PRODUCTION_OPERATION*)
- Skladník (*ROLE_STOREMAN*)
  - Vyskladnění materiálu (*ROLE_WAREHOUSE_OUTPUT*)
  - Naskladnění materiálu (*ROLE_WAREHOUSE_INPUT*)
- Zaměstnanec (*ROLE_WORKER*)
  - Vyskladnění materiálu (*ROLE_WAREHOUSE_OUTPUT*)
  - Vyžadován certifikát (*ROLE_X509_REQUIRED*)

5.4 Architektura systému

Podle požadavků v kapitole 2 byl systém rozčleněn do několika menších celků. Na obrázku 5.2 je toto rozdělení znázorněno. Komponenta *Helios* představuje informační systém HELIOS iNuvio popsaný v kapitole 4. Serverová část systému se na tuto komponentu bude napojovat přímo přes SQL rozhraní. Komponenta *Úložiště* je firemní server, na kterém jsou uloženy výkresy navázané na výrobky a další potřebné dokumenty. Se serverovou částí terminálové aplikace bude komunikovat protokolem SMB[2]. Komponenta označená jako Python obsahuje skripty, které předzpracovávají data potřebná k vizualizaci a analýze dat, aby databáze nebyla zatěžována dlouhými SQL dotazy při načítání přehledů. Ostatní části systému znázorněné na obrázku budou navrženy a popsány v této kapitole.

[2]SMB neboli Server Message Block je síťový komunikační protokol sloužící ke sdílení souborů a textových dat



Obrázek 5.2: Návrh systému - diagram komponent

Serverová část

Komponenta *Serverová částz* obrázku 2 je HTTP služba sjednocující data ze všech dostupných zdrojů (databáze informačního systému HELIOS, databáze terminálové aplikace a server s uloženými dokumenty). Sjednocená data vystavuje pod jednotným REST[3] rozhraním splňujícím specifikaci OpenAPI 3.1. Toto rozhraní slouží ke komunikaci s klientskou částí aplikace.

Ze sekce 5.2 vyplývá potřeba vydávání a ověřování X.509 certifikátů. Protože aplikace má být umístěna na vnitřní síti, je právě její serverová část certifikační autoritou. Cetifikační autorita certifikáty vydává, podepisuje a při každém dotazu přes REST API ověřuje jejich platnost.

Vzhledem k technologiím, které byly pro tuto práci použity (web), byl jako programovací jazyk zvolen PHP s frameworkem Symfony. Z použití Symfony vyplývá také MVC struktura projektu. MVC, neboli Model-View-Controller, je vzor, který aplikační část dělí na tři další vrstvy, a sice datový model (model), řídící logiku (controller, kontroler) a uživatelské rozhraní (view, pohled). Model zajišťuje namapování entit a obsahuje metody pro práci s databází a entitami jako takovými. Pohled obsahuje prvky pro tvorbu uživatelského rozhraní a obecně cokoliv, s čím přímo uživatel interaguje. V mnoha případech je výsledná obrazovka tvořena více pohledy zároveň. Kontroler je komponenta, která propojuje model a pohled a určuje, jak bude aplikace reagovat na uživatelské akce.[3, s. 1-3] V práci jsou v serverové části využity zejména prvky model a kontroler, pohledy jsou přesunuty do uživatelského rozhraní popsaného v sekci 5.4, které je tvořeno zvlášť. Objektové mapování modelů a abstrakci databázové vrstvy zajišťuje ORM[4] framework Doctrine. Standardizované REST API vytváří tzv. API-first[5] framework API Platform.

[3]REST, celým názvem Representational State Transfer

[4]ORM, celým názvem objektově relační mapování

[5]API-first přístup klade důraz na vytvoření samotného API na úplném začátku vývoje, na jehož základě se následně vyvíjí celá aplikace

Databázová vrstva

Z předchozích kapitol je zřejmé, že pro správné fungování systému bude nutné využít aktuální databázi systému HELIOS, kde jsou uchovávána všechna data týkající se výroby a dalších firemních procesů. Je taky zřejmé, že systém bude potřebovat další úložiště pro svá data, jinak by zahlcoval stávající databázi přebytečnými tabulkami, které by zbytečně znepřehledňovaly existující uživatelské rozhraní. Proto bude databázová vrstva rozdělena na dvě části - Helios a systémové jádro. Databázová vrstva informančního systému HELIOS byla popsána v sekci 4.3.

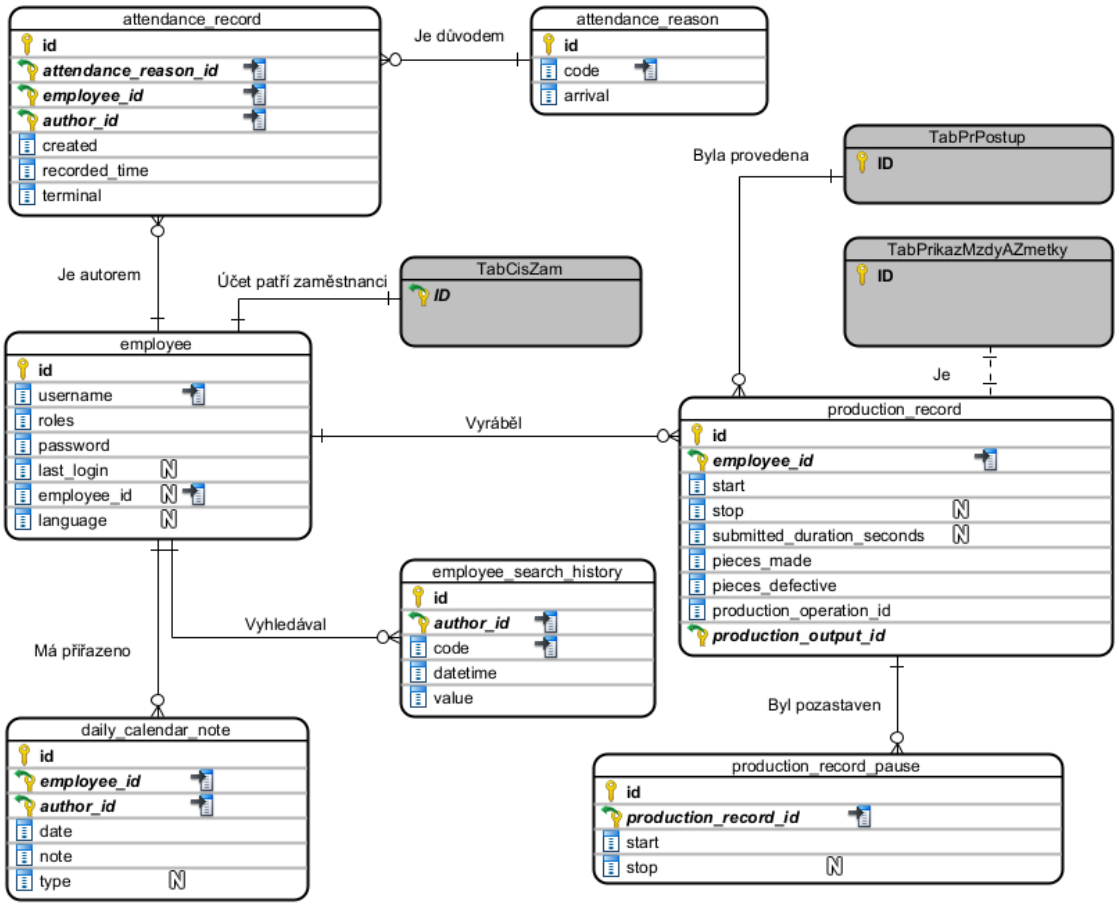
HELIOS iNuvio pro správné fungování vyžaduje Microsoft SQL Server[14] a zadavatelská firma také používá tento typ databáze. Bylo by možné jádro systému implementovat na jiné platformě, ale žádný požadavek ze strany firmy to nevyžaduje, ba naopak je jednodušší aplikaci nasadit a udržovat, obzvláště na lokálním firemním serveru, pokud se kombinuje pouze několik málo technologií. Z tohoto důvodu bude i jádro systému používat jako databázový systém Microsoft SQL Server.

Systémové entity Schéma hlavní části databáze terminálové aplikace je znázorněno na ER diagramu na obrázku 5.3. Pomocné entity jsou zobrazeny na obrázku 5.4. Diagramy byly vygenerovány automaticky z DDL[6] prototypu aplikace. Centrální entitou je employee, zaměstnanec, který představuje uživatele přihlášeného do terminálu. Vytvořením nové entity uživatele zajistíme bezpečnost systému oddělenou od stávajícího ERP systému HELIOS. Každý uživatel má na sebe navázanou entitu zaměstnance TabCisZam, v diagramu označenou šedou barvou. Všechny tabulky vyznačené v diagramu šedou barvou jsou entity popsané v sekci 4.3. Tabulka employee_search_history uchovává historii vyhledávání uživatelů, která urychluje další vyhledávání příkazů a výrobků.

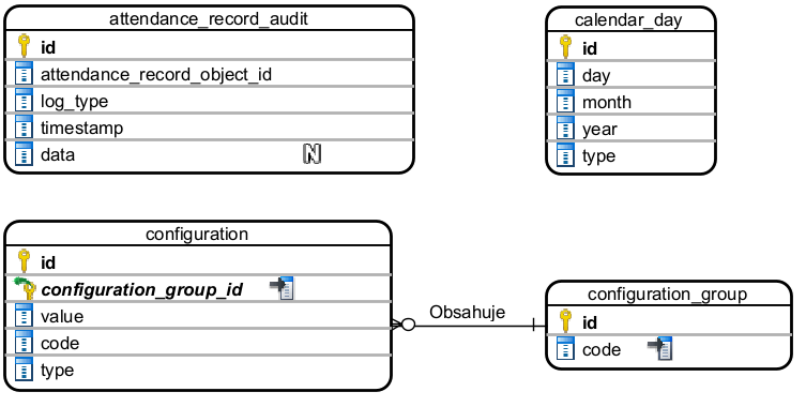
Docházka Docházku zaměstnanců je možné evidovat díky entitám attendance_record, která představuje jeden záznam uživatele v docházce, attendance_reason, shrnující důvod příchodu nebo odchodu a daily_calendar_note, poznámce, kterou můžou zaměstnanci přidávat ke každému dni. Součástí systému je i pomocná entita calendar_day, zobrazená v diagramu na obrázku 5.4. Díky této entitě je možné označit jednotlivé dny jako speciální, například státní svátky nebo firemní dovolenou. Na speciální dny se bere ohled při výpočtu odpracovaných hodin. Na tomto diagramu je také znázorněna entita attendance_record_audit, která uchovává historii změn záznamů v docházce. Díky ní je možné zpětně odhalit manuální zásahy do docházky. V tabulce je uložen čas editace (případně vzniku či smazání, sloupec timestamp) záznamu, obsah záznamu ve chvíli úpravy (sloupec data) a typ editace (sloupec log_type, možnost create, edit, update a delete).

Odvádění výroby Ačkoliv je odvádění výroby řešeno primárně v informačním systému HELIOS, jsou některá data uchovávána v databázi aplikace. Důvodem je jak lepší formát dat pro analýzu, tak splnění požadavků na automatický výpočet délky trvání výroby, aniž bychom měnili strukturu stávající databáze. K tomu slouží entita production_record na obrázku 5.3. Podobně jako entita TabPrikazMzdyAZmetky popsaná v sekci 4.3 i tato uchovává informace o průběhu výroby. Entita production_record_pause slouží k pozastavení probíhající výroby a zpětnému výpočtu celkové doby trvání operace.

[6] DDL, data definition language, je jazyk umožňující definovat strukturu dat



Obrázek 5.3: ER diagram hlavní části databáze terminálové aplikace



Obrázek 5.4: ER diagram pomocných entit databáze terminálové aplikace

Klientská část

Klientská část aplikace je navržena jako jednostránková aplikace, neboli SPA. (celým názevm single-page application). Výhodou jednostránkové aplikace proti standardním vícestránkovým je zejména dynamické načítání stránky, při kterém se automaticky aktualizuje pouze potřebná malá část celého obsahu. To umožňuje plynulý uživatelský zážitek, minimalizuje celkový čas načítání obsahu a vyvolává dojem, že se jedná o nativní aplikaci.[4, s. 497] V případě této práce má tento efekt zásadní význam, protože nejdůležitější cílovou skupinou jsou zaměstnanci přistupující k aplikaci z terminálů. Zaměstnanci smí mít pouze omezený přístup k software, a proto terminály operují v takzvaném kiosk režimu, a i když se jedná o webovou aplikaci, celý adresní řádek je pro ně nepřístupný. Kiosk, neboli režim veřejného terminálu, je takový režim, ve kterém jsou možnosti zařízení limitovány na několik málo předem definovaných funkcí.[25].

5.5 Uživatelské rozhraní

Při návrhu uživatelského rozhraní bylo upřednostněno jednoduché používání aplikace na úkor moderních prvků a trendů, zejména kvůli využití v průmyslu. Ačkoliv je celková aplikace navrhována jako webová a bude možné ji používat ze všech zařízení, nejvíce ji budou využívat zaměstnanci firmy při zápisu docházky a odvádění výroby a materiálu na terminálech. Z toho důvodu je při návrhu kladen největší důraz na pohodlné využívání aplikace na dotykových obrazovkách.

Vzhledem k rozsahu aplikace, která obsahuje celkem devět modulů, z nichž některé se dále člení na další podmoduly a stránky, budou v této kapitole uvedeny pouze ty nejdůležitější obrazovky. Obrazovky, které zde uvedeny nebudou, využívají prvky, které jsou popsané v následujících sekcích, a jsou navrhovány v souladu s celkovým grafickým konceptem aplikace.

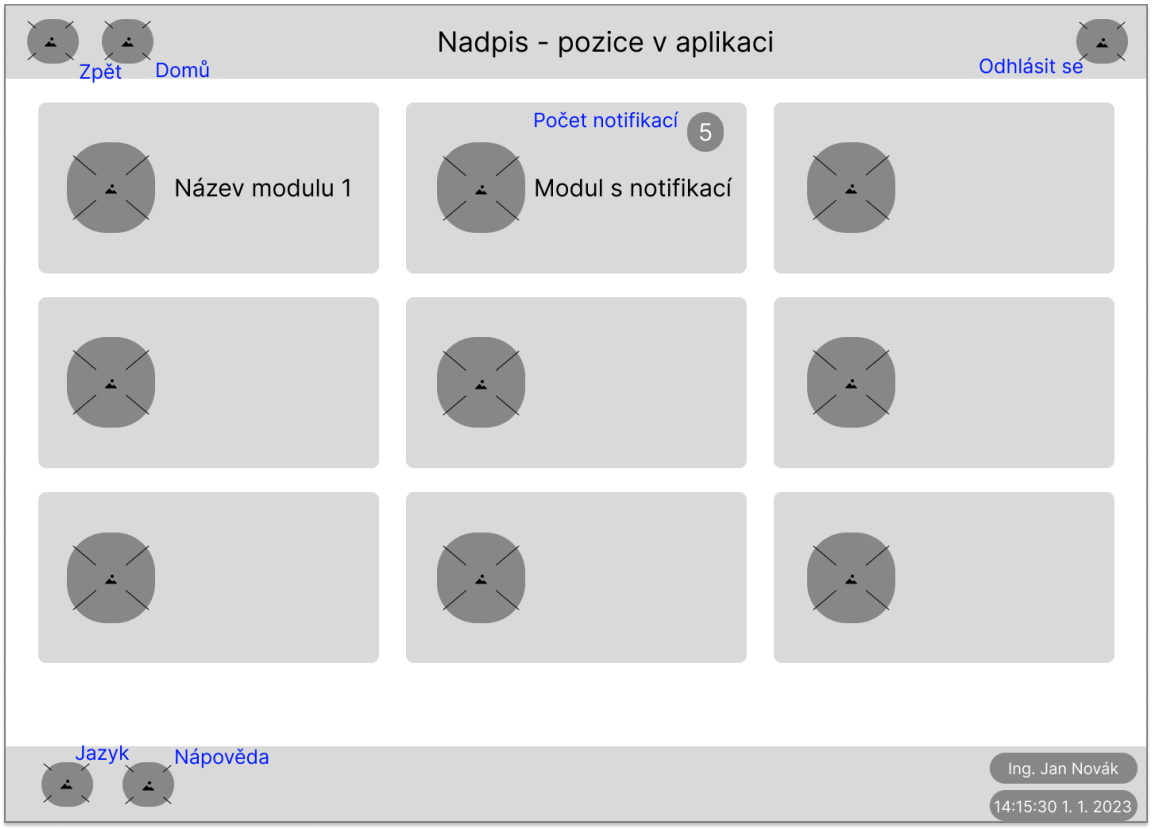
Domovská obrazovka

Návrh domovské obrazovky je zobrazen na obrázku 5.5. Základní rozložení stránky je shodné pro všechny následující obrazovky. Horní lišta vždy obsahuje tlačítka zpět, které uživatele vrací o jeden krok dozadu, a domů, odkazující na domovskou obrazovku (vlevo), nadpis, který udává pozici uživatele v aplikaci (uprostřed), a tlačítko pro odhlášení (vpravo). Na spodní liště jsou tlačítka pro změnu jazyka (čeština, angličtina a slovenština) a nápověda (vlevo). Po kliknutí na nápovědu se zobrazí vyskakovací okno s popisem dané stránky a instrukcemi. V pravém dolním rohu je zobrazen právě přihlášený uživatel a aktuální čas s přesností na sekundy.

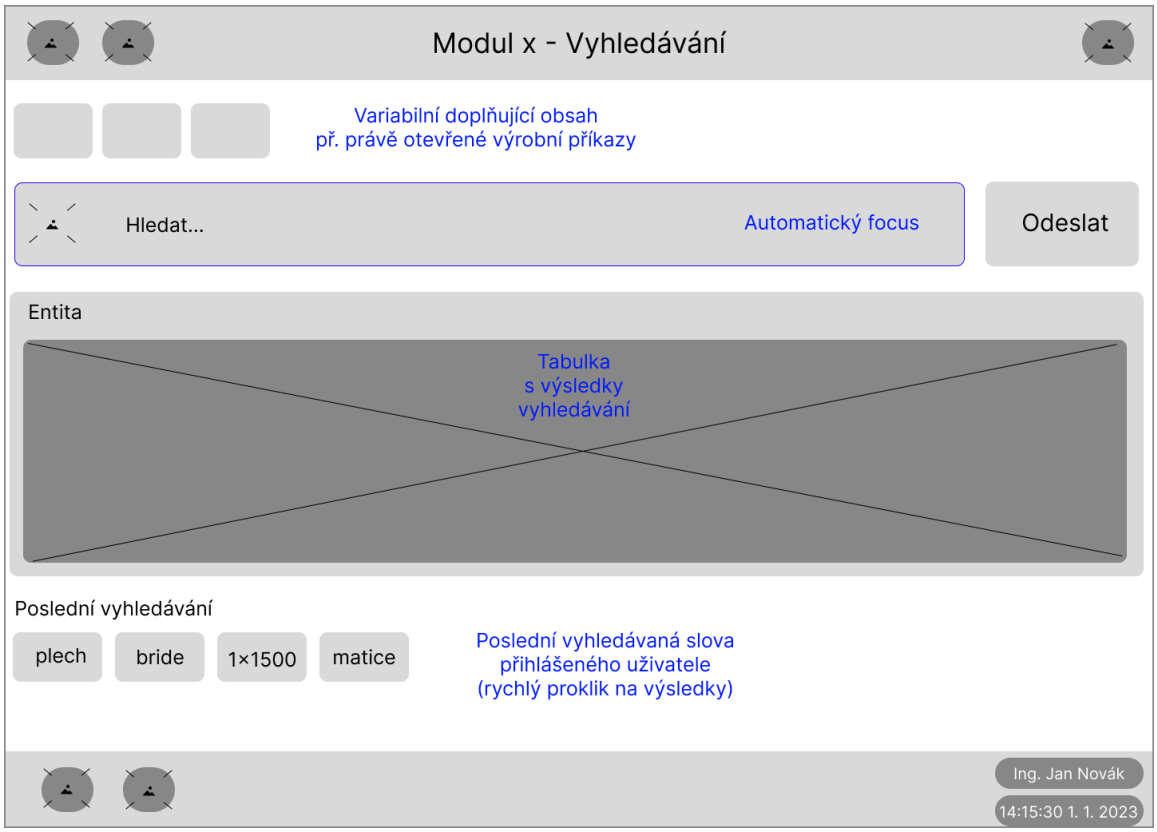
Aplikace je členěna na několik modulů. Domovská obrazovka slouží jako rozcestník s odkazy ke všem modulům, která má aktuálně přihlášený uživatel zpřístupněné. Stejným vizuálním způsobem, a to pomocí dlaždic s názvy modulů a ikonkami, jsou řešeny případné rozcestníky v samotných modulech, pokud jsou dále členěny na několik dalších.

Vyhledávání

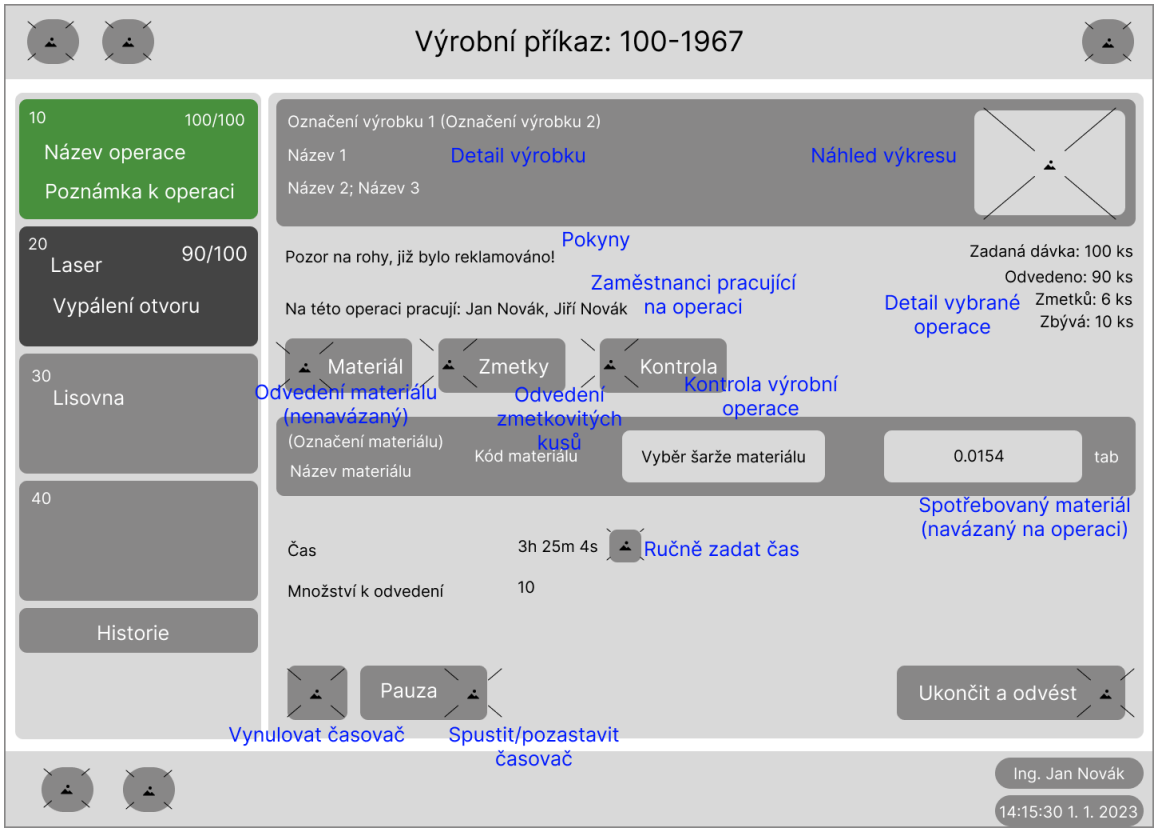
Několik modulů využívá funkci vyhledávání entit pomocí čtečky čárových kódů i manuálního hledání (výrobní příkazy, materiály, výrobky, správa uživatelů). Vyhledávácí obrazovka pro tyto části modulů je velmi podobná a její obecný koncept je zobrazen na obrázku 5.6.



Obrázek 5.5: Grafický návrh domovské obrazovky



Obrázek 5.6: Grafický návrh modulu odvádění výroby



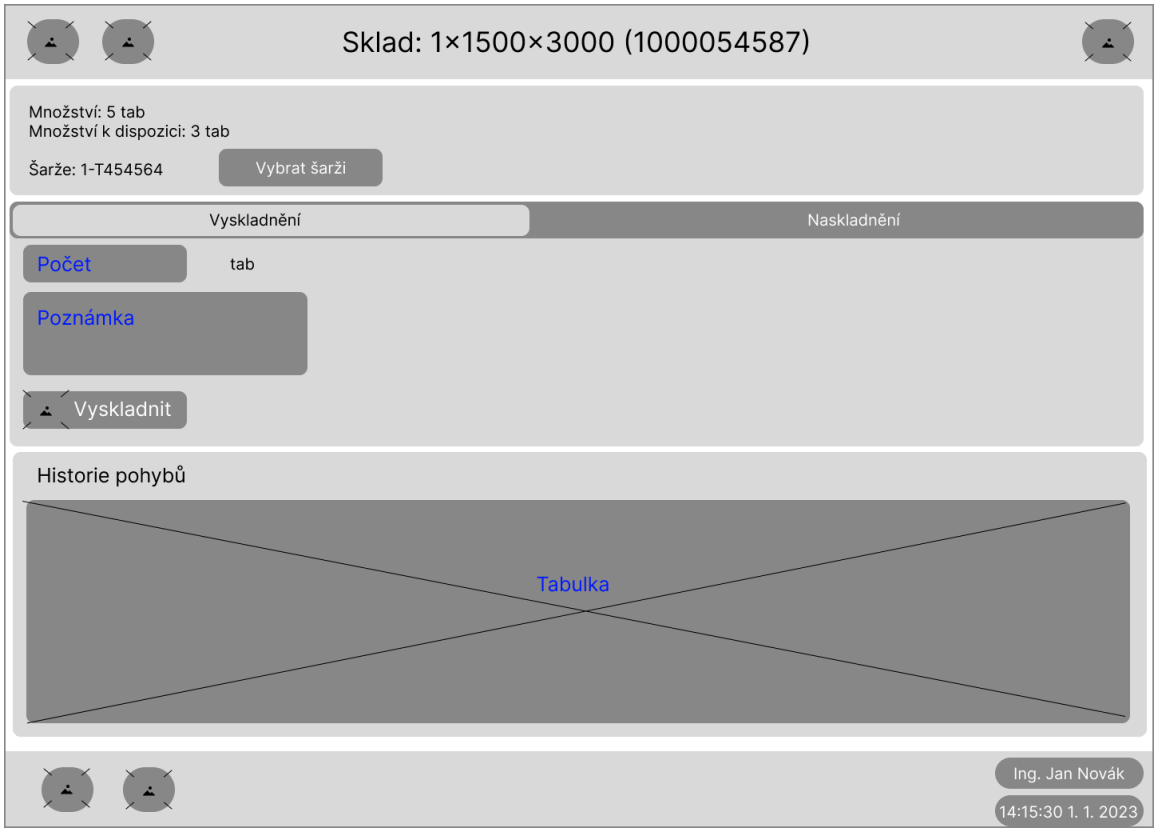
Obrázek 5.7: Grafický návrh modulu odvádění výroby

Důležité je, aby při načtení stránky bylo automaticky zaměřeno vyhledávací pole. Díky tomu je vyhledávání pomocí čtečky rychlejší a pohodlnější. Hned pod vyhledávacím polem je zobrazena tabulka s výsledky vyhledávání. Ve většině případů jsou řádky tabulky odkazem do dalších částí aplikace, například na odvádění výroby přes daný výrobní příkaz. Pod tabulkou výsledků je seznam posledních vyhledávaných výrazů přihlášeného uživatele, zvlášť pro každý typ vyhledávání (historie vyhledávání výrobních příkazů je oddělená od historie vyhledávání výrobků).

Modul odvádění výroby

Návrh obrazovky pro odvádění výroby je zobrazen na obrázku 5.7. Každý výrobní příkaz je složen z několika operací, které jsou přehledně zobrazeny v levém panelu, a z nich může uživatel vybírat. Vybrání operace je znázorněno jinou barvou, v případě návrhu tmavě hnědou. Zelené podbarvení značí, že operace byla již splněna. Poslední tlačítko v menu je historie, které uživatele odkáže na přehledný seznam všech úkonů, které byly nad příkazem vykonány, seřazených vzestupně podle času.

V horní části okna je detail výrobku, který je vyráběn daným příkazem. Kliknutí na náhled v pravé části zobrazí vyskakovací okno s přiblížením daného výkresu. Spodní část okna je různá pro každou operaci. Jsou v něm zobrazeny jak pokyny k operaci a počty již odvedených kusů a zmetků, tak i zaměstnanci, kteří na výrobním příkaze pracují (kteří mají spuštěný automatický časovač).



Obrázek 5.8: Grafický návrh modulu správy materiálu

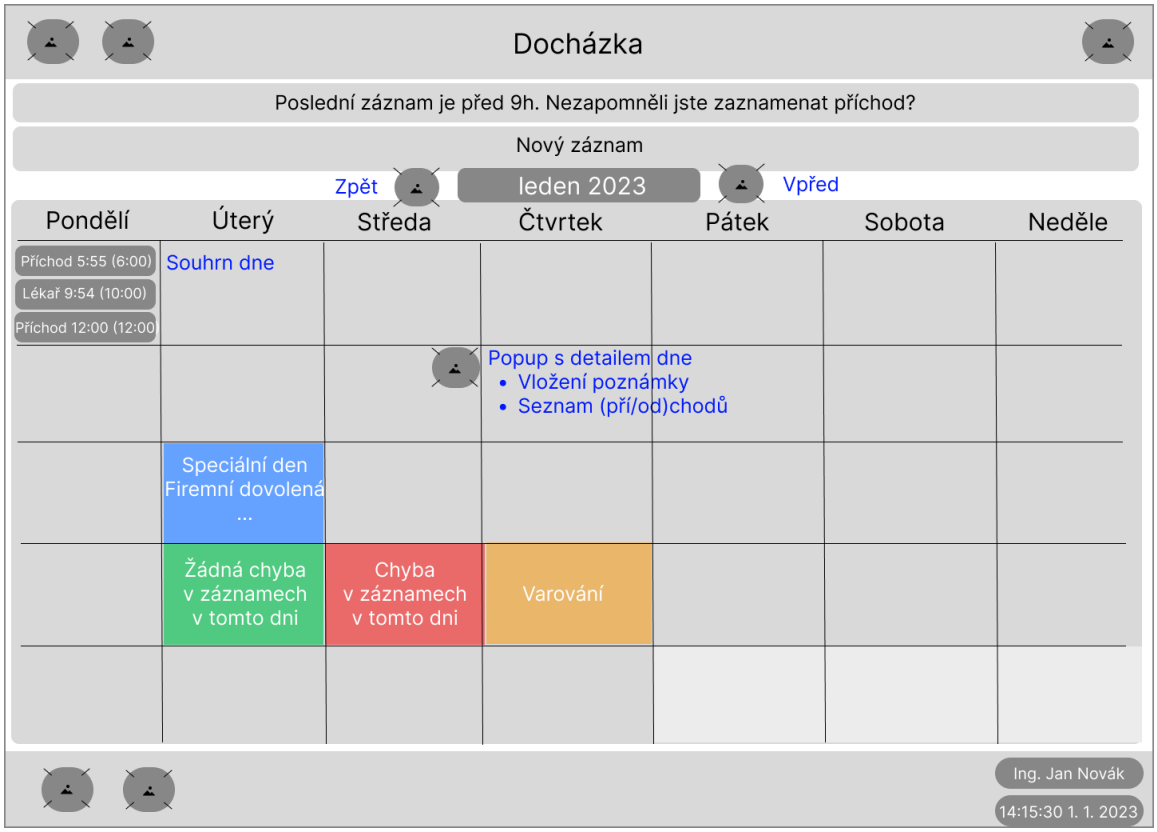
Tlačíko *Materiál* otvírá vyskakovací okno, kde je možné vyskladnit anebo naskladnit materiál, který nemá v systému evidovanou přímou návaznost na výrobní příkaz, viz část 5.5. Tlačítko *Zmetky* otvírá formulář, který uživateli umožní evidovat počet zmetkovitých kusů u dané operace. Poslední tlačítko, *Kontrola*, umožňuje uživatelům prohlížet provedené kontroly k dané výrobní operace, a pokud mají potřebné oprávnění, také kontrolu provést.

Pod tlačítky je sekce s materiály, které mají přímou návaznost na výrobní příkaz. Objem spotřebovaného materiálu je automaticky počítán podle počtu zadaných kusů (kusů, tabulí anebo jiných měrných jednotek) výsledného výrobku, ale uživateli je umožněno tento údaj ručně upravit. Stejně tak i automatický časovač, který je možné spustit anebo vynulovat pomocí tlačítek v spodní části obrazovky, je možné před samotným odvedením operace ručně upravit.

Modul správy materiálu

Obrázek 5.8 zobrazuje návrh obrazovky pro správu materiálu. Tento modul je omezen pro naskladňování a vyskladňování materiálů, nelze z něj tedy naskladnit například hotové výrobky, na které se váží výrobní příkazy.

V horní části obrazovky je detail daného materiálu s možností výběru konkrétní šarže, pokud to materiál umožňuje. Prostřední část obrazovky zobrazuje formulář pro vyskladnění a naskladnění. Po kliknutí na potvrzovací tlačítko se zobrazí vyskakovací okno shrnující informace, které uživatel zadal, a vyžadující potvrzení operace. V dolní části obrazovky je



Obrázek 5.9: Grafický návrh modulu docházky

tabulka s historií pohybů daného materiálu, která se automaticky aktualizuje po provedení jedné z možných operací.

Modul docházky

Grafický návrh modulu pro záznam docházky zaměstnanců je na obrázku 5.9. Hlavní částí modulu docházky je kalendář, který přehledně zobrazuje souhrn příchodů a odchodů přihlášeného zaměstnance v daném měsíci. Pomocí tlačítek vpřed a zpět může uživatel přecházet mezi dalšími měsíci a prohlédnout si historii své docházky. V každém dni jsou zobrazeny konkrétní záznamy. Jeden záznam obsahuje důvod příchodu nebo odchodu, skutečný čas záznamu a zaokrouhlený čas, který systém používá pro další výpočty. Okna kalendáře, představující jednotlivé dny, jsou pro přehlednost podsvícena různými barvami podle významu dne nebo správnosti záznamů. Celkově je použito pět typů zvýrazňujících podbarvení:

- Neexistující den v měsíci (v návrhu vyznačen světle šedou barvou), se kterým nelze nijak interagovat
- Speciální den, označuje například naplánovanou firemní dovolenou anebo státní svátek, vyznačen modře
- Varování, které značí nevýznamnou odchylku od předpokládaného chování (například záznam zapsán z jiného zařízení, než je terminál), v návrhu označen žlutě
- Chyba, například chybějící příchod, označen červeně

- Žádná chyba, zázamy jsou v pořádku, vyznačen zeleně

Po kliknutí na jednotlivá okna kalendáře se zobrazí vyskakovací okno, které ukáže daný den ve větším detailu a umožní uživateli přidat poznámku. Kliknutí na tlačítko *Nový záznam* zobrazí vyskakovací okno s formulářem docházky a předvoleným typem (důvodem) příchodu nebo odchodu. Text v horní části obrazovky uživateli oznámí počet odpracovaných hodin a případně ho upozorní na nekonzistence v záznamech v daném dni.

Modul analýzy dat

Vizuálně je modul analýzy dat, zobrazený na obrázku 5.10, velmi podobný domovské stránce navržené v části 5.5. Návrh vyplývá ze sekce 5.6, kde je popsáno, že pro vizualizaci analyzovaných dat bude využita služba, která umožňuje výsledné grafy vkládat jako iframe[7], se kterými může uživatel interagovat. Přesnější rozložení a zobrazení grafů a dalších podpůrných prvků nutných pro splnění všech požadavků na analýzu vyplyne při samotné implementaci.

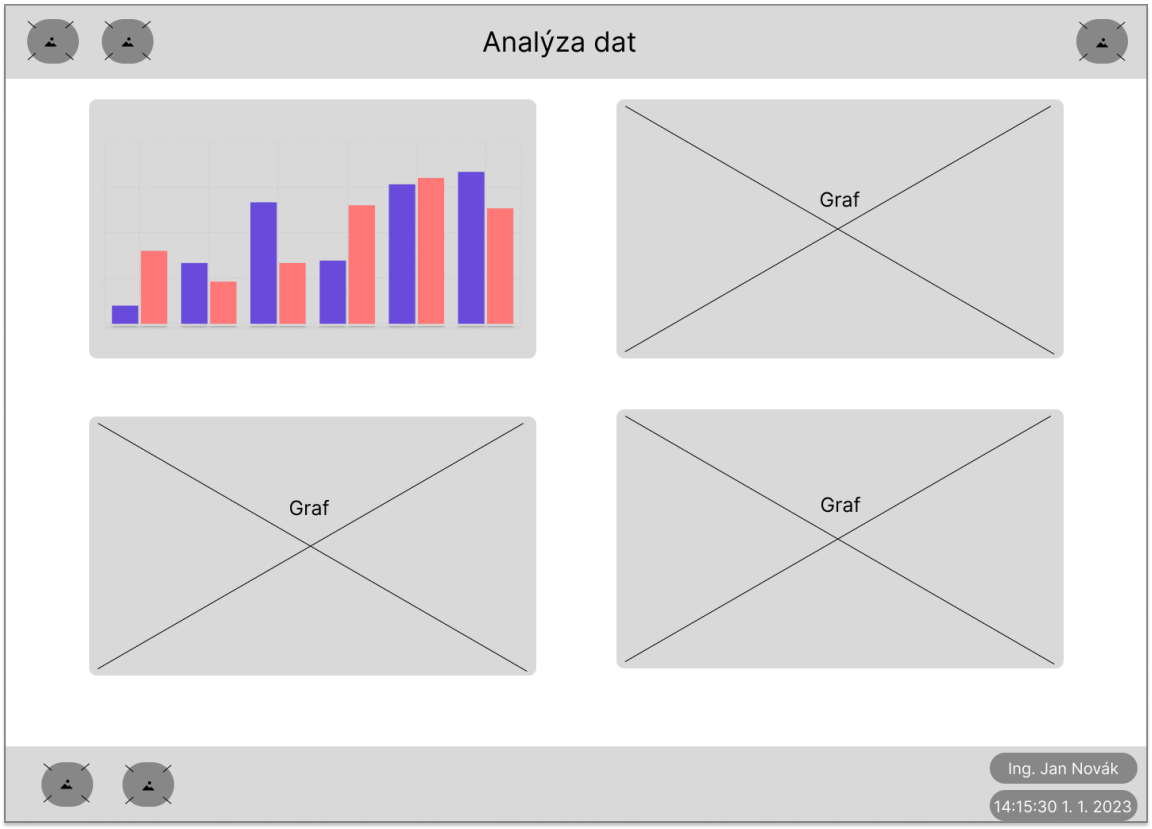
5.6 Vizualizace dat

Z požadavků na analýzu dat vyplývá, že se jedná zejména o statistickou analýzu, bez nutnosti vytvářet mnoho složitých skriptů nebo učících algoritmů. Statistickou analýzu je možné provádět přímo nad databázemi pomocí SQL příkazů. Pokud by byla zátěž serveru příliš velká, je možné periodicky z databáze systému vytvářet datový sklad. Vzhledem k množství uživatelů a předpokládané frekvenci analýzy vytížení systému pravděpodobně nebude tak velké, aby bylo nutné všechna data předzpracovávat, ukládat je v odděleném datovém úložišti nebo v jiné podobě.

Microsoft SQL Server, který využívá jak navržená aplikační databáze, tak databáze informačního systému HELIOS, umožňuje přímé spouštění skriptů nad databází, a to včetně kódů v jazyce Python.[15] Pro případnou prediktivní analýzu je možné vytvořit jednoduché modely přímo v databázi a je tím ponechán dostatek prostoru pro případné zlepšení nebo rozšíření analýzy.

Jako vizualizační nástroj byla vybrána Grafana, což je multiplatformí open-source webová aplikace. Je často využívána pro tvorbu přehledových nástěnek a umí zpracovávat data z různých zdrojů, včetně různých HTTP zdrojů anebo databází. Důvodem, proč byla vybrána zrovna Grafana, je to, že umožňuje vkládat jednotlivé grafy do jiné aplikace, se kterým může uživatel interagovat. Zároveň obsahuje řadu předpřipravených funkcionalit, které je možné využít pro vylepšení aplikace, například zasílání notifikací v případě nějaké události. Díky využití Grafany je možné v případě dalších požadavků firmy analýzu dat dále škálovat a jednoduše přizpůsobovat.

[7] Iframe (inline frame) je webová stránka vložená v jiné webové stránce



Obrázek 5.10: Grafický návrh modulu analýzy dat

Kapitola 6

Implementace

Následující kapitola popisuje způsob implementace jednotlivých částí systému. Protože je systém velmi rozsáhlý, není možné v práci popsat detailně implementaci veškerých jeho částí. Aplikace definuje a využívá přes 140 endpointů[1], skoro 50 různých entit a devět modulů, které se v několika případech člení na další podmoduly. Z toho důvodu byly v jednotlivých podkapitolách věnovaných implementaci serverové a klientské části a analýze dat vybrány pouze ty nejdůležitější principy a řešení největších problémů, které se v průběhu řešení práce vyskytly. Ukázky výsledné aplikace jsou v příloze C.

6.1 Serverová část

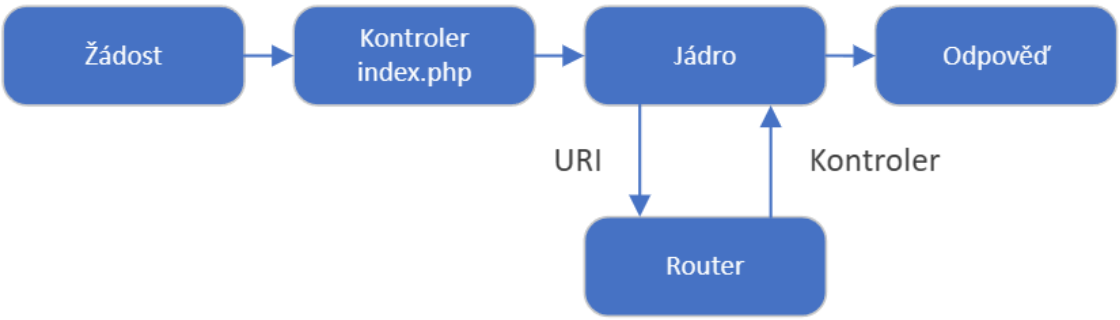
Aplikace je implementována pomocí frameworku Symfony. Využití frameworku umožňuje delegovat důležité úlohy, jakými jsou například mapování URL adres na kontrolery, vkládání závislostí, validace vstupních dat nebo autentizace uživatele, na ověřené a robustní knihovny. Nejdůležitější knihovnou, která společně se základními kontrolery Symfony sloužila pro tvorbu REST API aplikace, je API Platform, která pomocí sady serializačních a deserializačních nástrojů zajišťuje mapování entit na jednotlivé endpointy. Všechny použité knihovny a jejich verze jsou přehledně zobrazeny v příloze v tabulce B.1.

Princip tvorby aplikace

Obecný průběh vyhodnocení jednoho HTTP dotazu, tedy přetvoření Request (žádost) objektu na Response (odpověď) objekt, je zobrazen na obrázku 6.1. Jedná se o základní tok dat frameworku Symfony, který je dále rozvíjen podle konkrétní situace, a to pomocí událostí. Události jsou alfou a omegou tohoto frameworku a umožňují jeho jednoduché rozšiřování. [23]

V obecnosti můžeme říct, že celá aplikace byla vytvořena pomocí sady naslouchačů reagujících na události generované jádrem. V některých případech je využití tohoto vzoru zjevné, a to například ve třídách reagujících na životní cyklus entit popsaných dále v této kapitole, méně zjevnými případy jsou například vlastní procesory, které definují chování endpointu. Procesory jsou v zásadě třídy, které se volají v případě, že vznikne událost dosažení endpointu řízeného knihovnou API Platform, a v průběhu zpracování dat generují další události, na něž je možné reagovat.[17] Stejným způsob jsou vytvářeny všechny ostatní části aplikace.

[1] Za předpokladu, že považujeme různé operace (například PUT a GET) nad jednou URL adresou za dva endpointy, aplikace definuje přesně 144 endpointů, které jsou aktivně využívány v klientské části



Obrázek 6.1: Základní tok dat v Symfony aplikaci

```
#[ORM\Entity(repositoryClass: AttendanceReasonRepository::class)]
#[ApiResource()]
class AttendanceReason
{
    #[ORM\Id]
    #[ORM\GeneratedValue]
    #[ORM\Column]
    private ?int $id = null;

    #[ORM\Column(length: 255, unique: true)]
    private ?string $code = null;

    #[ORM\Column]
    private ?bool $arrival = null;
    ...
```

Výpis 6.1: Příklad tvorby REST API endpointu, třída

Tvorba API

Entity jsou mapovány z tabulek databáze na objekty (ORM) pomocí knihovny Doctrine. Namapované objekty tvoří základ pro REST API. V tom nejjednodušším případě dochází pouze k přemapování entity na tzv. API resource, tedy zdroj endpointu. Příklad takové entity je ve výpisu 6.1, odpověď na GET operaci je znázorněna ve výpisu 6.2. Jedná se o zjendušenou entitu důvodu záznamu v docházce, tedy například odchod k lékaři nebo začátek směny.

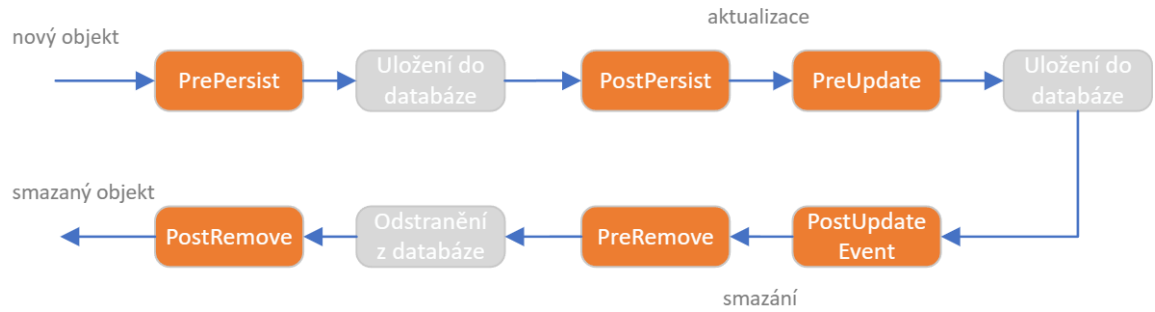
Názvy jednotlivých vlastností jde lehce přizpůsobit pomocí atributů, případně je možné API rozšířit pomocí tzv. getterů (nebo setterů pro POST, PUT a PATCH operace)[2] a pro jednoduché případy obyčejných CRUD[3] operací je tento postup zcela dostačující. Problém ale nastává v případě, kdy chceme implementovat vlastní logiku. Jedním z možných způsobů řešení je tvorba naslouchačů životního cyklu entit, které tvoří zdroj endpointu. V aplikaci

[2]Gettery a settery jsou metody v objektově orientovaném programování, které slouží k získání (getter) nebo nastavení (setter) hodnoty určitého atributu objektu

[3]CRUD neboli create (vytvoření), read (čtení), update (aktualizace) a delete (smazání), jsou základní operace prováděné nad záznamem

```
{
  "@context": "/api/contexts/AttendanceReason",
  "@id": "/api/attendance/attendance_reasons/13",
  "@type": "AttendanceReason",
  "id": 13,
  "code": "leaving_shiftend",
  "arrival": false
}
```

Výpis 6.2: Příklad tvorby REST API endpointu, odpověď na GET



Obrázek 6.2: Životní cyklus entity v Doctrine

jsou využity samozřejmě i další způsoby, jako například vytvoření vlastního procesoru anebo poskytovatele dat [4], ale postup jejich tvorby je komplikovanější.

Životní cyklus entit

Vhodným příkladem využití událostí pro tvorbu vlastní aplikační logiky jsou samotné entity. V případě aplikace je to například uchování historie docházky zaměstnance. Na jednu stranu chceme, aby záznam o docházce byl plně editovatelný a vyžadujeme CRUD funkcionalitu, na stranu druhou je žádoucí udržet historii docházky, abychom mohli odhalit případné podvody.

V průběhu života entity vzniká několik událostí, které jsou znázorněny na obrázku 6.2. V případě editace docházky reaguje systém na všechny události vzniklé po změně v databázi. Zjednodušený naslouchač je zobrazen ve výpisu 6.3.

Na tomto jednoduchém případě je ilustrován princip práce s událostmi v aplikaci. Stejný princip reakce na události je využit například pro ukončení rozpracovaných operací v případě odchodu zaměstnance (událost vytvoření nového záznamu typu odchod), načtení konfigurace systému (událost zavedení jádra), vypláchnutí cache (změna konfigurace systému) a mnoho dalších.

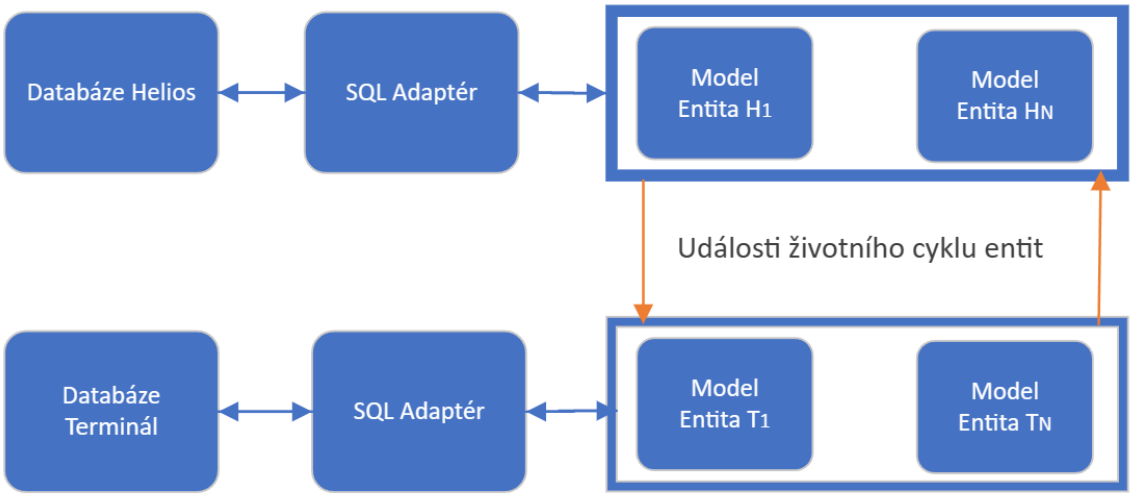
[4] Poskytovatel dat (data provider) je třída, která data dodává specifickám způsobem, který znemožňuje automatické vytvoření endpointu pomocí předpřipravených univerzálních metod.[16]

```
#[AsEntityListener(event: Events::postPersist, method: 'postPersist',
    entity: AttendanceRecord::class)]
#[AsEntityListener(event: Events::postUpdate, method: 'postUpdate', entity:
     AttendanceRecord::class)]
#[AsEntityListener(event: Events::postRemove, method: 'postRemove', entity:
     AttendanceRecord::class)]
class AttendanceRecordLifecycleListener
{
    public function postPersist(
        AttendanceRecord $attendanceRecord,
        PostPersistEventArgs $args
    ) {
        $this->createAttendanceHistory(
            $attendanceRecord,
            AttendanceRecordHistoryType::CREATE
        );
    }

    public function postUpdate(
        AttendanceRecord $attendanceRecord,
        PostUpdateEventArgs $args
    ) {
        $this->createAttendanceHistory(
            $attendanceRecord,
            AttendanceRecordHistoryType::UPDATE
        );
    }

    ...
```

Výpis 6.3: Příklad implementace naslouchače

Práce s databázemi

Aplikace využívá jak své vlastní úložiště, tak databázi systému HELIOS. Obě dvě úložiště jsou spravována pomocí Doctrine a používají Microsoft SQL Server. Ačkoliv se jedná o dvě oddělená úložiště, entity jsou mezi sebou provázané (například každý uživatel terminálu musí mít navázaného zaměstnance). Použité ORM Doctrine podporuje spojení s více databázemi pomocí několika správců entit, ale grafy těchto entit nesmí být propojené, alespoň ne na úrovni automatického mapování pomocí Doctrine.[22] I když Microsoft SQL Server podporuje dotazy mezi databázemi a v Doctrine je možné manuálně entity na sebe navzájem namapovat pomocí anotací obsahujících názvy databáze a tabulek, toto řešení není oficiálně podporováno a nemuselo by v případě budoucích rozšíření obstát. Proto jsou entity synchronizovány manuálně pomocí událostí životního cyklu, jak je znázorněno na obrázku 6.3.

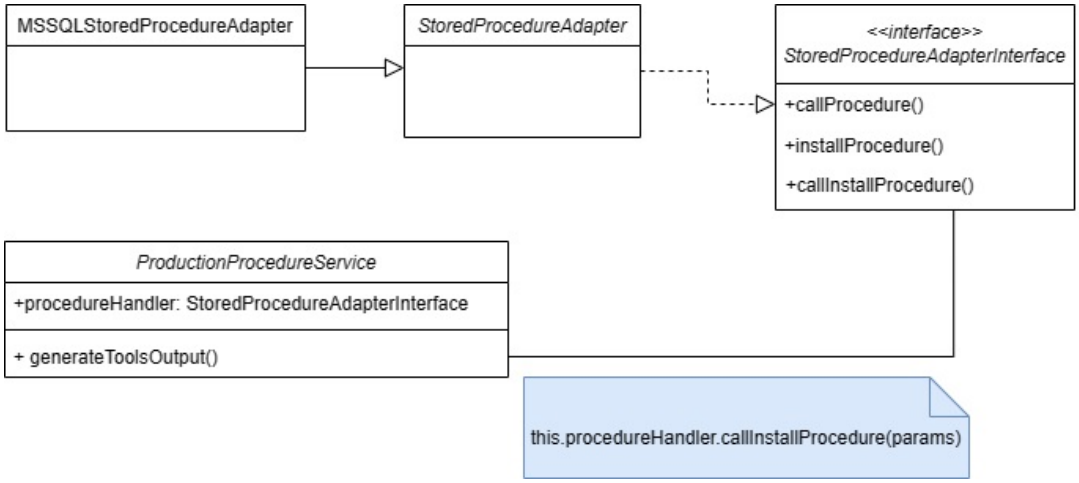


Obrázek 6.3: Práce s více databázemi v aplikaci, vztahy mezi entitami

Kromě základního mapování objektů bylo nutné pro implementaci některých funkcí vytvořit složité SQL dotazy, které nejen že využívají existující procedury systému HELIOS, ale také vytváří dočasné tabulky, které jsou nutné pro správné fungování triggerů[5]. Ačkoliv je velmi nepravděpodobné, že by byl změněn typ databáze, byl tento požadavek implementován pomocí vzoru adaptér. Využití tohoto vzoru umožňuje aplikaci do budoucna jednoduše rozšířit bez nutnosti zásahu do aplikační logiky, a to i v případě změny SQL nebo záměny typu databáze.[1]

Implementace volání procedur ve zjednodušené formě je zobrazena na obrázku 6.4. Konkrétní implementace adaptéru je vybírána za běhu aplikace podle použitého typu databázového spojení. Samotné procedury se načítají z předem definovaných souborů a instalují při jejich prvním spuštění, jak je zobrazeno ve výpisu 6.5. V případě radikální změny v chování systému HELIOS je nutné upravit pouze SQL soubor s procedurou pro danou implementaci adaptéru.

[5]Trigger neboli spouštěč je v kontextu databází speciální funkce nebo procedura, která je automaticky aktivována při určitých událostech v databázi, například při vložení, aktualizaci nebo odstranění záznamu



Obrázek 6.4: Práce s více databázemi v aplikaci, implementace volání procedur

```
app/
└── src/
    └── sql/
        └── Doctrine_DBAL_Platforms_SQLServer2012Platform
            ├── terminal_finishProductionOrder.sql
            ├── terminal_generateMaterialOutputFromOrder.sql
            └── terminal_warehouseMaterialMovement.sql
```

Obrázek 6.5: Soubory definující složité SQL procedury systému

6.2 Klientská část

Klientská část aplikace byla vytvořena podle grafických návrhů ze sekce design:client za použití javascriptového frameworku React[6] a vývojového prostředí Vite[7]. Pro usnadnění vytvoření uživatelského rozhraní byly použity knihovny Tailwind, Material Tailwind a Material UI. Všechny použité knihovny mimo vývojářských závislostí jsou zobrazeny v příloze v tabulce B.2.

Volání API

Aplikace pro práci s HTTP dotazy využívá vestavěnou funkci fetch jazyka JavaScript, která umožňuje provádět asynchronní dotazy na server. Pro efektivní řízení stavu dat a načítání z API je implementován vlastní hook[8] useDataFetcher, který využívá knihovnu React Query. React Query je knihovna určená pro práci s asynchronními daty v React aplikacích. Poskytuje jednoduché API pro správu stavu dat na straně klienta, včetně možnosti automatického načítání, ukládání do mezipaměti a obnovení dat.

Vzhledem k rozsahu aplikace, počtu různých typů entit a velikosti některých objektů (počtu navázaných a zanořených entit) nebylo vhodné používat pouze jednoduché volání

[6] https://react.dev/

[7] https://vitejs.dev/

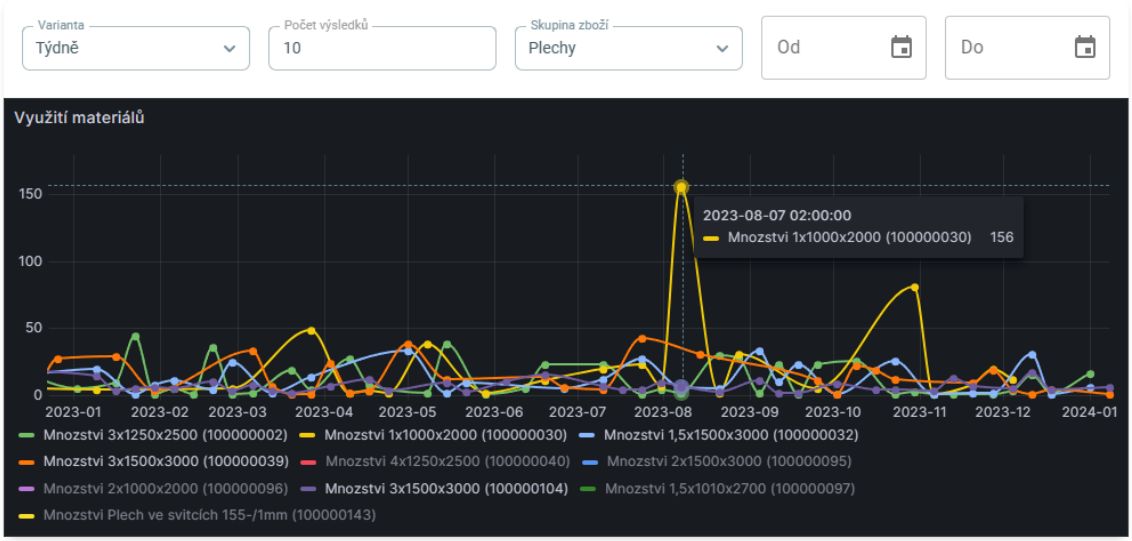
[8] Hook je funkcionalita frameworku React, která umožňuje použití lokálního stavu uvnitř funkčních komponent. Díky tomu je možné definovat jednotlivé komponenty jako znovupoužitelné funkce, které jsou odlišeny vnitřním stavem.[20]

API bez dalších ošetření. Díky tomu, že API splňuje specifikaci OpenAPI[9], bylo možné vytvořit sadu komponent, které zajišťují automatické vytváření objektů v souladu s jejich rozhraními a zejména optimalizují využití zdrojů do takové míry, jaké je v dané komponentě vyžadováno. Jednoduchým příkladem může být objekt ceny produktu, jehož definice a načtení je zobrazeno ve výpisu 6.4

Tento způsob volání API podobně jako GraphQL[10] omezuje počet nutných dotazů, ale zachovává jednoduchost samotné tvorby API v REST rozhraní i v případě komplikovanějších endpointů. Zároveň umožňuje automaticky vytvářet PATCH a POST žádosti z formulářů ve správném formátu. Pro další zrychlení byl v aplikaci také implementován systém cache. Další optimalizace systému nebyly potřebné, ale v případě nutnosti je možné API dodatečně zrychlit pomocí technologie Vulcain[11], která přednačítává zanořená data již při dotazu na první entitu, a při následujících dotazech vrací odpověď rychleji.

6.3 Analýza a vizualizace dat

Vizualizaci dat zajišťuje open-source platforma Grafana. Tato služba je schopná pracovat s různými typy zdrojů dat od databází přes různá API až po soubory. Výsledkem je nástěnka s panely, které je možné vložit do aplikace pomocí iframe. Ukázka vložení je ve výpisu 6.5.



Obrázek 6.6: Ukázka panelu vytvořeného v nástroji Grafana

Panely vložené tímto způsobem jsou responzivní a lze s nimi různě manipulovat. Ukázka panelu je na obrázku 6.6, další náhledy nástěnek je možné nalézt v příloze C.

Datový zdroj

Vizualizační nástroj čerpá data přímo z databáze systému HELIOS. Ačkoliv existuje i možnost využití existující webové služby a dodávání dat prostřednictvím API například v JSON

[9] https://www.openapis.org/

[10] GraphQL je specifikace dotazovacího jazyka, jehož výhodou je (například proti REST API), že při tvorbě dotazu specifikujeme rozhraní dat, která chceme získat. Tím omezujeme počet prováděných dotazů a snižujeme nutný přenos dat [24]

[11] https://github.com/dunglas/vulcain

```
// Mapování
ProductionItemPrice: {
    id: 'number',
    currency: 'string',
    productionItem: 'iri',
    changeDatetime: 'date',
    priceLevel: 'number'
},

interface ProductionItemPrice {
    id: number;
    currency: string;
    productionItem: string;
    changeDatetime: Date;
    priceLevel: number;
}

interface ProductionItemPriceResponse {
    price: string;
    currency: string;
    productionItem: string;
    changeDatetime: string;
    priceLevel: string;
}

// Automaticky načte celý objekt ceny, včetně productionItem (produkt)a
    všech jeho následujících závislostí

// Může vyústit v mnoho volání API
const { builtObjects: priceObjs } = useResponseObjectsBuilder<
       ProductionItemPriceResponse, ProductionItemPrice
    >({ responses: prices })

// Donačte pouze vlastnosti definované v buildProperties, a to jen do
    úrovně 1
// Jedno volání API
const { builtObjects: priceObjs } = useResponseObjectsBuilder<
    ProductionItemPriceResponse, ProductionItemPrice
    >({
      responses: prices,
      buildLevel: 1,
      buildProperties: []
    }
)
```

Výpis 6.4: Ukázka automatického vytváření objektů z definice v klientské části aplikace

```
interface CompaniesPercentageWidgetProps {
    storage: StorageDetail | Storage | StorageDetailResponse |
        StorageResponse;
}

export default function CompaniesPercentageWidget(props:
    CompaniesPercentageWidgetProps) {
    const { storage } = props;

    const companiesUrl = `${import.meta.env.VITE_GRAFANA_URL}/d-solo/
        a1210783-12db-4bf8-b256-9eb829b7c8c3/analyza-materialu?orgId=1&
        panelId=2&var-storage=${storage.id}`;

    return (
        <div className="w-full h-72">
            <iframe src={companiesUrl} className="w-full h-full"></iframe>
        </div>
    );
}
```

Výpis 6.5: Vložení parametrizovaného grafu z Grafany do webové stránky

formátu, byl zvolen přímý přístup do databáze. Díky tomu je dotazování rychlé a tvorba nástěnek jednoduchá, zejména pro komplikované SQL dotazy. Grafana navíc sama umí řešit problémy jako je kupříkladu stránkování.

Aby bylo vytváření nástěnek co nejjednodušší a nejpřehlednější a zároveň šlo aplikaci dostatečně zabezpečit, je databázová struktura systému HELIOS rozšířena o několik pohledů a funkcí vracející tabulky vytvořených speciálně pro potřeby Grafany. Tyto pohledy, v případě parametrizovatelných dotazů pak funkce, poskytují základní datovou strukturu pro vizualizaci dat. Příklad takové funkce a jejího volání prostřednictvím vizualizačního nástroje je ve výpisu 6.6. Tato funkce vrací parametrizovatelný přehled odpracovaných hodin zaměstnance na jednotlivých stanovištích, jak je zobrazeno na obrázku 6.7. Všechny zdrojové kódy funkcí a pohledů jsou k dispozici v příloze D.

Předzpracování dat

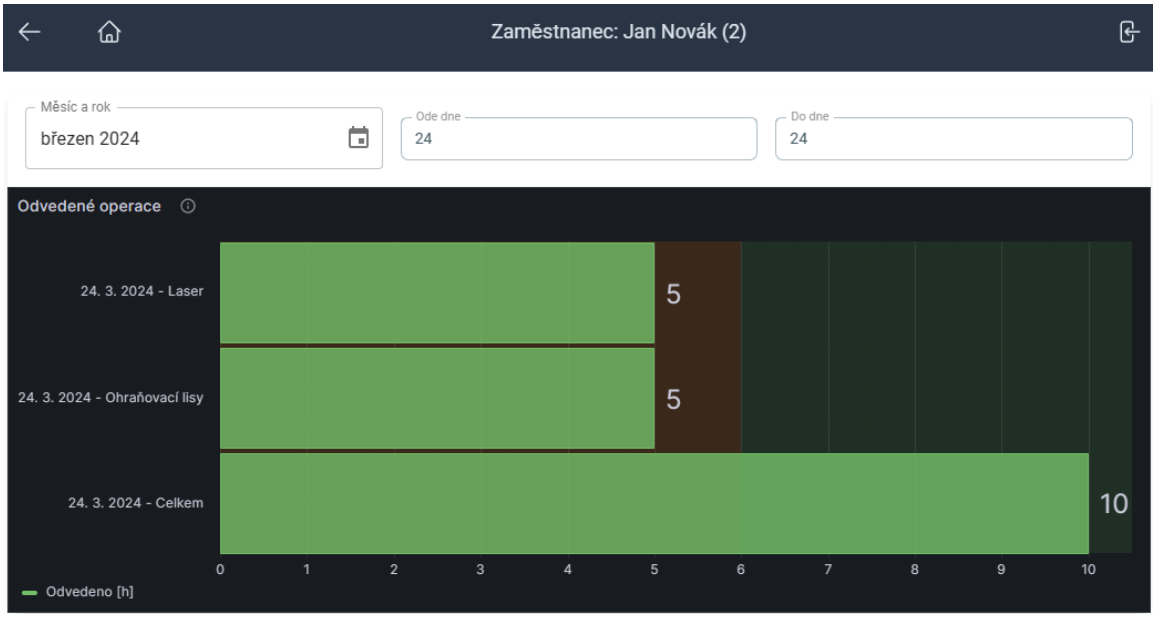
Většina panelů s vizualizacemi je vytvořena pomocí SQL dotazů. V rámci analýzy dat je ale potřebné počítat průměrnou dobu trvání jednotlivých operací na všech výrobních příkazech a jejich průměrnou cenu a zároveň tato data očistit od významně odlišných hodnot, které vznikají špatným používáním systému. Tento výpočet byl pro SQL dotaz, byť i v podobě materializovaného pohledu[12], příliš složitý. Z toho důvodu byl vytvořen jednoduchý skript v jazyce Python.

Skript pro každou operaci vypočítává průměrnou chybovost (zmetkovitost) výrobků a celkový počet vyrobených čistých i zmetkovitých kusů. Poté normalizuje čas potřebný na výrobu jednoho kusu pomocí z-skóre a odstraňuje odlehlé hodnoty. Nakonec spojuje

[12]Data v materializovaném pohledu jsou podobně jako v klasickém tvořena pomocí SELECT dotazu nad existujícími tabulkami, jsou ale uložena v paměti, takže se výsledný pohled počítá pouze při aktualizaci dat, a ne při každém dotázání se na něj [19]

```
-- Analyza zamestnancu - Odvedene operace
CREATE FUNCTION Tabx_Analyza_Zamestnanci_Operace (
    @employee INT,
    @year INT,
    @month INT,
    @dayFrom INT,
    @dayTo INT
)
RETURNS TABLE
AS
RETURN (
    SELECT
        CONCAT(DatumUkonceniOp_D, '. ', DatumUkonceniOp_M, '. ',
            DatumUkonceniOp_Y) AS 'Datum',
        CONCAT(DatumUkonceniOp_D, '. ', DatumUkonceniOp_M, '. ',
            DatumUkonceniOp_Y, ' - ', COALESCE(p.Nazev, 'Celkem')) AS 'Datum
             - Operace',
        SUM(Sk_cas_S)/3600 AS 'Odvedeno [h]',
        COALESCE(p.Nazev, 'Celkem') AS 'Nazev'
    FROM
        TabPrikazMzdyAZmetky tpmz
    LEFT JOIN TabCPraco p ON p.ID = tpmz.IDPracoviste
    WHERE
        Zamestnanec = @employee AND
        DatumUkonceniOp_Y = @year AND
        DatumUkonceniOp_M = @month AND
        DatumUkonceniOp_D >= @dayFrom AND
        DatumUkonceniOp_D <= @dayTo
    GROUP BY
        GROUPING SETS ((DatumUkonceniOp_Y, DatumUkonceniOp_M,
            DatumUkonceniOp_D, p.Nazev), (DatumUkonceniOp_Y,
            DatumUkonceniOp_M, DatumUkonceniOp_D))
)
-- SELECT * FROM Tabx_Analyza_Zamestnanci_Operace([[employee]], [[year]],
    [[month]], [[dayFrom]], [[dayTo]]) ORDER BY 'Datum'
```

Výpis 6.6: Ukázka způsobu vizualizace a analýzy dat



Obrázek 6.7: Ukázka panelu vytvořeného pomocí dotazu ve výpisu 6.6

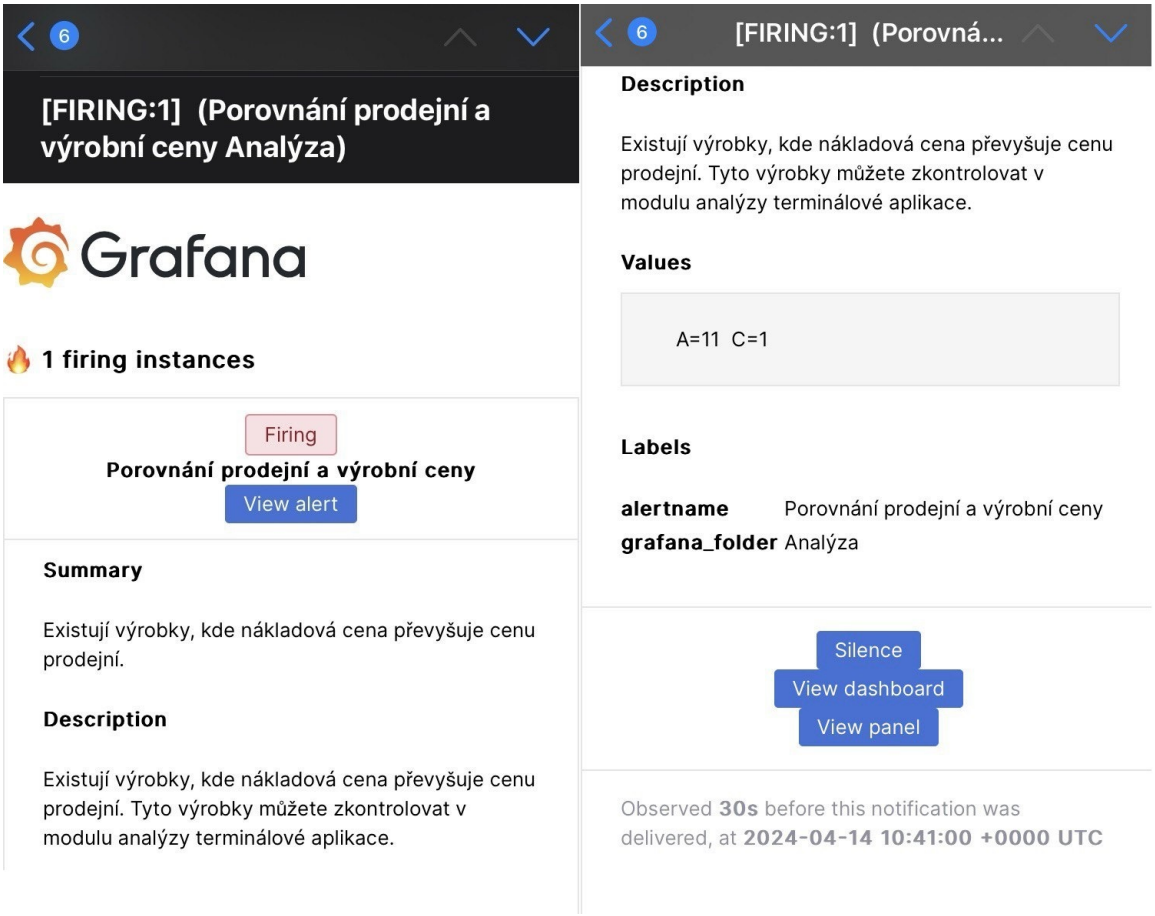
výsledné informace o průměrné zmetkovitosti, času i ceny výroby s původními daty a ukládá výsledky zpět do databáze. V databázi jsou pak k dispozici tyto informace pro všechny výrobní operace každého výrobku:

- Procentuální i celková zmetkovitost
- Počet vyrobených kusů
- Průměrná cena za operaci na jeden kus výsledného výrobku, pokud neuvažujeme zmetky
- Průměrný čas připadající na danou operaci na jeden kus výsledného výrobku
- Průměrná cena za operaci s ohledem na vzniklé zmetky
- Minimální a maximální čas i cena operace, které spadají do vypočítaného z-skóre (například pro zjištění, jestli se daný pracovník pohybuje mimo akceptovanou dobu)

Varování zasílané Grafanou

Grafana umožňuje zasílání notifikací na základě sady nakonfigurovaných pravidel. Notifikace je možné zasílat na různé platformy, například na e-mail, Discord, Slack, Telegram a mnoho dalších. Pravidla jsou tvořena skupinami kritérií, na základě nichž se určuje stav daného pravidla a adekvátní reakce na stav. Pravidla je možné seskupovat do skupin a každé ze skupin přiřadit jiné nastavení, jako například frekvence vyhodnocování a na jakou platformu zasílat upozornění.[18]

V diplomové práci bylo nakonfigurováno jedno pravidlo, které hlídá poměr výrobní a prodejní ceny výrobku. Vychází z cen zahrnujících zmetkovité kusy předpočítaných pro jednotlivé operace, jak bylo popsáno v sekci 6.3. Toto pravidlo se vyhodnocuje jednou denně kromě víkendů a v případě přiblížení se výrobní ceny na devadesát procent ceny prodejní



Obrázek 6.8: E-mailová notifikace zasílaná Grafanou

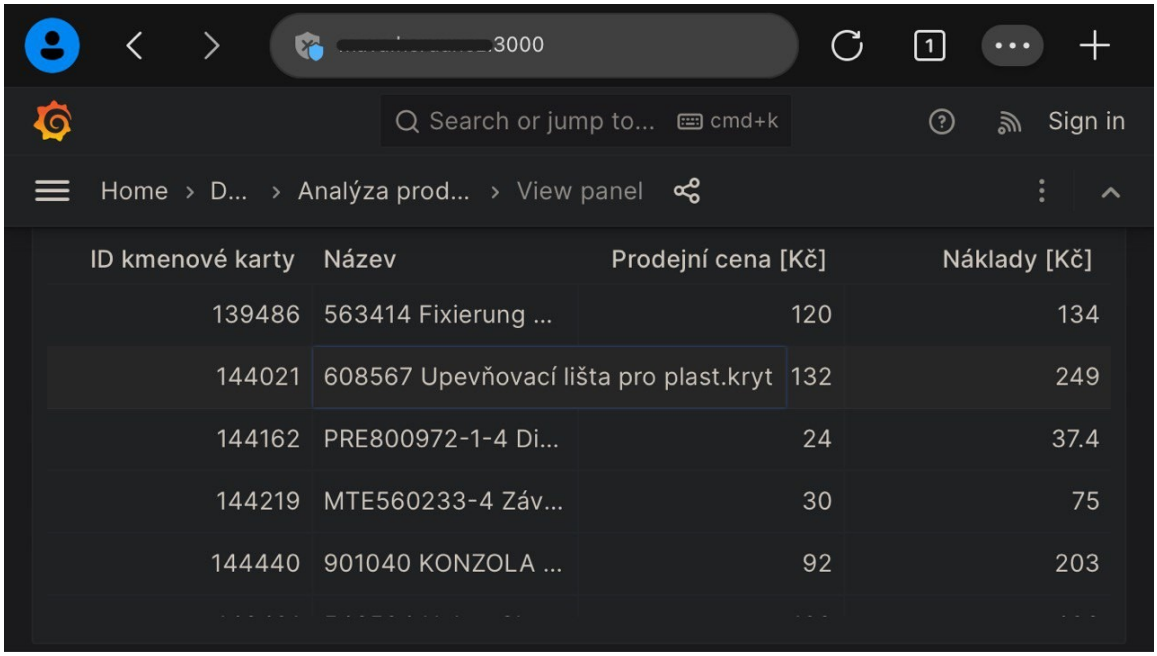
u alespoň jednoho výrobku zasílá Grafana e-mailovou notifikaci administrátorovi systému. E-mailová notifikace je zobrazena na obrázku 6.8. Pokud má adresát e-mailu dostatečná práva, může se z obdrženého e-mailu dostat na panel, který přehledně zobrazuje všechny výrobky, u nichž došlo ke spuštění pravidla. Panel je zobrazen na obrázku 6.9.

Zabezpečení

V náhledu panelu na obrázku 6.6 i v ukázce kódu vložení 6.5 je vidět, že grafy je možné parametrizovat. Parametry se předávají pomocí URL adresy a nahrazují konstanty v SQL dotazech. Tento způsob je sice efektivní, ale vystavuje aplikaci riziku SQL injection, pokud by k panelu dostala přístup neoprávněná osoba. Protože Grafana přistupuje přímo k databázi systému HELIOS, je nutné přístup k datům omezit na nejnutnější minimum.

Z toho důvodu je nutné vytvořit unikátního uživatele, který nejen že bude mít pouze čtecí práva, ale zároveň bude moci přistupovat pouze k několika vybraným funkcím a pohledům, které byly vytvořeny na míru systému:

```
CREATE LOGIN [GrafanaUser] WITH PASSWORD = 'ChangeMe';
CREATE USER [GrafanaUser] FOR LOGIN [GrafanaUser];
GRANT SELECT ON [Tabx_Analyza_Produkty_Vyvoj_prodeje] TO [GrafanaUser];
GRANT SELECT ON [Tabx_Analyza_Produkty_Vyvoj_ceny] TO [GrafanaUser];
```



Obrázek 6.9: Ukázka panelu zobrazujícího špatně naceněné produkty

...

Kapitola 7

Testování

Testování systému probíhalo na několika úrovních. Tou první, úplně nejzákladnější, je statická analýza kódu, která dokáže odhalit chyby, které bychom jiným způsobem testování ladili jenom velmi těžko, například překlepy, špatné užití operátorů jako ?? nebo && a špatné podmínky vedoucí k nedosažitelným částem kódu. Tento způsob testování byl prováděn průběžně pomocí knihovny PHPStan[1]. Knihovna umožňuje testování od úrovně nula až po úroveň deset, kdy nula je nejméně striktní. Aplikace byla testována a opravena až do úrovně pět. Chyby vyšších úrovní často vychází z knihoven třetích stran, například definice funkcí rozhraní.

Dalším krokem testování bylo automatizované testování pomocí nástroje PHPUnit[2]. Toto testování zahrnovalo funkční testy, jako například živost endpointů. Kromě automatizovaného testování bylo provedeno i uživatelské testování. Uživatelské testování zahrnovalo zapojení skutečných uživatelů do testování aplikace za účelem získání zpětné vazby ohledně použitelnosti a přívětivost uživatelského rozhraní a jeho funkčnosti.

7.1 Automatizované testování

Pro automatizované testování byl využit nástroj PHPUnit. Pomocí tohoto nástroje je možné jednoduše spouštět sadu testů, nebo pouze některé z nich. Při tomto druhu testování byly využity databáze oddělené od výsledného systému. Terminálovou databázi automaticky plní migrační nástroj Doctrine pomocí tzv. fixtures, což jsou třídy generující základní sadu dat určenou zejména pro testování nebo prvotní nastavení aplikace.[21] V tomto kroku se databáze naplní základní konfigurací, vytvoří se uživatelé, důvody odchodů a příchodů pro docházku. Druhá databáze je kopie databáze systému HELIOS z ostrého provozu zadavatelské firmy. PHPUnit databázi automaticky obnoví do tohoto výchozího stavu po každém jednotlivém testu, nehledě na to, kolik se jich vykoná. Tím je zajištěno, že testy na sobě nejsou závislé. To znamená, že i když se testuje například deset různých API endpointů najednou, při testování každého z nich dojde k základnímu nastartování jádra Symfony, případnému přihlášení uživatele, a na konci pak také k navrácení databáze do původního stavu před spuštěním testu.

[1] https://phpstan.org/
[2] https://phpunit.de/index.html

Jednotkové testy

Jednotkovými testy byla otestována služba AttendanceTimeCalculationService, která má za úkol počítat korektní časy docházky zaměstnanců z evidovaného času příchodu nebo odchodu. Nutnost výpočtu vychází ze zadání v sekci 2.2 a výpočet se řídí těmito pravidly:

- Typ zaokrouhlení definuje část hodiny, na kterou se zaokrouhluje. Typ 2 odpovídá půlhodině (časy hh:00 a hh:30), typ 4 čtvrthodině (hh:00, hh:15, hh:30, hh:45) atd., kde hh je libovolná hodina.
- Příchody zaměstnanců se zaokrouhlují na nejbližší nižší hodnotu definovanou typem zaokrouhlení, pokud spadají do rozmezí nejbližší nižší hodnota plus rezerva, v opačném případě na hodnotu nejbližší vyšší
- Odchody se naopak zaokrouhlují na nejbližší nižší hodnotu, pokud nedojde k zaokrouhlení v rámci rezervního období na hodnotu nejbližší vyšší

Tato služba zároveň zahrnuje do výpočtu celkové odpracované doby povinné přestávky zaměstnanců. Pokud zaměstnanec pracuje déle, než povoluje stanovený počet minut, odečte z celkové odpracované doby přestávku. Testy obou funkcí je možné spustit následovně:

```
php ./vendor/bin/phpunit --group AttendanceTimeCalculationServiceTest

Testing
..                                  2 / 2 (100%)

Time: 00:00.016, Memory: 10.00 MB

OK (2 tests, 10 assertions)
```

Integrační testy

Pro otestování funkčnosti API a nejdůležitějších funkcí systému byla vytvořena sada integračních testů.

API testy Vzhledem k rozsáhlosti systému nebyly testovány všechny možné endpointy, ale pouze vybrané. Mezi ně patří všechny metody, které poskytují třídy AttendanceReason (důvod příchodu nebo odchodu), AttendanceRecord (záznam o docházce) Document (oznámení přiřazená zaměstnanci). Bylo otestováno následující:

1. Oprávněné získání kolekce i konkrétní položky AttendanceReason
2. Oprávněné vytvoření, aktualizace i smazání entity AttendanceReason
3. Neoprávněné vytvoření, aktualizace a smazání položky AttenadanceReason (úspěšným scénářem je selhání operace)
4. Vytvoření záznamu o docházce pomocí čísla čipu zaměstnance
5. Oprávněné získání docházky přihlášeného zaměstnance
6. Oprávněné manuální vytvoření, úprava a smazání záznamu v docházce

7. Oprávněné získání kolekce novinek
8. Oprávněné získání kolekce novinek přihlášeného uživatele (pro případ notifikací o nepřečtených oznámeních)
9. Neoprávněné získání kolekce novinek přiřazených zaměstnanci (úspěšným scénářem je selhání operace)

Celkem 16 testů je možné spustit následovně:

```
php ./vendor/bin/phpunit --group ApiTest

Testing
................                                  16 / 16 (100%)

Time: 00:01.278, Memory: 54.50 MB

OK (16 tests, 95 assertions)
```

Odvádění výroby Testování procesu odvádění výroby zahrnovalo odvádění jak vyrobených kusů (služba ProductionOutputPiecesMadeService), tak i kusů zmetkovitých (nepovedených, služba ProductionOutputPiecesDefectiveService). Byly otestovány tyto scénáře:

1. Jednoduché odvedení výroby, a sice odvedení takového počtu kusů u takové operace, jejichž kombinace nevyžaduje další zásahy, jako například kaskádní generování vyrovených kusů na předcházejících operacích (typicky první operace na výrobním příkaze)
2. Kaskádní odvádění výroby. V tomto případě jde o nutnost systémového odvedení předchozích operací výrobního příkazu, protože se zaměstnanec pokouší odvést více kusů, než vzniklo v předešlém kroku.
3. Realizace výrobní operace. Jedná se o poslední operaci výrobního příkazu. V tomto kroku vzniká příjemka vyráběného zboží a toto zboží je naskladněno.
4. Odvedení operace s navázaným spotřebovaným materiálem (materiál je vyskladněn a spojen s operací)
5. Jednoduché odvedení zmetkovitých kusů
6. Kaskádní odvedení zmetkovitých kusů (nelze odvést více zmetků, než bylo vyrobeno čistých kusů v předešlém kroku)

Všechny testy lze spustit následovně:

```
php ./vendor/bin/phpunit --group ProductionOutputTest

Testing
......                                              6 / 6 (100%)

Time: 00:51.903, Memory: 38.50 MB

OK (6 tests, 71 assertions)
```

Odvádění materiálu Testováním služby WarehouseMaterialMovementService byly otestovány tyto scénáře:

1. Naskladnění zboží spadající do skupiny produktů (nezvýší počet na skladě, to lze pouze výrobou)
2. Naskladnění materiálu (zvýší počet zboží na skladě)
3. Vyskladnění materiálu (sníží počet zboží na skladě)
4. Naskladnění a vyskladnění konkrétní šarže zboží

Testy lze spustit následujícím příkazem:

```
php ./vendor/bin/phpunit --group MaterialMovementTest

Testing
.....                                   5 / 5 (100%)

Time: 04:30.797, Memory: 38.50 MB

OK (5 tests, 101 assertions)
```

7.2 Uživatelské testování

Před zahájením zkušebního provozu byl systém nainstalován na jednom terminálovém zařízení a po zaškolení dán k dispozici k základnímu otestování požadované funkčnosti. Bylo připraveno několik akceptačních testů, kteře ověřují splnění nejdůležitějších požadavků na systém. Tyto akceptační testy zahrnují především ověření různých typů přihlašování, a to přístup pouze z terminálových zařízení a vyžadování hesla pro pracovníky s vyšším přístupem. Také se ověřuje správné chování systému při vyhledávání, naskladňování, vyskladňování, různých typů odváděcích operací a odvádění zmetkovitých kusů. Všechny akceptační testy jsou detailně i se všemi možnými scénáři popsány v příloze E.

Kapitola 8

Závěr

Cílem práce bylo komplexní digitalizování a automatizace klíčových procesů ve strojírenské firmě s důrazem na nahrazení nevyhovujícího řešení odvádění výroby a jeho rozšíření o efektivní správu materiálů a současně optimalizovat využití existujícího informačního systému HELIOS iNuvio. Byla provedena analýza firemních procesů a na základě ní identifikovány jejich nedostatky. Součástí analýzy bylo studium rozhraní informačního systému HELIOS iNuvio, což poskytlo hlubší porozumění jeho funkcionalit a možností.

V reakci na tato zjištění byl navržen nový systém, který se snaží nedostatky řešit. Pro jednotlivé problémové oblasti byly v rámci navrženého systému vytvořeny moduly, které řeší konkrétní nedostatky. Píchací hodiny a manuální výpočet odpracovaných hodin byl nahrazen automatickým modulem docházky. Analytický modul poskytuje hodnotné informace o výdělečnosti firmy, spotřebě materiálů a v neposlední řadě i efektivitě zaměstnanců . Modul výroby a materiálů nahrazuje stávající terminálovou aplikaci a umožňuje zaměstnancům odvádět výrobu a naskladňovat i vyskladňovat materiály. Správa strojů, kontrola výrobních operací i podepisování důležitých dokumentů bylo převedeno do digitální podoby.

Informační systém byl implementován jako webová aplikace, což umožňuje jeho univerzální použití na různých typech zařízení. Rozsáhlé požadavky zejména na způsob autentizace uživatelů z kapitoly 2.1 jsou splněny využitím X.509 certifikátů nainstalovaných na terminálových zařízeních ve výrobních halách. Díky nim a také obsáhlé správě rolí jsou umožněny k systému různé druhy přístupů. Je možné mít uživatele, kteří mají přístup pouze z terminálů, uživatele, kteří se mohou přihlásit pouze čipem a také uživatele, u nichž je vynucen přístup s heslem, a to z libovolného zařízení. Také je možné libovolně omezit přístup pouze k vybraným částem systému.

Výsledná aplikace se skládá z několika celků. Serverová část, která obsahuje REST API pro klientskou část a obstarává komunikaci s databázemi, byla vytvořena v jazyce PHP s použitím frameworku Symfony. Klientská část je vytvořena jako jednostránková aplikace v javascriptovém frameworku React. Vizualizace a analýza dat jsou zajištěny pomocí platformy Grafana, která čerpá data přímo z databáze systému HELIOS. Za tímto účelem byla databázová struktura systému HELIOS rozšířena o několik pohledů a funkcí a byl napsán skript v jazyce Python, který data předzpracovává.

V sekci 7.1 bylo popsáno automatizované testování systému. Z výpisu spouštění testů lze vidět, že všechny automatické testy úspěšně prochází bez chyby. Během jejich tvorby tomu tak nebylo ve všech případech, například jednotkové testy pokrývající funkčnost docházky odhalily drobné odchylky od očekávaného chování, což dokazuje důležitost testů při vývoji informačních systémů. Po ukočení automatizovaného testování bylo zahájeno testování uživatelské pomocí sady akceptačních testů popsaných v sekci 7.2. Kromě ověření funkčnosti

sloužilo toto období pro ohodnocení kvality systému a spokojenosti uživatelů s rozhraním. Bylo vzneseno několik drobných požadavků na změnu uživatelského rozhraní, například podbarvení víkendů v kalendáři nebo automatické zavření vyskakovacích formulářových oken po úspěšném ukončení operace. Tyto požadavky byly do systému doimplementovány a poté byl systém nasazen do zkušebního provozu.

Během zkušebního provozu byla vyzkoušena funkčnost všech částí systému a míra splnění požadavků. Byly odhalené drobné chyby těch částí, které nebyly pokryty akceptačními a ani automatizovanými testy, například při exportu docházky do formátu typu xlsx docházelo k nechtěnému přepisování nesprávných buněk. Tyto chyby byly opraveny, a žádné další, vážnější, se při provozu neprojevily. Všechny požadavky uvedené v kapitole 2 byly splněny. Potenciál k dalšímu rozšíření je zejména v analýze, obzvlášť při delším provozu jak systému HELIOS, tak informačního systému navrženého v této práci. Je možné monitorovat více částí výroby a zasílat další notifikace pomocí Grafany, vytvořit prediktivní modely pro usnadnění nákupu materiálů a mnoho dalšího.

Literatura

[1] Gamma, E., Helm, R., Johnson, R. a Vlissides, J. *Design Patterns: Elements of Reusable Object-Oriented Software*. 37. vyd. Addison-Wesley, 1994. ISBN 0-201-63361-2.

[2] Sodomka, P. a Klčová, H. *Informační systémy v podnikové praxi*. 2. vyd. Computer Press, a.s., 2010. ISBN 978-80-251-2878-7.

[3] Pitt, C. *Pro PHP MVC*. 1. vyd. Apress, listopad 2012. ISBN 978-1430241645.

[4] Flanagan, D. *JavaScript: The Definitive Guide*. 5. vyd. O'Reilly Media, Inc., 2006. ISBN 9780596554477.

[5] Richards, M. *Software Architecture Patterns: Understanding Common Architecture Patterns and When to Use Them*. 1. vyd. O'Reilly Media, Inc., 2015. ISBN 978-1-491-92424-2.

[6] Jones, M., Bradley, J. a Sakimura, N. *JSON Web Token (JWT)* [online]. RFC 7519. RFC Editor, květen 2015 [cit. 2024-04-05]. Dostupné z: http://www.rfc-editor.org/rfc/rfc7519.txt.

[7] Housley, R., Polk, W., Ford, W. a Solo, D. *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile* [online]. RFC 3280. RFC Editor, duben 2002. Dostupné z: http://www.rfc-editor.org/rfc/rfc3280.txt.

[8] Běhálek, M. *Komponenty COM a distribuované aplikace* [online]. 2007 [cit. 2023-10-05]. Dostupné z: http://behalek.cs.vsb.cz/vyuka/pcsharp/text/ch10s01.html.

[9] Asseco Solutions, a. s. *ERP systémy HELIOS* [online]. 2023 [cit. 2023-10-03]. Dostupné z: https://www.helios.eu/.

[10] Asseco Solutions, a. s. *HELIOS iNuvio pro střední firmu* [online]. 2023 [cit. 2023-10-03]. Dostupné z: https://www.helios.eu/helios-inuvio.

[11] Asseco Solutions, a. s. *Výroba HELIOS iNuvio* [online katalogový list]. 2023 [cit. 2023-10-03]. Dostupné z: https://www.helios.eu/files/al-inuvio-vyroba.pdf.

[12] Asseco Solutions, a. s. *Skladová evidence HELIOS iNuvio* [online katalogový list]. 2023 [cit. 2023-10-03]. Dostupné z: https://www.helios.eu/files/sklady-v-inuviu.pdf.

[13] Asseco Solutions, a. s. *HELIOS Space* [online]. 2023 [cit. 2023-10-03]. Dostupné z: https://www.helios.eu/helios-space.

[14] Asseco Solutions, a. s. *Technické požadavky HELIOS iNuvio* [online]. 2023 [cit. 2023-10-03]. Dostupné z: https://www.helios.eu/technicke-pozadavky-helios-inuvio.

[15] Assaf, W., Hansen, D. P. et al. *Introduction to SQL Server Machine Learning Services with Python* [online]. 8. ledna 2023 [cit. 2023-11-22]. Dostupné z: https://learn.microsoft.com/en-us/sql/machine-learning/sql-server-machine-learning-services?view=sql-server-ver16.

[16] Dunglas, K. *API Platform: Data Providers* [online]. 2023 [cit. 2024-05-05]. Dostupné z: https://api-platform.com/docs/v2.6/core/data-providers.

[17] Dunglas, K. *API Platform: State processors* [online]. 2023 [cit. 2024-05-05]. Dostupné z: https://api-platform.com/docs/core/state-processors.

[18] Grafana Labs. *Introduction to Alerting* [online]. 2024 [cit. 2024-04-14]. Dostupné z: https://grafana.com/docs/grafana/latest/alerting/fundamentals.

[19] Hattemer, A. *Understanding Materialized Views* [online]. 24. března 2020 [cit. 2024-02-04]. Dostupné z: https://materialize.com/guides/materialized-views.

[20] Modi, M. *React Hooks: A Beginners Guide* [online]. 2024 [cit. 2024-04-20]. Dostupné z: https://www.loginradius.com/blog/engineering/react-hooks-guide.

[21] Symfony SAS. *DoctrineFixturesBundle Documentation* [online]. 2024 [cit. 2024-03-14]. Dostupné z: https://symfony.com/bundles/DoctrineFixturesBundle/current/index.html.

[22] Symfony SAS. *How to Work with Multiple Entity Managers and Connections* [online]. 2024 [cit. 2024-02-04]. Dostupné z: https://symfony.com/doc/current/doctrine/multiple_entity_managers.html.

[23] Symfony SAS. *Symfony and HTTP Fundamentals* [online]. 2024 [cit. 2024-02-04]. Dostupné z: https://symfony.com/doc/current/introduction/http_fundamentals.html.

[24] The GraphQL Foundation. *Introduction to GraphQL* [online]. 24. března 2024 [cit. 2024-04-20]. Dostupné z: https://graphql.org/learn/.

[25] *Kiosk software* [online]. 5. března 2024 [cit. 2024-05-04]. Dostupné z: https://en.wikipedia.org/wiki/Kiosk_software.

Seznam příloh

Příloha A

Obsah přiloženého paměťového média

Na přiloženém paměťovém médiu se nachází tyto složky a soubory:

- **aplikace** - složka obsahující všechny zdrojové kódy aplikace
- **aplikace\README.md** - soubor s pokyny k instalaci a popisem jednotlivých složek v adresáři
- **aplikace\LICENSE** - licence k software
- **tex** - zdrojové soubory pro vygenerování technické zprávy
- **ukazky** - ukázky aplikace (snímky obrazovky)
- **text.pdf** - technická zpráva
- **text_print.pdf** - technická zpráva určená pro tisk

Příloha B

Seznam knihoven použitých pro tvorbu aplikace

Použité technologie	**Důvod použití**	**Verze**
Symfony	Framework pro backend	7.0.x
Doctrine	ORM	2.18
PHPStan	Statická analýza kódu	1.10
PHPUnit	Automatizované testování	9.6
ApiPlatform	REST API	3.2
metaclass-nl/filter-bundle	Pokročilé filtrování v REST API	3.1
nelmio/cors-bundle	REST API	2.4
icewind/smb	Napojení na SMB v PHP	3.6
lexik/jwt-authentication-bundle	Přihlašování (JWT)	2.20
gesdinet/jwt-refresh-token-bundle	Správa JWT	1.3
phpoffice/phpspreadsheet	Export docházky	2.0
phpdocumentor/reflection-docblock	Jednodušší práce v proměnnými (pole)	5.3

Tabulka B.1: Technologie použité pro tvorbu serverové části

Knihovna	**Důvod použití**	**Verze**
@heroicons/react	Ikony pro React aplikace	2.1.1
@material-tailwind/react	Komponenty ve stylu Material Design, Tailwind	2.1.9
@mui/material	Komponenty využívající Material-UI	5.13.4
@mui/x-date-pickers	Formulářové prvky	6.7.0
dayjs	Práce s daty a časy	1.11.8
formik	Práce s formuláři	2.4.1
yup	Validace dat formuláře	1.2.0
html-react-parser	Zobrazení formátovaného textu z databáze	4.2.5
https	Práce s HTTP dotazy	1.0.0
react-pdf	Práce s PDF soubory	7.3.3
react-query	Knihovna pro správu stavu a dat v React aplikacích	3.39.3
react-router-dom	Routování	6.11.1
i18next	Lokalizace	23.2.6
i18next-browser-languagedetector	Detekce jazyka v prohlížeči	7.1.0
flag-icons	Sady vlajkových ikon	6.11.1
uuid	Uživatelské notifikace	9.0.0
@emotion/react	Závislost Material UI	11.11.1
@emotion/styled	Závislost Material UI	11.11.0

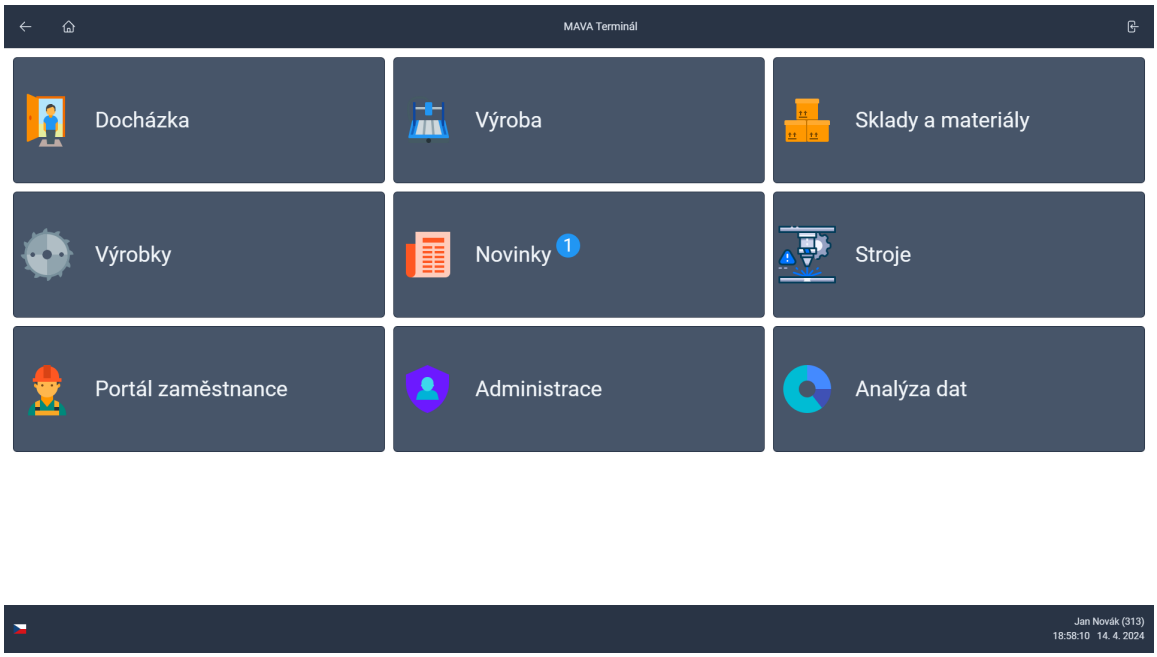
Tabulka B.2: Technologie použité pro tvorbu klientské části

Příloha C

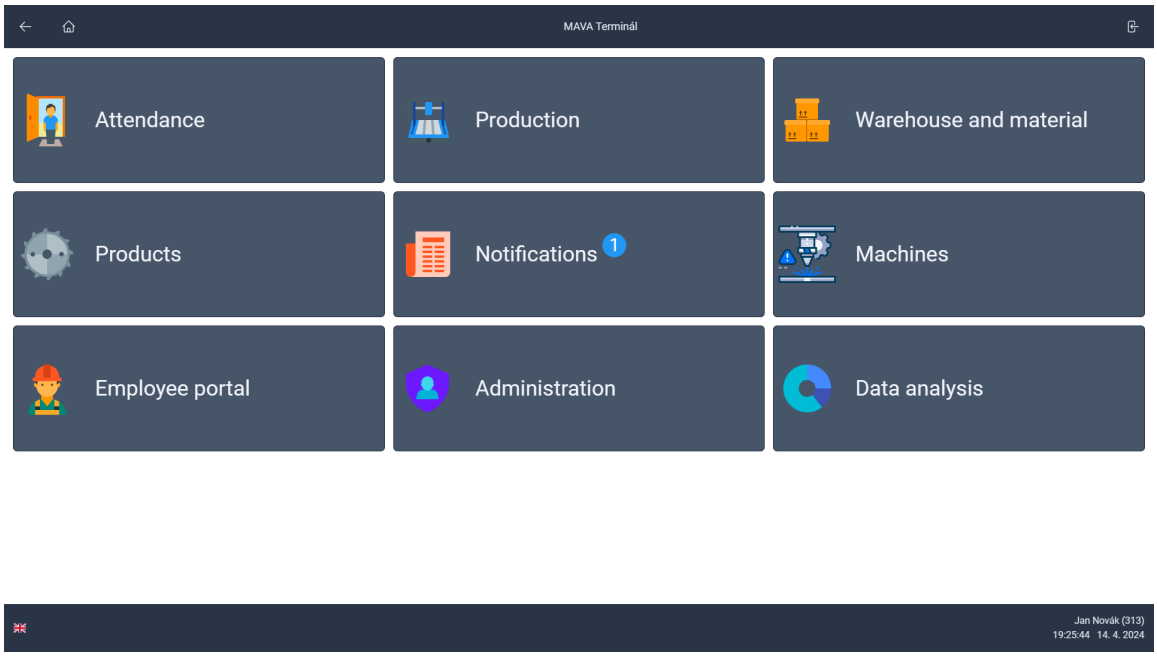
Ukázky systému

V této příloze jsou snímky vybraných obrzazovek. V některých případech bylo nutné data pozměnit a anonymizovat, jako například u konkrétních částek v grafech fakturací.

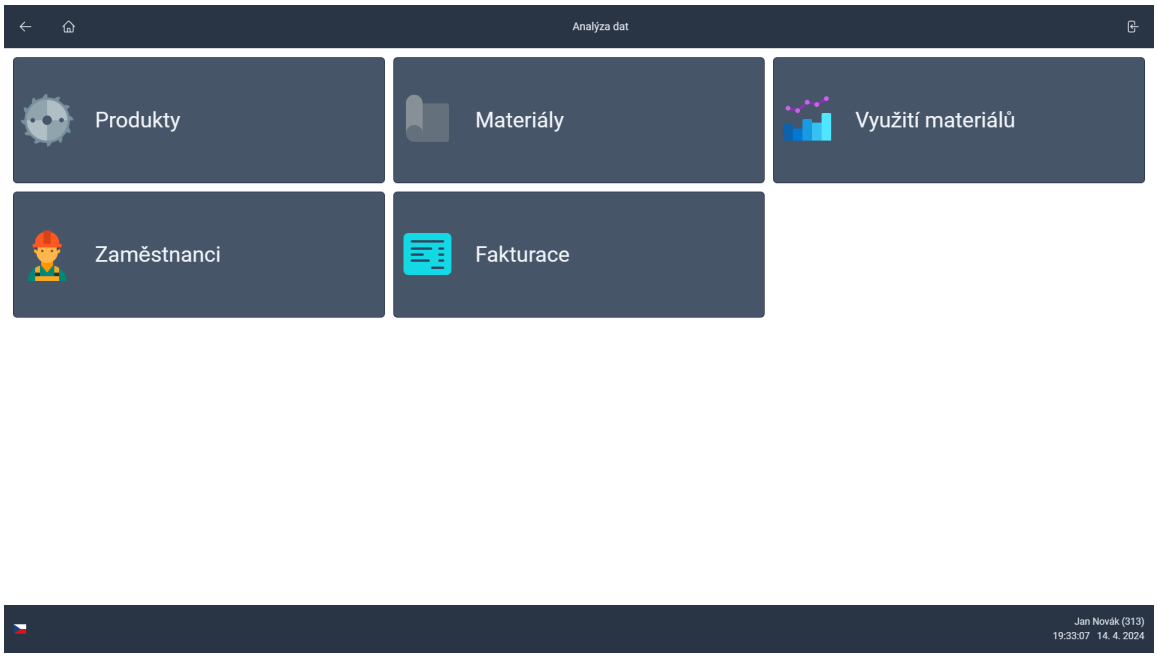
Obrázek C.1: Terminálová aplikace, přihlašovací obrazovka



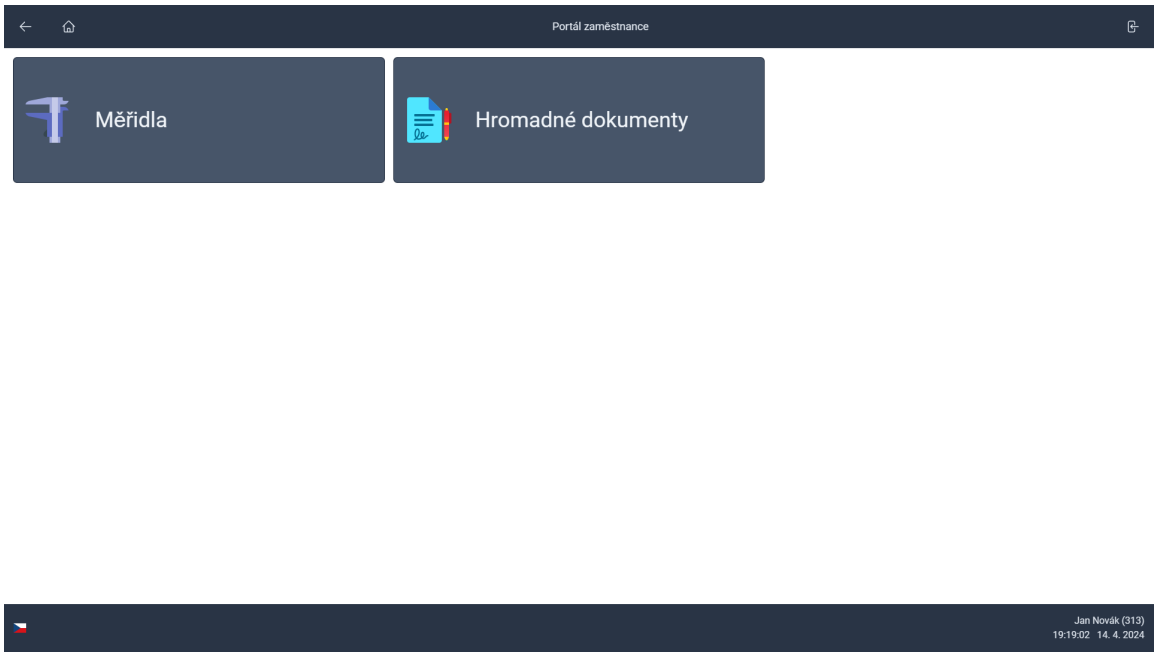
Obrázek C.2: Terminálová aplikace, úvodní obrazovka s moduly



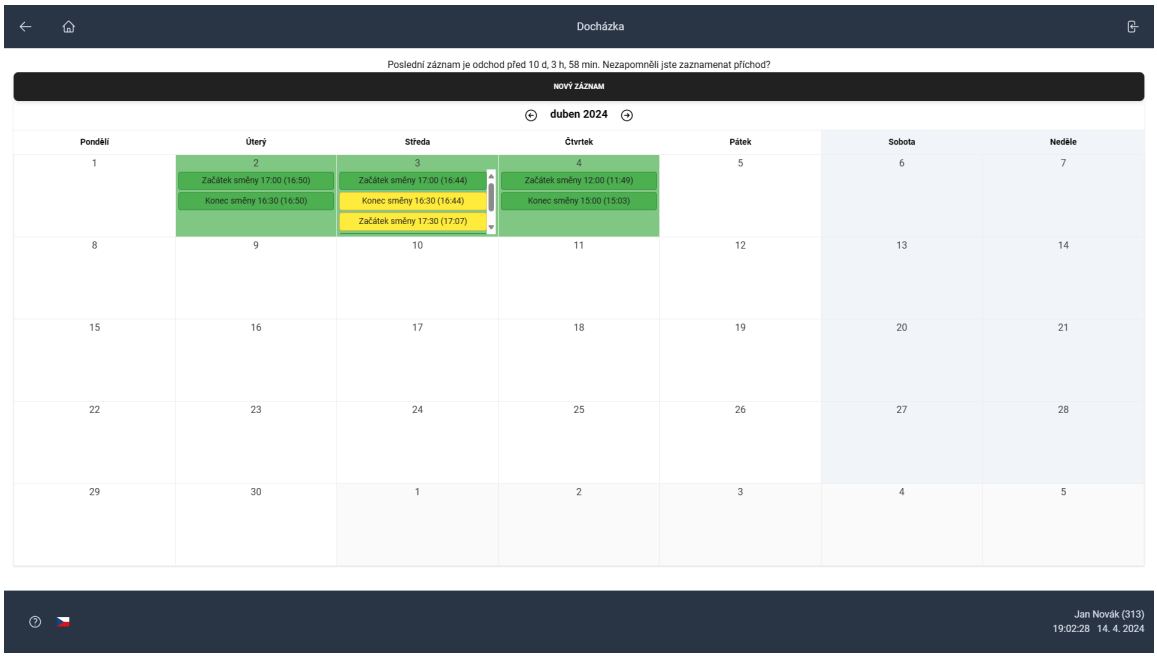
Obrázek C.3: Terminálová aplikace, úvodní obrazovka s moduly, anglický jazyk



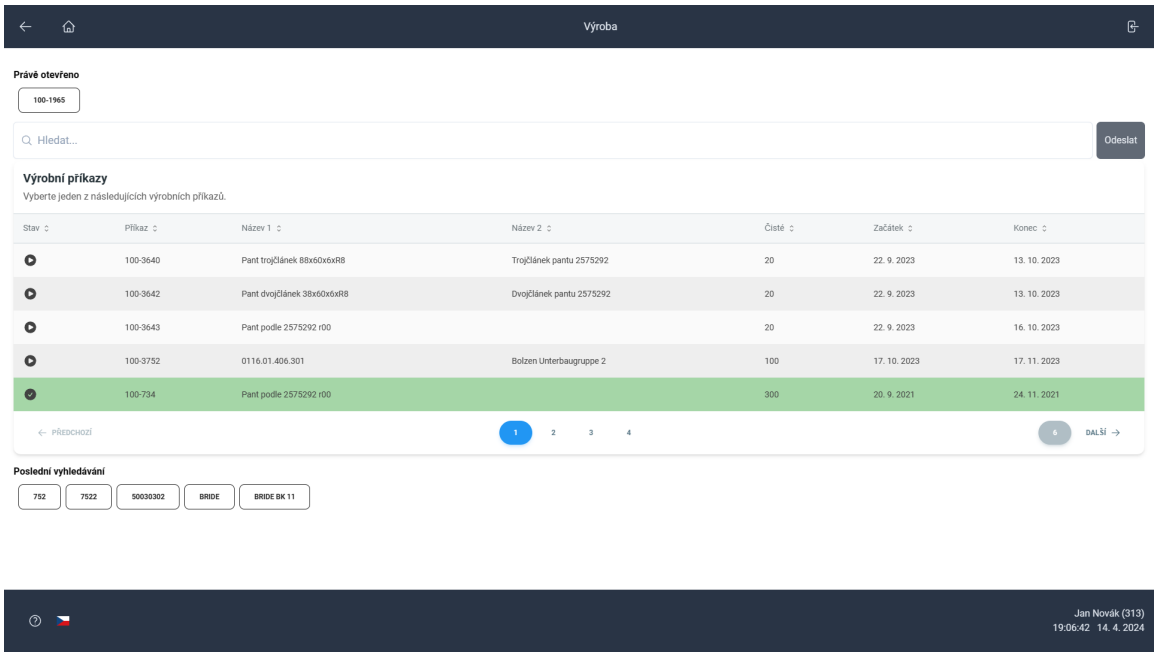
Obrázek C.4: Terminálová aplikace, menu modulu „Analýza dat“



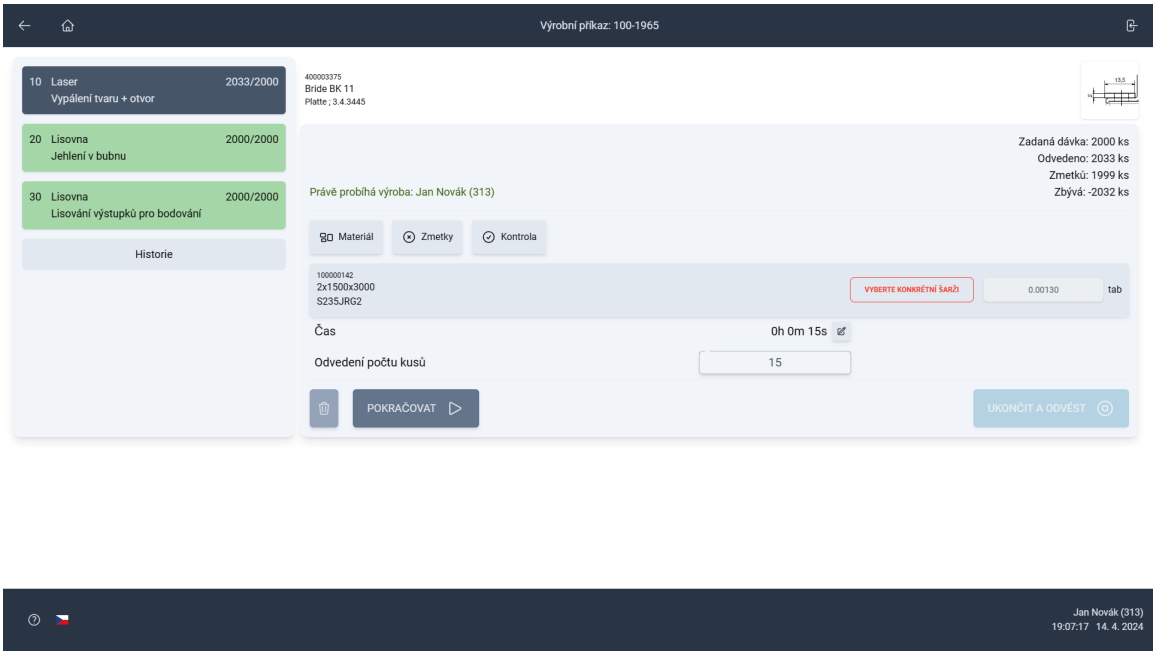
Obrázek C.5: Terminálová aplikace, menu modulu „Portál zaměstnance“



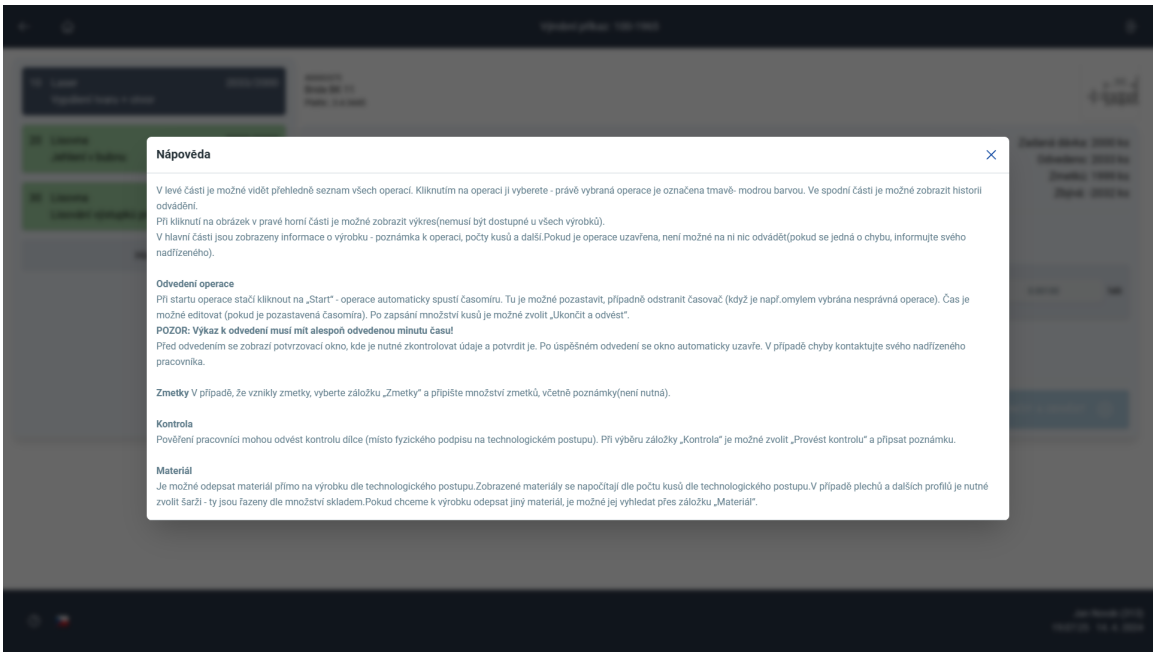
Obrázek C.6: Terminálová aplikace, docházka zaměstnanců



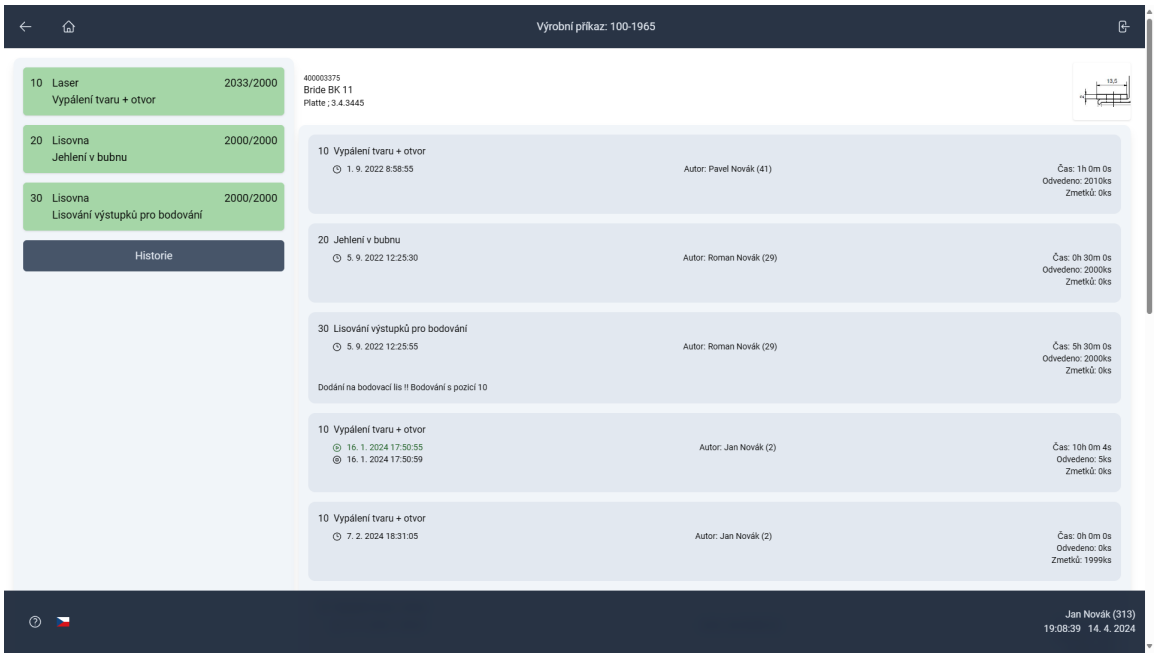
Obrázek C.7: Terminálová aplikace, vyhledávání výrobních příkazů



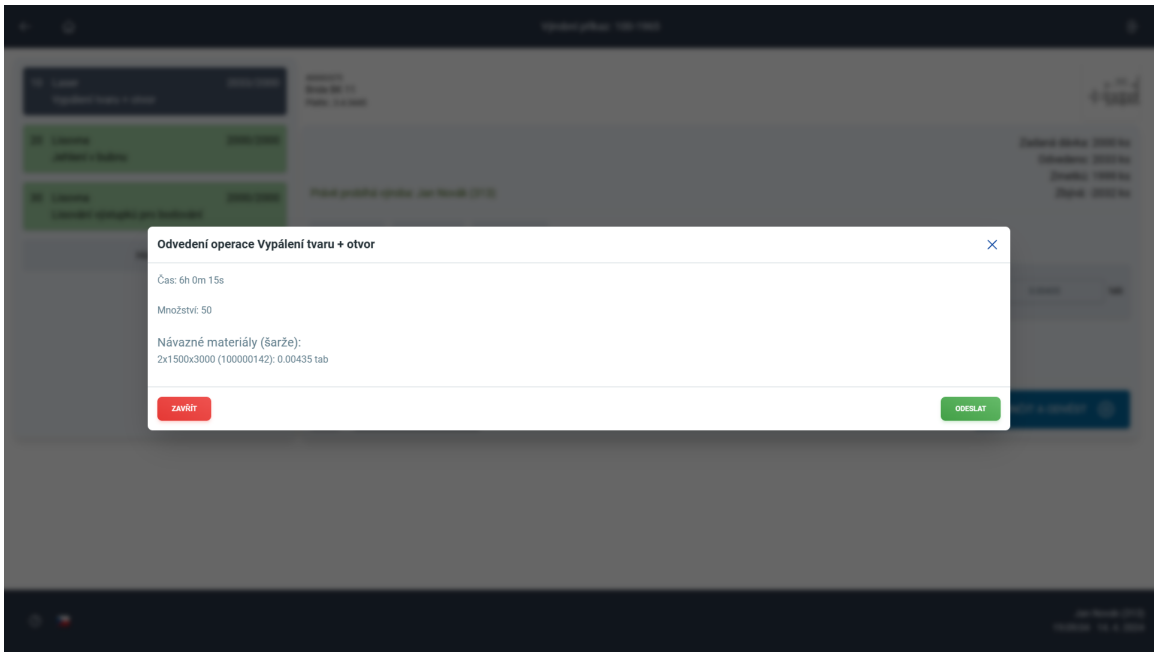
Obrázek C.8: Terminálová aplikace, operace odvádění výroby



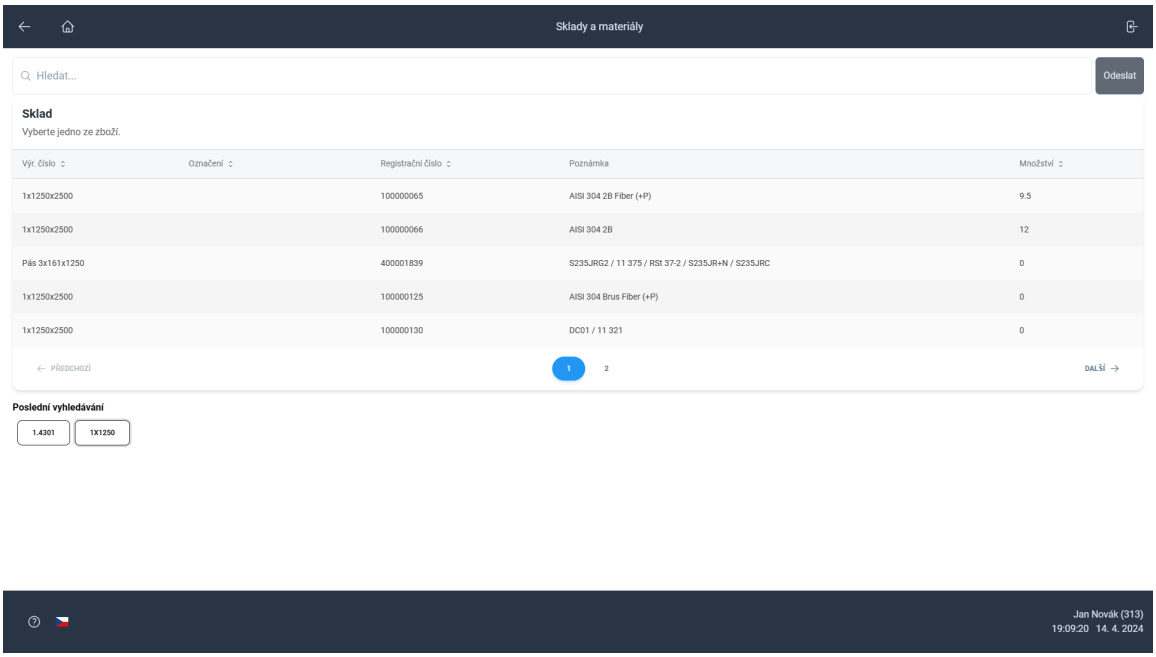
Obrázek C.9: Terminálová aplikace, operace odvádění výroby, nápověda pro uživatele systému



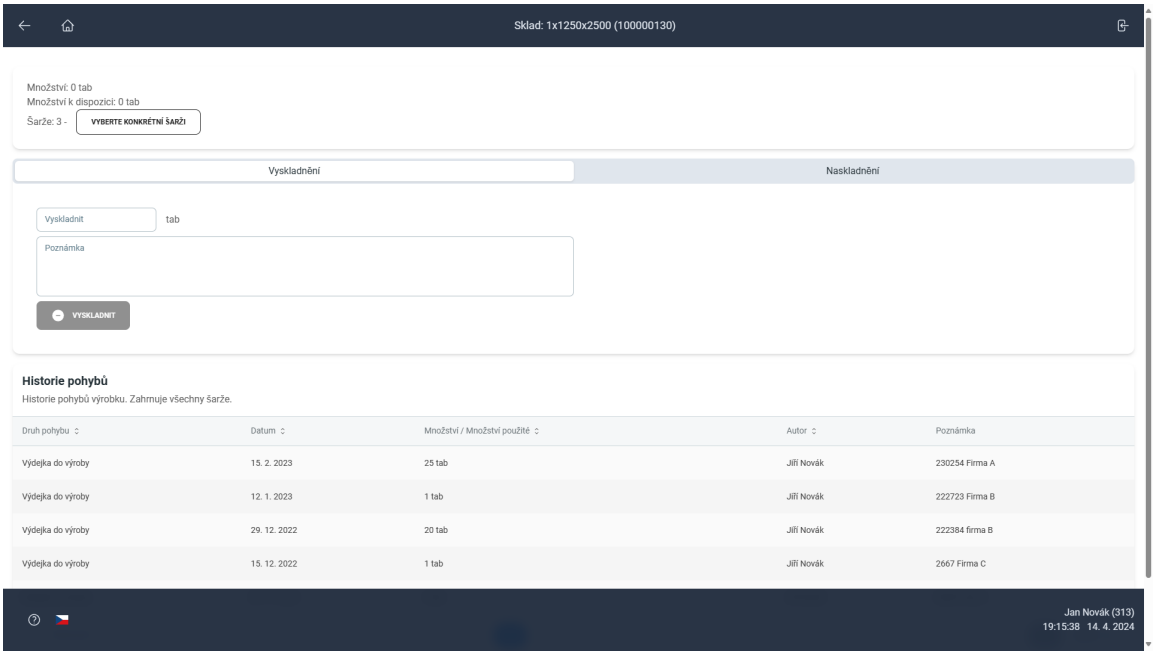
Obrázek C.10: Terminálová aplikace, historie odvádění výrobní operace



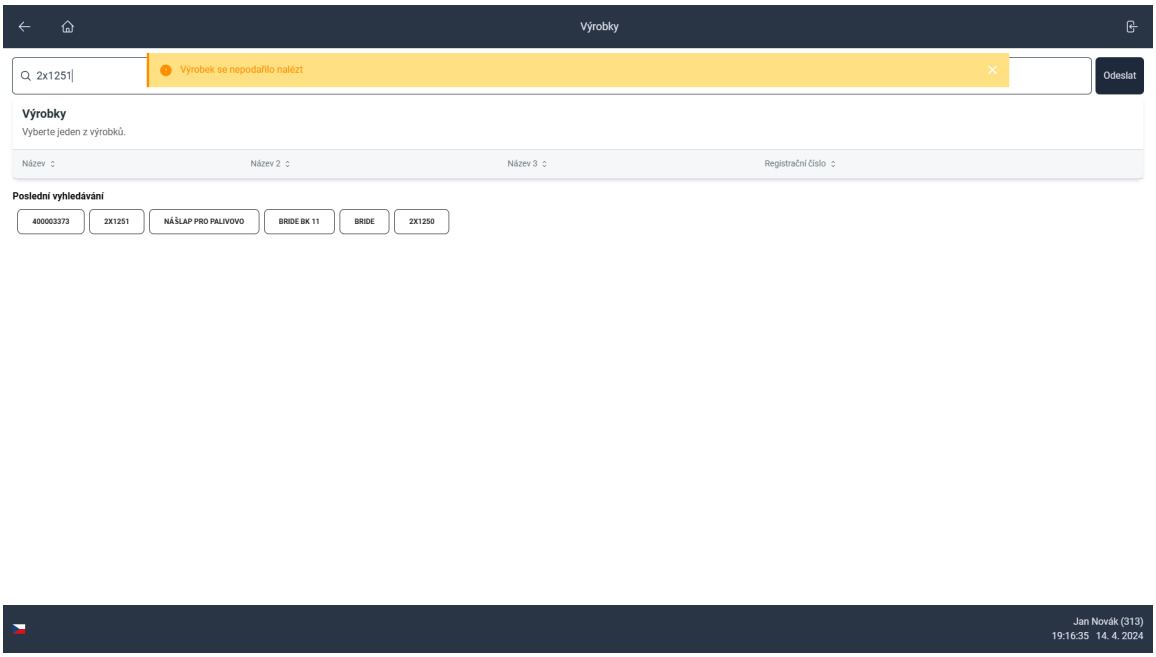
Obrázek C.11: Terminálová aplikace, potvrzovací formulář operace odvedení výroby



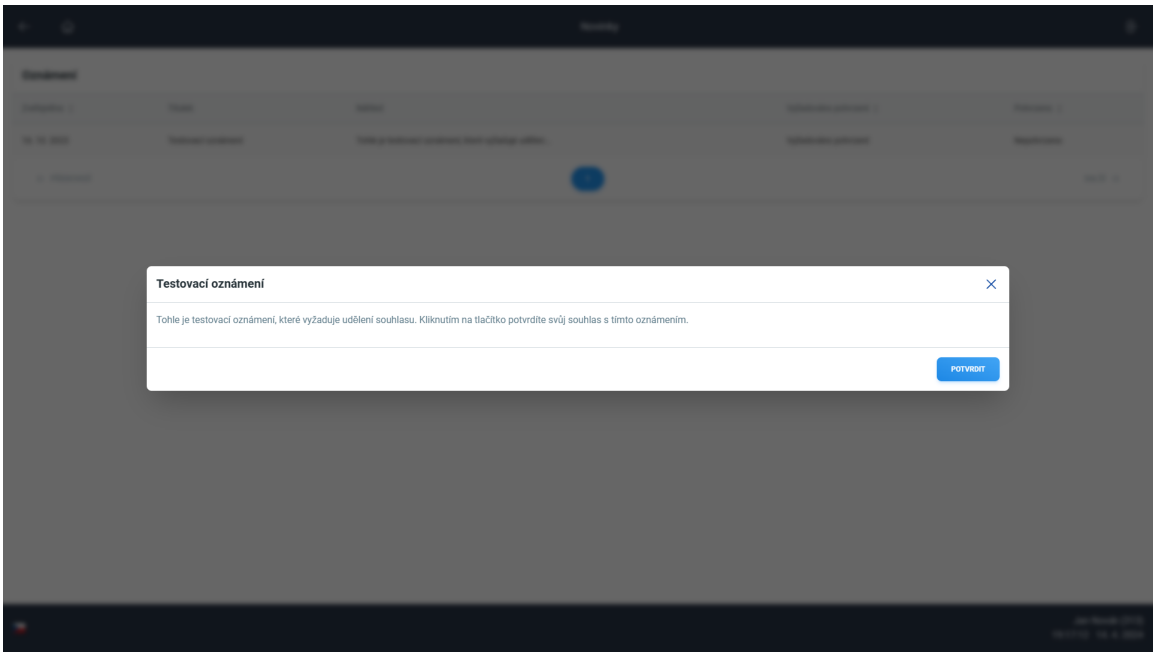
Obrázek C.12: Terminálová aplikace, vyhledávání materiálů



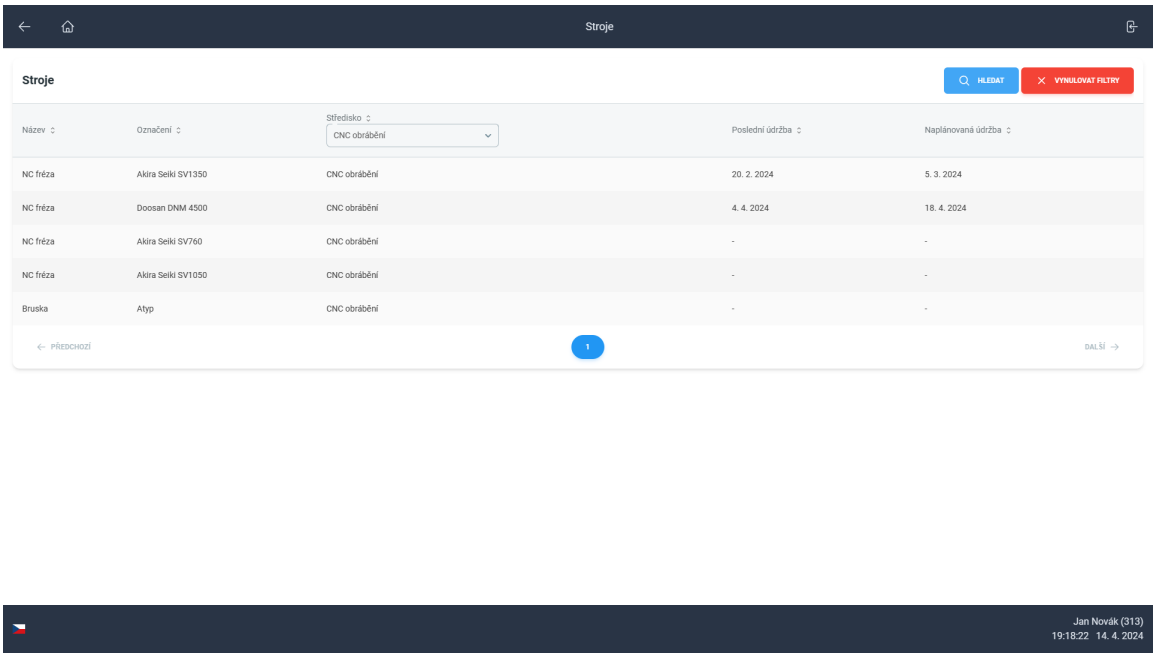
Obrázek C.13: Terminálová aplikace, naskladňování a vyskladňování materiálů



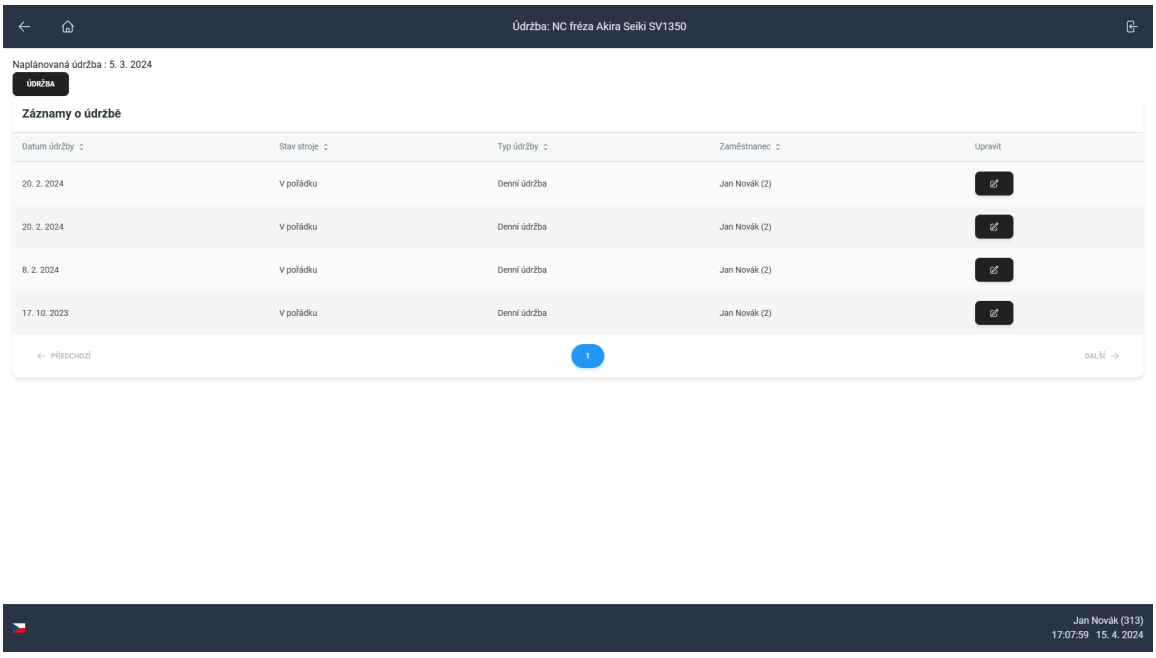
Obrázek C.14: Terminálová aplikace, vyhledávání výrobků



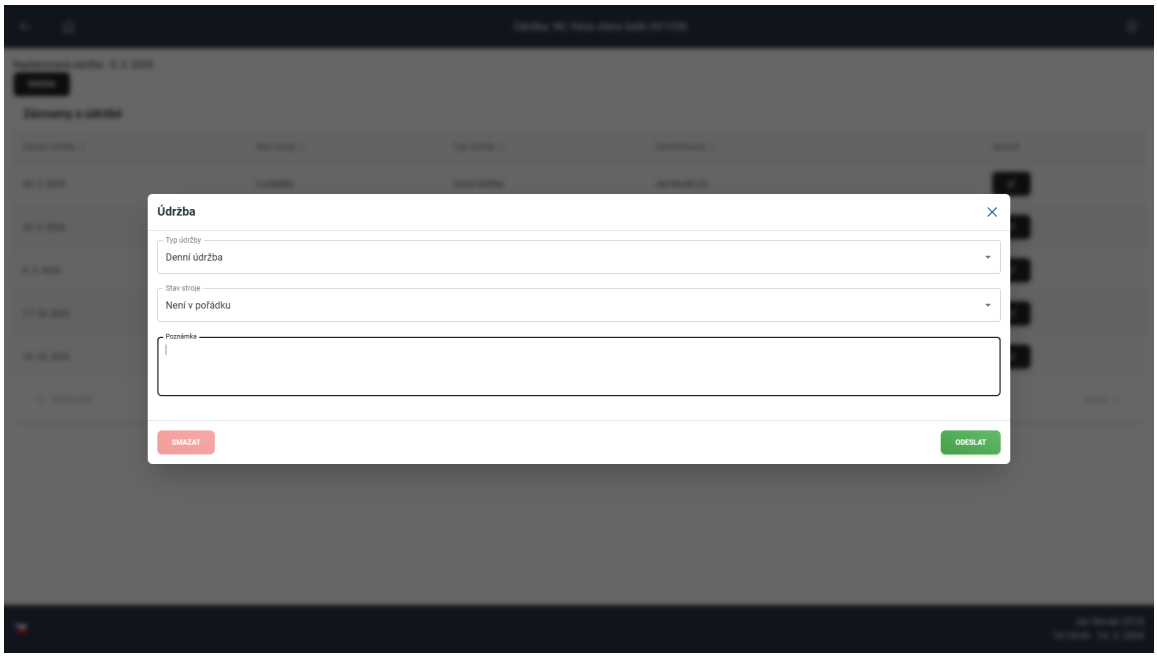
Obrázek C.15: Terminálová aplikace, novinky a oznámení



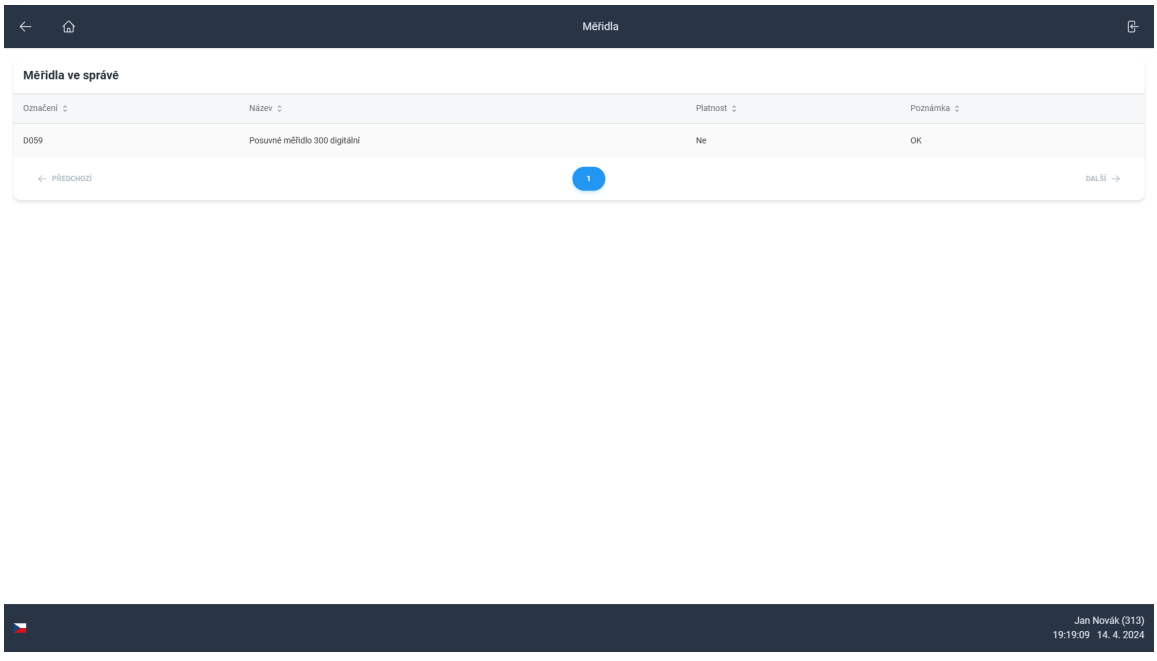
Obrázek C.16: Terminálová aplikace, přehled strojů ve firmě s filtrací



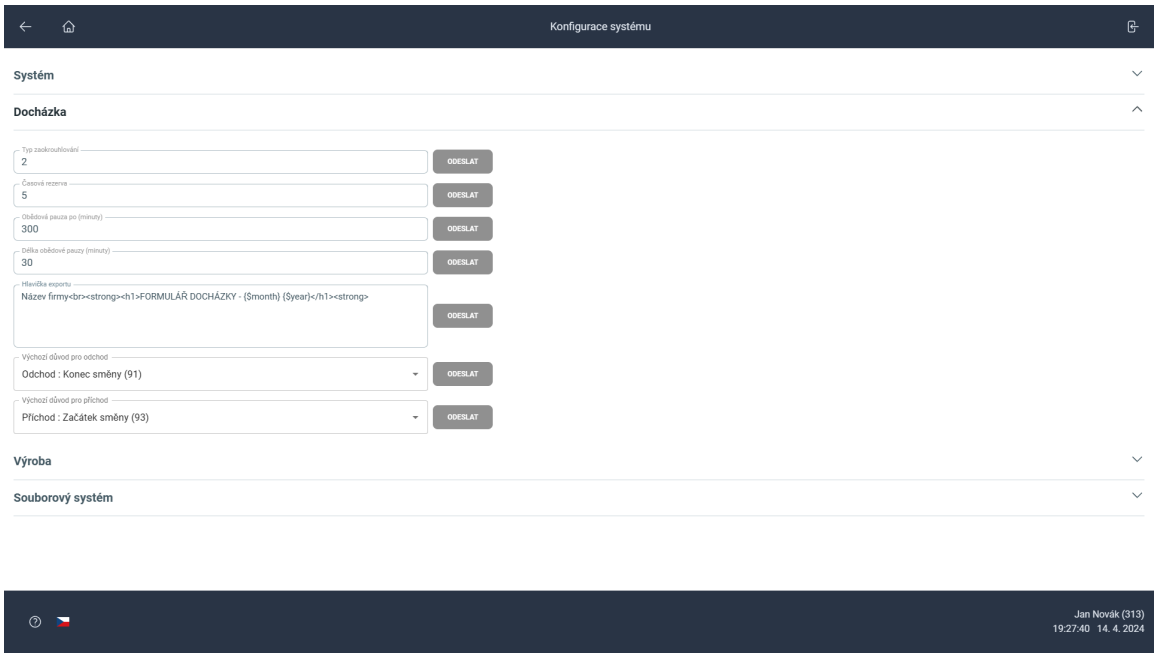
Obrázek C.17: Terminálová aplikace, přehled údržby stroje



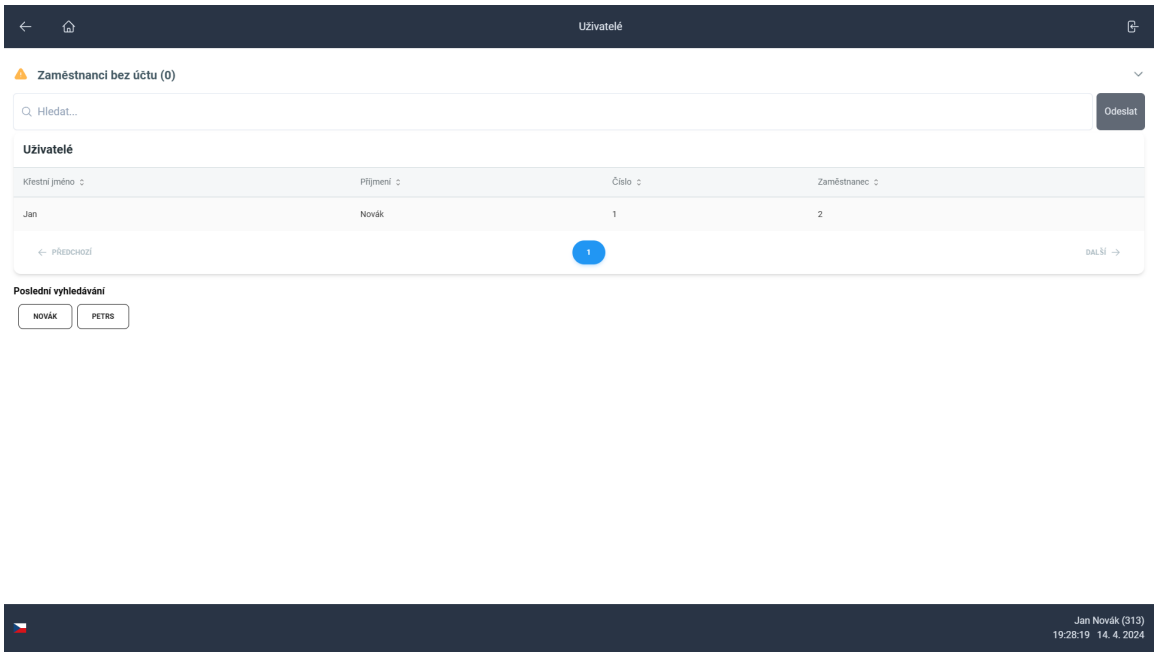
Obrázek C.18: Terminálová aplikace, zápis o údržbě stroje



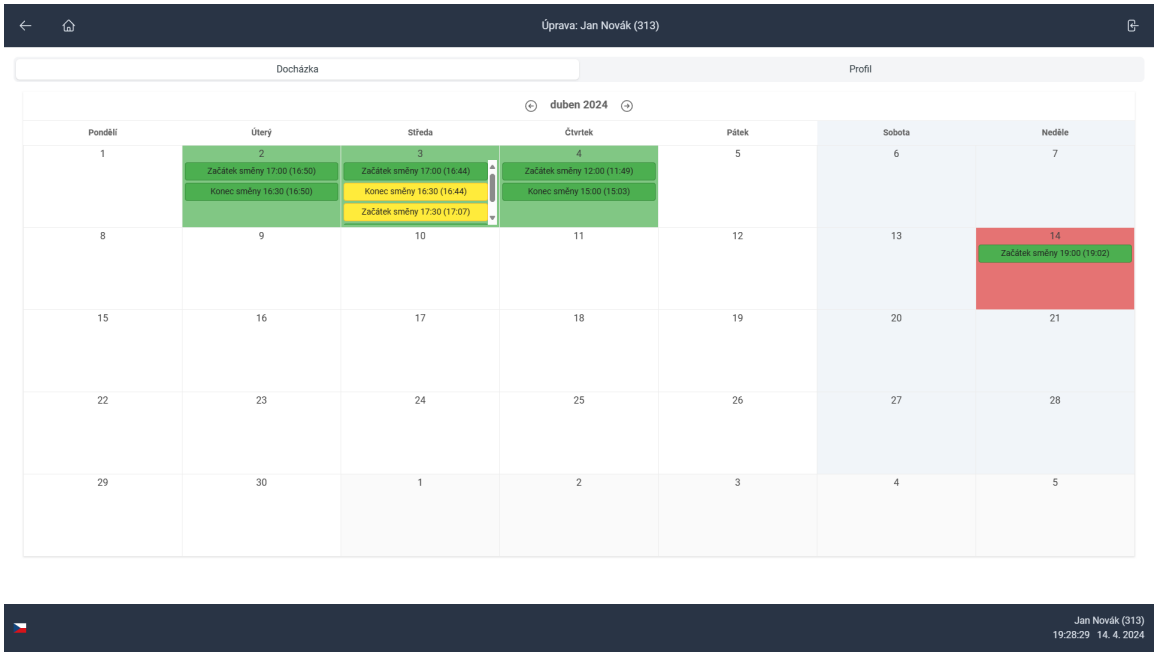
Obrázek C.19: Terminálová aplikace, přehled měřidel ve správě přihlášeného zaměstnance



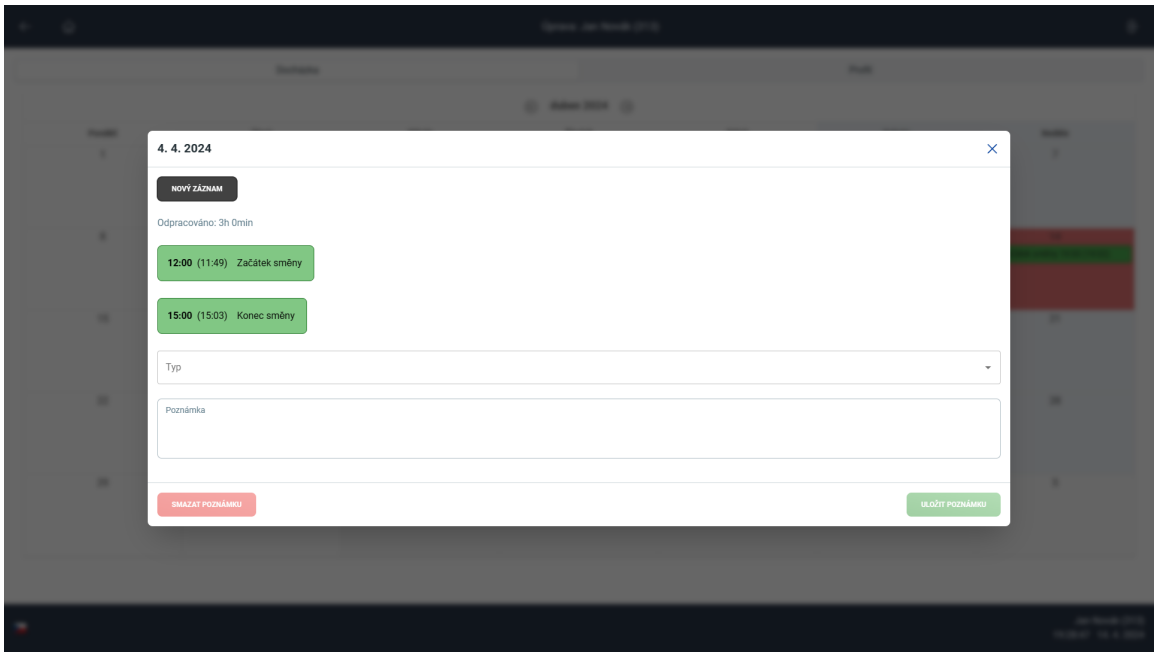
Obrázek C.20: Terminálová aplikace, administrátorská konfigurace systému



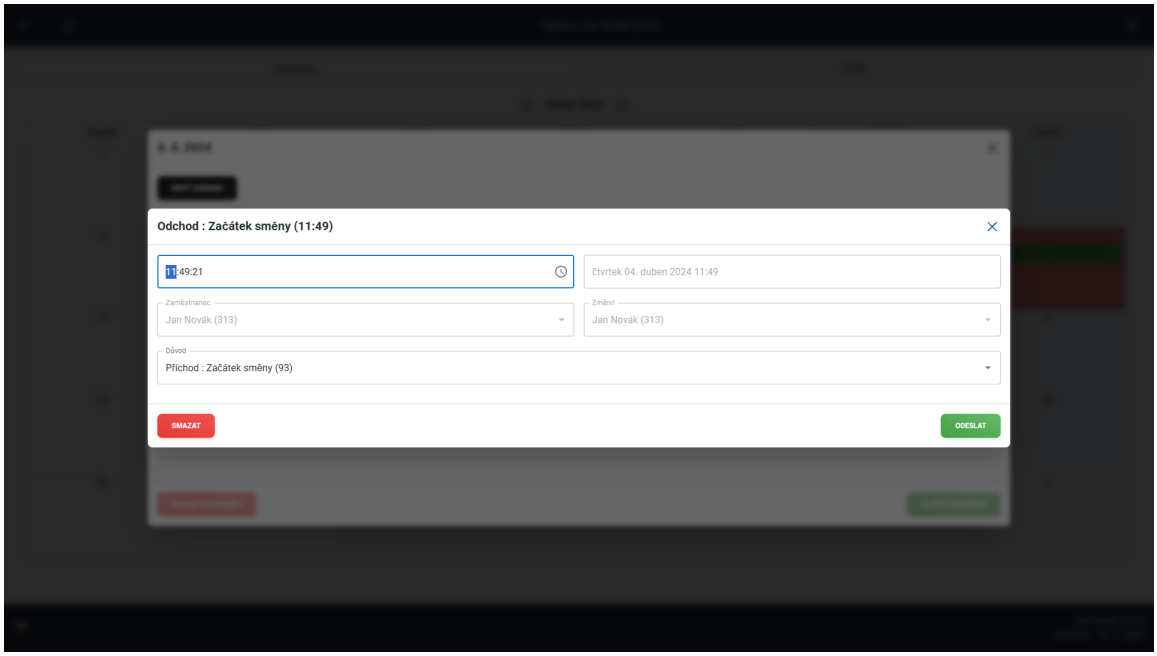
Obrázek C.21: Terminálová aplikace, administrátorská správa uživatelů, přehled



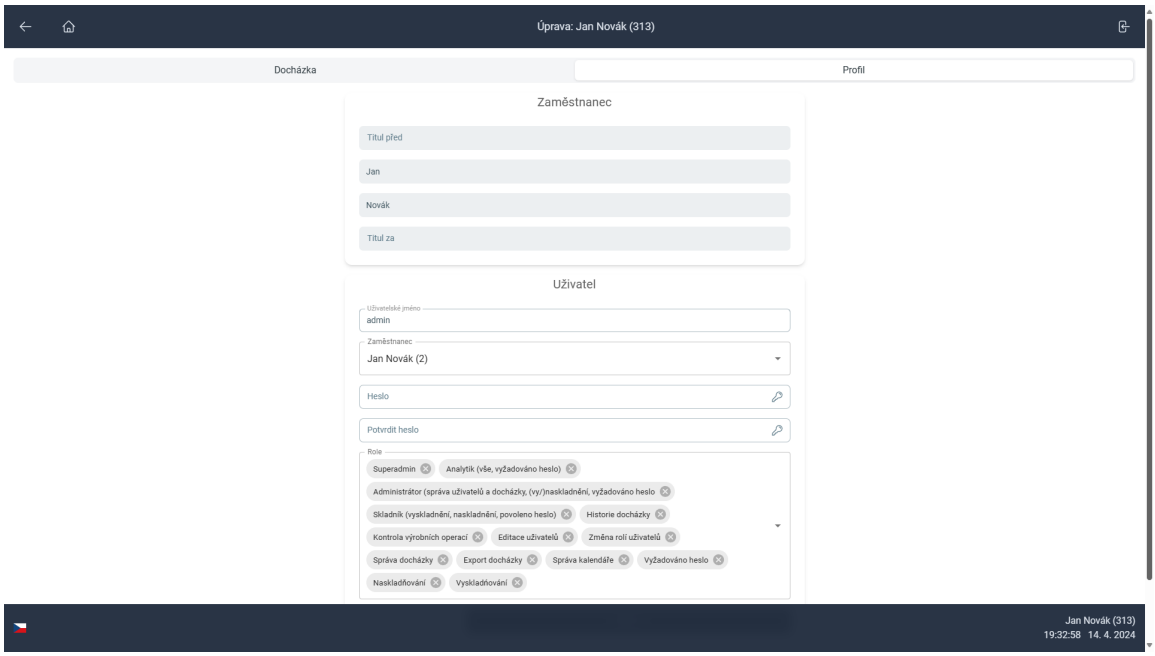
Obrázek C.22: Terminálová aplikace, administrátorská správa uživatelů, detail vybraného uživatele a jeho docházka



Obrázek C.23: Terminálová aplikace, administrátorská správa uživatelů, detail vybraného uživatele a přehled vybraného dne v jeho docházce



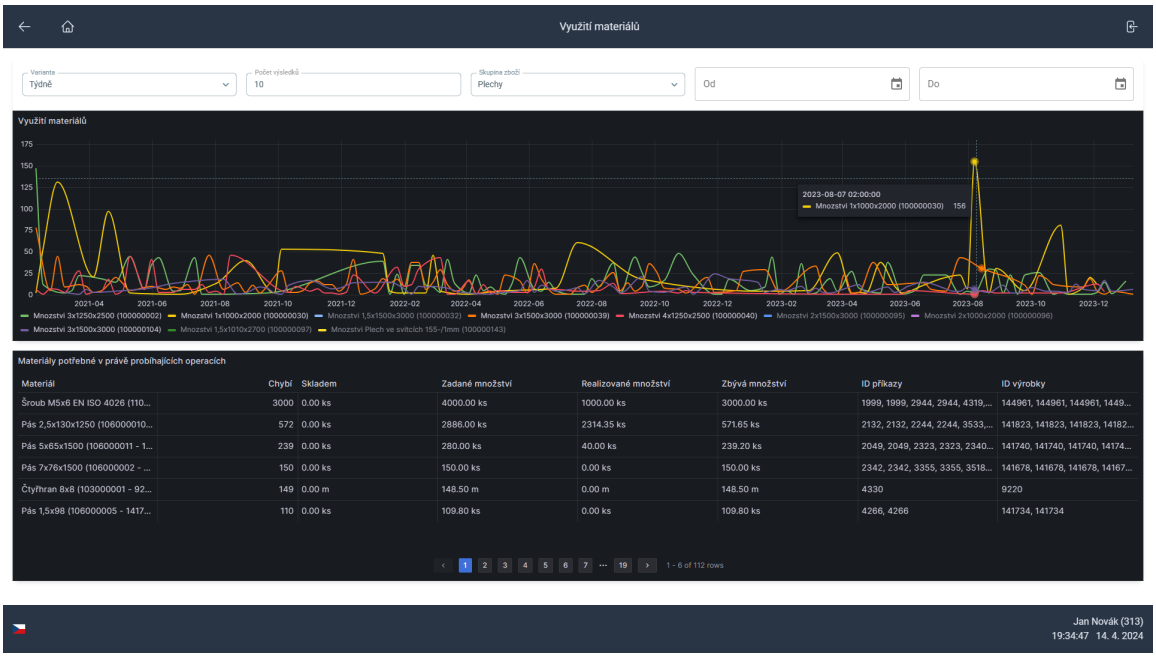
Obrázek C.24: Terminálová aplikace, administrátorská správa uživatelů, detail vybraného uživatele a úprava jednoho záznamu v jeho docházce



Obrázek C.25: Terminálová aplikace, administrátorská správa uživatelů, detail vybraného uživatele a jeho uživatelský profil

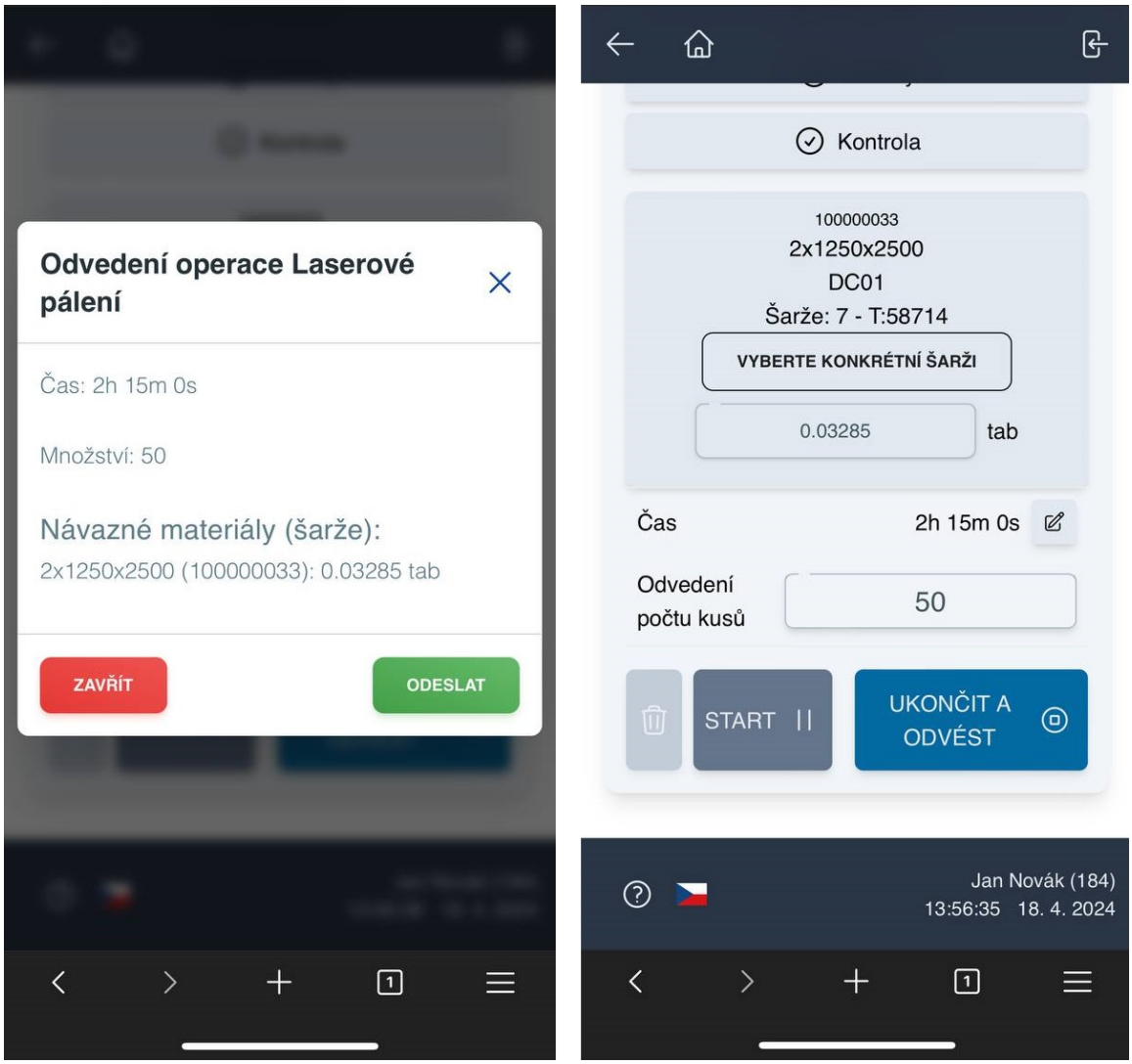
Obrázek C.26: Terminálová aplikace, modul analýzy dat, částečný přehled statistik výrobku

Obrázek C.27: Terminálová aplikace, modul analýzy dat, částečný přehled pohybů na skladě vybraného výrobku

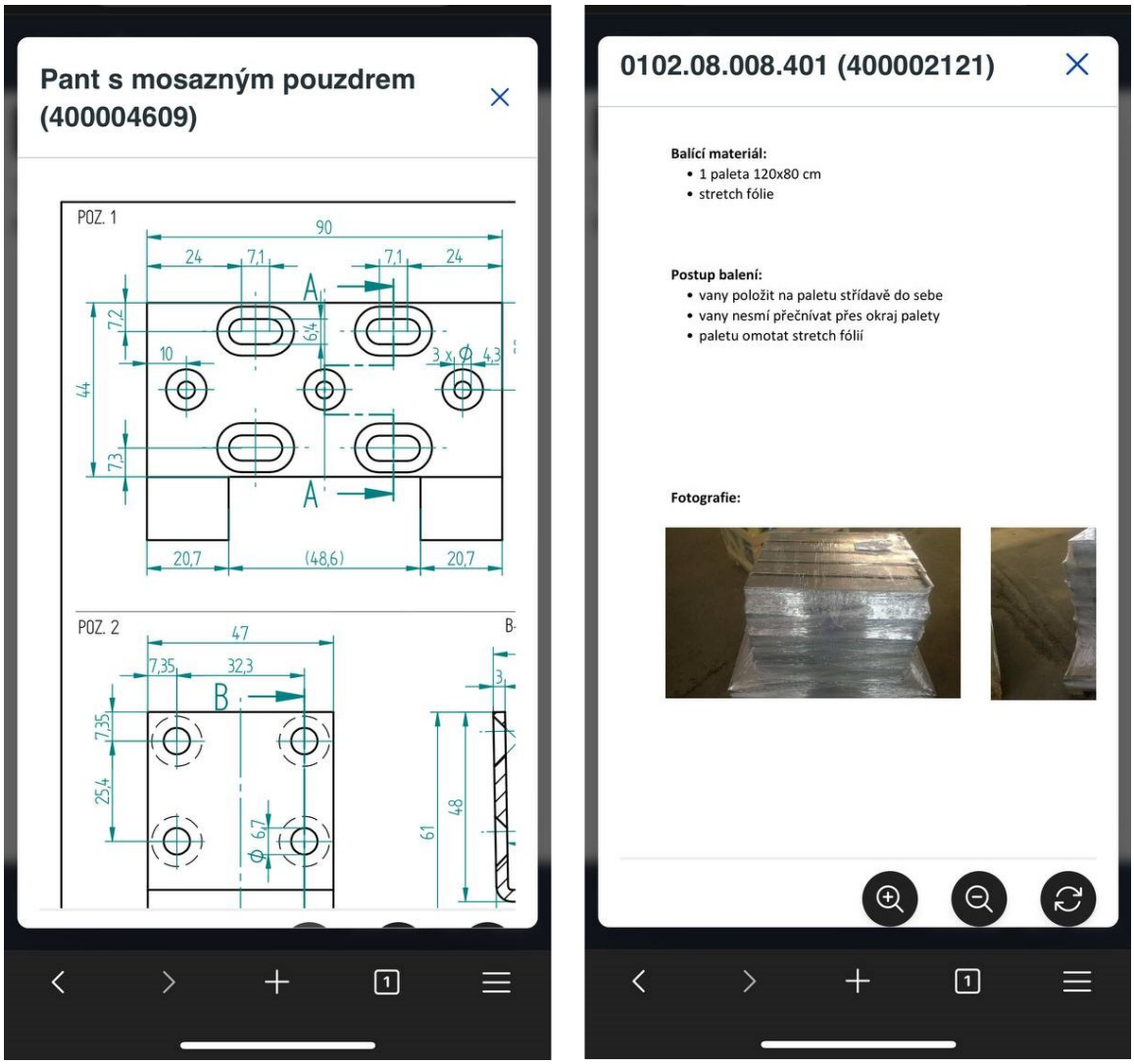


Obrázek C.28: Terminálová aplikace, modul analýzy dat, přehled statistik využití materiálů

Obrázek C.29: Terminálová aplikace, modul analýzy dat, částečný přehled statistik fakturací a příjmů



Obrázek C.30: Terminálová aplikace, zobrazení v telefonu, odvedení výrobní operace



Obrázek C.31: Terminálová aplikace, zobrazení v telefonu, zobrazení PDF výkresu (vlevo) a balicího předpisu (vpravo)

Příloha D

Zdrojové kódy analýzy dat

Tato příloha obsahuje zdrojové kódy pro analýzu dat. České názvy sloupců a tabulek vychází ze struktury systému HELIOS, neboť tyto pohledy jsou viditelné z administrace systému a anglické názvy by byly pro pracovníky matoucí. Sloupce s českým názvem jsou přejaty ze systému HELIOS, sloupce s anglickým názvem jsou čistě interní, sloužící pro další zpracování, například v Grafaně.

```
-- Definice tabulek nutných pro předzpracování dat pomocí Pythonu

-- Metadata o operacích
IF NOT EXISTS (SELECT * FROM sys.objects WHERE object_id =
OBJECT_ID(N'Tabx_Statistiky_Vyrobnich_Operaci') AND type in (N'U'))
BEGIN
    CREATE TABLE Tabx_Statistiky_Vyrobnich_Operaci (
        IDTabKmen INT,
        Doklad INT,
        Nazev NVARCHAR(100),
        Defectiveness FLOAT,
        TotalMade FLOAT,
        TotalDefective FLOAT,
        AvgTimePerPiece FLOAT,
        AvgCostPerPiece FLOAT,
        AvgCostPerPieceWithDefective FLOAT,
        MinTimePerPiece FLOAT,
        MaxTimePerPiece FLOAT,
        MinCostPerPiece FLOAT,
        MaxCostPerPiece FLOAT
    );

    CREATE INDEX IX_Tabx_Statistiky_Vyrobnich_Operaci_IDTabKmen ON
Tabx_Statistiky_Vyrobnich_Operaci (IDTabKmen);
    CREATE INDEX IX_Tabx_Statistiky_Vyrobnich_Operaci_IDTabKmen_Doklad ON
Tabx_Statistiky_Vyrobnich_Operaci (IDTabKmen, Doklad);
    CREATE UNIQUE INDEX IX_Tabx_Statistiky_Vyrobnich_Operaci_IDTabKmen_Doklad_Nazev
ON Tabx_Statistiky_Vyrobnich_Operaci (IDTabKmen, Doklad, Nazev);
END
```

```
-- Pomocné tabulky (pohledy) pro Grafanu
-- Slouží pro zjednodušení tvorby nástěnek. Grafana sama řeší stránkování a
další uživatelské prvky, je zbytečné přidávat další vrstvu navíc (Grafana
<- x <- databáze), ale zároveň používá proměnné, je tedy důležité, aby i
v případě například SQL injection se uživatel Grafany nedostal k žádným
dalším datům. Grafana uživatel má přístup pouze k těmto pohledům s čtecími
právy.

-- Analýza materiálů - celkový přehled operací
CREATE VIEW Tabx_Analyza_Materialy_Vyuziti_Operace AS
WITH MaterialySub AS (
    SELECT
        tp.ID AS IDPrikaz,
        tpp.Doklad AS CisloOperace,
        tpkv.nizsi AS material,
        tpkv.vyssi AS vyrobek,
        SUM(tpkv.mnoz_zad) AS material_operace_mnozstvi_zadane,
        SUM(tpkv.mnoz_zad * (tpp.Kusy_real/tpp.Kusy_zad)) AS
material_operace_mnozstvi_realizovane,
        SUM(tpkv.mnoz_zad * (tpp.KusyPredOperaci/tpp.Kusy_zad)) AS
material_operace_mnozstvi_zbyva
    FROM
        TabPrikaz tp
    LEFT JOIN
        TabDokladyZbozi tdz ON tdz.IDPrikaz = tp.ID
    LEFT JOIN
        TabPohybyZbozi tpz ON tpz.IDDoklad = tdz.ID
    INNER JOIN
        TabPrPostup tpp ON tpp.IDPrikaz = tp.ID
    INNER JOIN
        TabPrKVazby tpkv ON tpkv.IDPrikaz = tp.ID AND tpkv.Doklad =
tpp.Doklad
    WHERE
        StavPrikazu IN (10, 20, 30) -- Rozpracováno
    GROUP BY
        tp.ID, tpp.Doklad, tpkv.nizsi, tpkv.vyssi
)
SELECT
    CONCAT(
        COALESCE(NULLIF(tkzm.Nazev1, ''), tkzm.Nazev2),
        ' (', tkzm.RegCis, ' - ', tkzm.ID, ')'
    ) AS 'Materiál',
    CAST(
        SUM(ms.material_operace_mnozstvi_zbyva) - SUM(tss.Mnozstvi) AS
DECIMAL(18, 2)
    ) AS 'Chybí',
    CONCAT(
```

```
        CAST(SUM(tss.Mnozstvi) AS DECIMAL(18, 2)), ' ', tkzm.MJEvidence
    ) AS 'Skladem',
    CONCAT(
        CAST(SUM(ms.material_operace_mnozstvi_zadane) AS DECIMAL(18, 2)), '
', tkzm.MJEvidence
    ) AS 'Zadané množství',
    CONCAT(
        CAST(SUM(ms.material_operace_mnozstvi_realizovane) AS DECIMAL(18,
2)), ' ', tkzm.MJEvidence
    ) AS 'Realizované množství',
    CONCAT(
        CAST(SUM(ms.material_operace_mnozstvi_zbyva) AS DECIMAL(18, 2)), '
', tkzm.MJEvidence
    ) AS 'Zbývá množství',
    STRING_AGG(ms.IDPrikaz, ', ') AS 'ID příkazy',
    STRING_AGG(tkzm.ID, ', ') AS 'ID výrobky'
FROM
    MaterialySub ms
INNER JOIN
    TabKmenZbozi tkzv ON tkzv.ID = ms.vyrobek
INNER JOIN
    TabKmenZbozi tkzm ON tkzm.ID = ms.material
INNER JOIN
    TabStavSkladu tss ON tss.IDKmenZbozi = ms.material
WHERE
    tkzm.SkupZbo LIKE '1%'
GROUP BY
    ms.material, tkzm.ID, tkzm.MJEvidence, tkzm.Nazev1, tkzm.Nazev2,
tkzm.RegCis
-- SELECT * FROM Tabx_Analyza_Materialy_Vyuziti_Operace ORDER BY 'Chybí'
DESC;

-- Parametrizované využití materiálů
CREATE FUNCTION Tabx_Analyza_Materialy_Vyuziti(@GroupByType NVARCHAR(50),
@TopN INT, @dateTo DATETIME, @dateFrom DATETIME, @itemGroup NVARCHAR(50))
RETURNS TABLE
AS
RETURN
(
    SELECT
        CONCAT(tkz.Nazev1, ' (', tkz.RegCis, ')') AS ProductName,
        CASE
            WHEN @GroupByType = 'yearly' THEN DATEFROMPARTS(YEAR(tdz.DatRealizace),
1, 1)
            WHEN @GroupByType = 'monthly' THEN DATEFROMPARTS(YEAR(tdz.DatRealizace),
MONTH(tdz.DatRealizace), 1)
            WHEN @GroupByType = 'weekly' THEN DATEADD(WEEK, DATEDIFF(WEEK,
0, tdz.DatRealizace), 0)
```

```
            WHEN @GroupByType = 'daily' THEN CAST(tdz.DatRealizace AS DATE)
        END AS DateGroup,
        SUM(tpz.Mnozstvi) AS Mnozstvi
    FROM
        TabPohybyZbozi tpz
    INNER JOIN
        TabDokladyZbozi tdz ON tpz.IDDoklad = tdz.ID
    INNER JOIN
        TabStavSkladu tss ON tss.ID = tpz.IDZboSklad
    INNER JOIN
        TabKmenZbozi tkz ON tss.IDKmenZbozi = tkz.ID
    WHERE
        tkz.SkupZbo = @itemGroup AND
        (@dateFrom IS NULL OR tdz.DatRealizace >= @dateFrom) AND
        (@dateTo IS NULL OR tdz.DatRealizace <= @dateTo) AND
        tss.IDKmenZbozi IN (
            SELECT TOP (@TopN) tss_inner.IDKmenZbozi
            FROM TabStavSkladu tss_inner
            INNER JOIN TabPohybyZbozi tpz_inner ON tss_inner.ID =
tpz_inner.IDZboSklad
            INNER JOIN TabDokladyZbozi tdz_inner ON tpz_inner.IDDoklad =
tdz_inner.ID
            INNER JOIN TabKmenZbozi tkz ON tss_inner.IDKmenZbozi = tkz.ID
            WHERE tdz_inner.DruhPohybuZbo = 4
            AND tkz.SkupZbo = @itemGroup
            AND tdz_inner.DatRealizace IS NOT NULL
            AND tss_inner.IDSklad != 600
            GROUP BY tss_inner.IDKmenZbozi
            ORDER BY SUM(tpz_inner.Mnozstvi) DESC
        )
        AND tss.IDSklad != 600
        AND tdz.DruhPohybuZbo = 4
        AND tdz.DatRealizace IS NOT NULL
    GROUP BY
        tkz.ID,
        tkz.RegCis,
        tkz.Nazev1,
        CASE
            WHEN @GroupByType = 'yearly' THEN DATEFROMPARTS(YEAR(tdz.DatRealizace),
1, 1)
            WHEN @GroupByType = 'monthly' THEN DATEFROMPARTS(YEAR(tdz.DatRealizace),
MONTH(tdz.DatRealizace), 1)
            WHEN @GroupByType = 'weekly' THEN DATEADD(WEEK, DATEDIFF(WEEK,
0, tdz.DatRealizace), 0)
            WHEN @GroupByType = 'daily' THEN CAST(tdz.DatRealizace AS DATE)
        END
)
-- SELECT *
```

```
-- FROM Tabx_Analyza_Materialy_Vyuziti(
--     '[[variant]]',
--     [[maximumCount]],
--     CASE WHEN [[dateTo]] IS NOT NULL THEN CONVERT(DATETIME,
CAST('[[dateTo]]' AS VARCHAR(8))) END,
--     CASE WHEN [[dateFrom]] IS NOT NULL THEN CONVERT(DATETIME,
CAST('[[dateFrom]]' AS VARCHAR(8))) END,
--     '[[itemGroup]]'
-- );

-- Analýza materiálů - Vývoj ceny a nákupů
CREATE FUNCTION Tabx_Analyza_Materialy_Vyvoj_ceny (
    @storage INT
)
RETURNS TABLE
AS
RETURN
(
    SELECT
        CASE
            WHEN tpz.Mena = 'CZK' THEN tpz.JCbezDaniVal
            ELSE NULL
        END AS 'Cena [CZK]',
        CASE
            WHEN tpz.Mena != 'CZK' THEN tpz.JCbezDaniVal
            ELSE NULL
        END AS 'Cena [EUR]',
        tdz.DatPorizeni
    FROM TabPohybyZbozi tpz
    INNER JOIN TabDokladyZbozi tdz ON tdz.ID = tpz.IDDoklad
    WHERE tpz.IDZboSklad = @storage
    AND tdz.DruhPohybuZbo = 18
)
-- SELECT * FROM Tabx_Analyza_Materialy_Vyvoj_ceny([[storage]]);
CREATE FUNCTION Tabx_Analyza_Materialy_Vyvoj_nakupu (
    @storage INT
)
RETURNS TABLE
AS
RETURN
(
    SELECT tpz.Mnozstvi as Množství, tdz.DatPorizeni, tpz.MJ FROM
TabPohybyZbozi tpz
    INNER JOIN TabDokladyZbozi tdz ON tdz.ID = tpz.IDDoklad
    WHERE tpz.IDZboSklad = @storage
    AND tdz.DruhPohybuZbo = 18
)
-- SELECT * FROM Tabx_Analyza_Materialy_Vyvoj_nakupu([[storage]]);
```

```
-- Analýza materiálů - Vývoj stavu skladu
CREATE FUNCTION Tabx_Analyza_Materialy_Stav_skladu_prijem (
    @storage INT
)
RETURNS TABLE
AS
RETURN
(
    SELECT tpz.DatPorizeni, tpz.Mnozstvi AS Příjem
    FROM TabPohybyZbozi tpz
    INNER JOIN TabDokladyZbozi tdz ON tdz.ID = tpz.IDDoklad
    WHERE
    tpz.IDZboSklad = @storage AND
    tdz.DatRealizace IS NOT NULL AND
    tdz.DruhPohybuZbo = 0
)
-- SELECT * FROM Tabx_Analyza_Materialy_Stav_skladu_prijem([[storage]]);

CREATE FUNCTION Tabx_Analyza_Materialy_Stav_skladu_vydej (
    @storage INT
)
RETURNS TABLE
AS
RETURN
(
    SELECT tpz.DatPorizeni, tpz.Mnozstvi AS Výdej FROM TabPohybyZbozi tpz
INNER JOIN TabDokladyZbozi tdz ON tdz.ID = tpz.IDDoklad
    WHERE
    tpz.IDZboSklad = @storage AND
    tdz.DatRealizace IS NOT NULL AND
    tdz.DruhPohybuZbo = 4
)
-- SELECT * FROM Tabx_Analyza_Materialy_Stav_skladu_vydej([[storage]]);

CREATE FUNCTION Tabx_Analyza_Materialy_Stav_skladu_souhrn (
    @storage INT
)
RETURNS TABLE
AS
RETURN
(
    SELECT
    tpz.DatPorizeni,
    (
        (
            SELECT
                SUM(tpz_inner.Mnozstvi)
```

```
            FROM
                TabPohybyZbozi tpz_inner
            INNER JOIN
                TabDokladyZbozi tdz_inner ON tdz_inner.ID =
tpz_inner.IDDoklad
            WHERE
                tpz_inner.IDZboSklad = @storage
                AND tdz_inner.DatRealizace IS NOT NULL
                AND tdz_inner.DruhPohybuZbo = 0
                AND tpz_inner.DatPorizeni <= tpz.DatPorizeni
        ) - (
            SELECT
                SUM(tpz_inner.Mnozstvi)
            FROM
                TabPohybyZbozi tpz_inner
            INNER JOIN
                TabDokladyZbozi tdz_inner ON tdz_inner.ID =
tpz_inner.IDDoklad
            WHERE
                tpz_inner.IDZboSklad = @storage
                AND tdz_inner.DatRealizace IS NOT NULL
                AND tdz_inner.DruhPohybuZbo = 4
                AND tpz_inner.DatPorizeni <= tpz.DatPorizeni
        )
    ) AS 'Stav skladu'
FROM
    TabPohybyZbozi tpz
INNER JOIN
    TabDokladyZbozi tdz ON tdz.ID = tpz.IDDoklad
WHERE
    tpz.IDZboSklad = @storage
    AND tdz.DatRealizace IS NOT NULL
    AND tdz.DruhPohybuZbo IN (0, 4)
)
-- SELECT * FROM Tabx_Analyza_Materialy_Stav_skladu_prijem([[storage]]);

-- Analýza materiálů - Dodavatelské firmy
CREATE FUNCTION Tabx_Analyza_Materialy_Dodavatelske_firmy (
    @storage INT
)
RETURNS TABLE
AS
RETURN
(
    WITH Counts AS (
        SELECT
            COUNT(*) AS TotalCount
        FROM
```

```
            TabPohybyZbozi tpz
        INNER JOIN
            TabDokladyZbozi tdz ON tdz.ID = tpz.IDDoklad
        WHERE
            tpz.IDZboSklad = @storage
            AND tdz.DruhPohybuZbo = 18
    )
    SELECT
        co.Nazev AS Name,
        COUNT(*) * 100.0 / (SELECT TotalCount FROM Counts) AS Percentage
    FROM
        TabPohybyZbozi tpz
    INNER JOIN
        TabDokladyZbozi tdz ON tdz.ID = tpz.IDDoklad
    INNER JOIN
        TabCisOrg co ON co.CisloOrg = tdz.CisloOrg
    WHERE
        tpz.IDZboSklad = @storage
        AND tdz.DruhPohybuZbo = 18
    GROUP BY
        co.Nazev
);
-- SELECT * FROM Tabx_Analyza_Materialy_Dodavatelske_firmy([[storage]]);

-- Analýza produktů - Vývoj ceny a prodeje
CREATE FUNCTION Tabx_Analyza_Produkty_Vyvoj_ceny(@productionItem INT)
RETURNS TABLE
AS
RETURN
(
    SELECT
        CASE
            WHEN tpz.Mena = 'CZK' THEN tpz.JCbezDaniVal
            ELSE NULL
        END AS 'Cena [CZK]',
        CASE
            WHEN tpz.Mena != 'CZK' THEN tpz.JCbezDaniVal
            ELSE NULL
        END AS 'Cena [EUR]',
        tdz.DatPorizeni
    FROM TabPohybyZbozi tpz
    INNER JOIN TabDokladyZbozi tdz ON tdz.ID = tpz.IDDoklad
    WHERE tpz.IDZboSklad IN (
        SELECT ID
        FROM TabStavSkladu
        WHERE IDKmenZbozi = @productionItem
    )
    AND tdz.DruhPohybuZbo = 13
```

```
)
CREATE FUNCTION Tabx_Analyza_Produkty_Vyvoj_prodeje(@productionItem INT)
RETURNS TABLE
AS
RETURN
(
    SELECT tpz.Mnozstvi AS 'Množství', tdz.DatPorizeni
    FROM TabPohybyZbozi tpz
    INNER JOIN TabDokladyZbozi tdz ON tdz.ID = tpz.IDDoklad
    WHERE tpz.IDZboSklad IN (
        SELECT ID
        FROM TabStavSkladu
        WHERE IDKmenZbozi = @productionItem
    )
    AND tdz.DruhPohybuZbo = 13
)
-- SELECT * FROM Tabx_Analyza_Produkty_Vyvoj_ceny([[productionItem]]);
-- SELECT * FROM Tabx_Analyza_Produkty_Vyvoj_prodeje([[productionItem]]);

-- Analýza produktů - Statisiky výrobku
CREATE FUNCTION Tabx_Analyza_Produkty_Statistiky_Vyrobku (
    @productionItem INT
)
RETURNS TABLE
AS
RETURN (
    SELECT
        avgs.Doklad AS 'Pořadí operace',
        avgs.Nazev As 'Název',
        avgs.AvgTimePerPiece AS 'Čas [s]',
        avgs.AvgCostPerPiece AS 'Cena [CZK]',
        avgs.AvgCostPerPieceWithDefective AS 'Cena se zmetky [CZK]',
        avgs.Defectiveness * 100 AS 'Zmetkovitost [%]'
    FROM Tabx_Statistiky_Vyrobnich_Operaci avgs
    WHERE avgs.IDTabKmen = @productionItem
)
-- SELECT * FROM Tabx_Analyza_Produkty_Statistiky_Vyrobku([[productionItem]])
ORDER BY 'Pořadí operace'

-- Analýza produktů - Statistiky operací nad výrobkem
CREATE FUNCTION Tabx_Analyza_Produkty_Statistiky_Operace (
    @productionItem INT,
    @operation VARCHAR(300),
    @operationNumber INT
)
RETURNS TABLE
AS
RETURN (
```

```
    SELECT
        GETDATE() AS CurrentDate,
        za.CisloJmenoTituly,
        AVG(NULLIF(tpz.Sk_cas_S, 0) / NULLIF(tpz.kusy_odv, 0) ) AS
'Průměrný čas [s/ks]',
        CASE
            WHEN SUM(tpz.kusy_odv) <= 0 AND SUM(tpz.Operace_zmet_neopr) > 0
THEN 100
            ELSE (SUM(tpz.Operace_zmet_neopr) / NULLIF(SUM(NULLIF(tpz.kusy_odv,
0)), 0))*100
        END AS 'Zmetkovitost [%]',
        SUM(tpz.Operace_zmet_neopr) AS 'Zmetky celkem',
        SUM(tpz.kusy_odv) AS 'Vyrobeno celkem'
    FROM TabPrikazMzdyAZmetky tpz
    INNER JOIN TabPrPostup pp ON pp.IDPrikaz = tpz.IDPrikaz AND pp.Doklad =
tpz.DokladPrPostup
    INNER JOIN TabCisZam za ON za.ID = tpz.Zamestnanec
    WHERE IDTabKmen = @productionItem AND
        pp.nazev = @operation AND
        pp.Doklad = @operationNumber
    GROUP BY tpz.Zamestnanec, za.CisloJmenoTituly
)
-- SELECT *
-- FROM Tabx_Analyza_Produkty_Statistiky_Operace (
--     [[productionItem]],
--     '[[operation]]',
--     [[operationNumber]]
-- );

-- Analýza zaměstnanců - Odvedené operace
CREATE FUNCTION Tabx_Analyza_Zamestnanci_Operace (
    @employee INT,
    @year INT,
    @month INT,
    @dayFrom INT,
    @dayTo INT
)
RETURNS TABLE
AS
RETURN (
    SELECT
        CONCAT(DatumUkonceniOp_D, '. ', DatumUkonceniOp_M, '. ',
DatumUkonceniOp_Y) AS 'Datum',
        CONCAT(DatumUkonceniOp_D, '. ', DatumUkonceniOp_M, '. ',
DatumUkonceniOp_Y, ' - ', COALESCE(p.Nazev, 'Celkem')) AS 'Datum -
Operace',
        SUM(Sk_cas_S)/3600 AS 'Odvedeno [h]',
        COALESCE(p.Nazev, 'Celkem') AS 'Nazev'
```

```
    FROM
        TabPrikazMzdyAZmetky tpmz
    LEFT JOIN TabCPraco p ON p.ID = tpmz.IDPracoviste
    WHERE
        Zamestnanec = @employee AND
        DatumUkonceniOp_Y = @year AND
        DatumUkonceniOp_M = @month AND
        DatumUkonceniOp_D >= @dayFrom AND
        DatumUkonceniOp_D <= @dayTo
    GROUP BY
        GROUPING SETS ((DatumUkonceniOp_Y, DatumUkonceniOp_M,
DatumUkonceniOp_D, p.Nazev), (DatumUkonceniOp_Y, DatumUkonceniOp_M,
DatumUkonceniOp_D))
)
--  SELECT * FROM Tabx_Analyza_Zamestnanci_Operace([[employee]], [[year]],
[[month]], [[dayFrom]], [[dayTo]]) ORDER BY 'Datum'

-- Analýza zaměstnanců - Porovnání efektivity
CREATE FUNCTION Tabx_Analyza_Zamestnanci_Efektivita (
    @employee INT
)
RETURNS TABLE
AS
RETURN (
    SELECT
    RegCis AS 'Registrační čislo',
    Nazev1 AS 'Výrobek (Název 1)',
    Nazev2 AS 'Výrobek (Název 2)',
    Nazev3 AS 'Výrobek (Název 3)',
    Nazev AS 'Operace',
    AvgTimePerPiece AS 'Průměrný čas [s/ks]',
    EmployeeAvgTimePerPiece AS 'Zaměstnanec - průměrný čas [s/ks]',
    AvgCostPerPiece AS 'Průměrná cena [cena/MJ]',
    EmployeeAvgCostPerPiece AS 'Zaměstnanec - Průměrná cena [cena/MJ]',
    Defectiveness AS 'Průměrná zmetkovitost',
    EmployeeAvgDefectiveness AS 'Zaměstnanec - Průměrná zmetkovitost',
    TimeDifference AS 'Rozdíl - čas',
    CostDifference AS 'Rozdíl - cena',
    DefectDifference AS 'Rozdíl - zmetkovitost'
FROM
    (
        SELECT
            tkz.RegCis,
            tkz.Nazev1,
            tkz.Nazev2,
            tkz.Nazev3,
            avgs.IDTabKmen,
            avgs.Doklad,
```

```
            avgs.Nazev,
            avgs.AvgTimePerPiece AS AvgTimePerPiece,
            avgs.AvgCostPerPiece AS AvgCostPerPiece,
            avgs.Defectiveness AS Defectiveness,
            EmployeeAvgTimePerPiece,
            EmployeeAvgCostPerPiece,
            EmployeeAvgDefectiveness,
            ABS(avgs.AvgTimePerPiece - EmployeeAvgTimePerPiece) AS
TimeDifference,
            ABS(avgs.AvgCostPerPiece - EmployeeAvgCostPerPiece) AS
CostDifference,
            ABS(avgs.Defectiveness - EmployeeAvgDefectiveness) AS
DefectDifference,
            ABS(avgs.AvgTimePerPiece - EmployeeAvgTimePerPiece)
+ ABS(avgs.AvgCostPerPiece - EmployeeAvgCostPerPiece) +
ABS(avgs.Defectiveness - EmployeeAvgDefectiveness) AS SumOfDifferences
        FROM
            (
                SELECT
                    avgs.IDTabKmen,
                    avgs.Doklad,
                    avgs.Nazev,
                    AvgTimePerPiece,
                    AvgCostPerPiece,
                    Defectiveness
                FROM
                    Tabx_Statistiky_Vyrobnich_Operaci avgs
                INNER JOIN TabPrikazMzdyAZmetky tpmz ON avgs.IDTabKmen =
tpmz.IDTabKmen
            ) AS avgs
        INNER JOIN (
            SELECT
                tpmz.IDTabKmen,
                tpp.Doklad,
                tpp.nazev,
                AVG(CAST(tpmz.Sk_cas_S AS FLOAT) / NULLIF(tpmz.kusy_odv,
0)) AS EmployeeAvgTimePerPiece,
                AVG(CAST(tpmz.Mzda AS FLOAT) / NULLIF(tpmz.kusy_odv, 0)) AS
EmployeeAvgCostPerPiece,
                AVG(CAST(tpmz.Operace_zmet_neopr AS FLOAT)) AS
EmployeeAvgDefectiveness
            FROM
                TabPrikazMzdyAZmetky tpmz
            INNER JOIN TabPrikaz tp ON tp.ID = tpmz.IDPrikaz
            INNER JOIN TabPrPostup tpp ON tpp.IDPrikaz = tpmz.IDPrikaz AND
tpp.Doklad = tpmz.DokladPrPostup
            WHERE
                tpmz.Zamestnanec = @employee
```

```
            GROUP BY
                tpmz.IDTabKmen, tpp.Doklad, tpp.nazev
        ) AS EmployeeProducts ON avgs.IDTabKmen = EmployeeProducts.IDTabKmen
AND EmployeeProducts.Doklad = avgs.Doklad AND EmployeeProducts.nazev =
avgs.Nazev
        INNER JOIN TabKmenZbozi tkz ON tkz.ID = avgs.IDTabKmen
        GROUP BY
            tkz.RegCis, tkz.Nazev1, tkz.Nazev2, tkz.Nazev3, avgs.IDTabKmen,
avgs.Doklad, avgs.Nazev, avgs.AvgTimePerPiece, avgs.AvgCostPerPiece,
avgs.Defectiveness, EmployeeAvgTimePerPiece, EmployeeAvgCostPerPiece,
EmployeeAvgDefectiveness
    ) AS Results
)

-- Analýza fakturací
-- Vydané fakturace
CREATE VIEW Tabx_Analyza_Fakturace_Vydane AS
    SELECT
        YEAR(tdz.DUZP_X) AS Year,
        MONTH(tdz.DUZP_X) AS Month,
        SUM(CASE WHEN tdz.Mena = 'CZK' THEN tdz.SumaValBezDPH ELSE 0 END)
AS SumCZK,
        SUM(CASE WHEN tdz.Mena = 'EUR' THEN tdz.SumaValBezDPH ELSE 0 END)
AS SumEUR,
        SUM(CASE WHEN tdz.Mena = 'CZK' THEN 1 ELSE 0 END) AS CountCZK,
        SUM(CASE WHEN tdz.Mena = 'EUR' THEN 1 ELSE 0 END) AS CountEUR
    FROM TabDokladyZbozi tdz
    WHERE tdz.DruhPohybuZbo = 13
    GROUP BY YEAR(tdz.DUZP_X), MONTH(tdz.DUZP_X);
-- SELECT SumCZK AS 'Suma CZK - Vydané', DATEFROMPARTS(Year, Month, 1) AS
Date FROM Tabx_Analyza_Fakturace_Vydane ORDER BY Year ASC, Month ASC;
-- SELECT SumEUR AS 'Suma EUR - Vydané', DATEFROMPARTS(Year, Month, 1) AS
Date FROM Tabx_Analyza_Fakturace_Vydane ORDER BY Year ASC, Month ASC;

-- Splatnost faktur (příjmy)
CREATE VIEW Tabx_Analyza_Fakturace_Vydane_Splatnost AS
    SELECT
        CASE
            WHEN tdz.DnuProdleniNazapl <= 0 THEN tdz.DnuProdleniNazapl -
(tdz.DnuProdleniNazapl % 7)
            ELSE ((tdz.DnuProdleniNazapl - 1) / 7 + 1) * 7
        END AS DaysRange,
        SUM(CASE WHEN tdz.Mena = 'CZK' THEN tdz.SumaValBezDPH ELSE 0 END)
AS SumCZK,
        SUM(CASE WHEN tdz.Mena = 'EUR' THEN tdz.SumaValBezDPH ELSE 0 END)
AS SumEUR,
        SUM(CASE WHEN tdz.Mena = 'CZK' THEN 1 ELSE 0 END) AS CountCZK,
        SUM(CASE WHEN tdz.Mena = 'EUR' THEN 1 ELSE 0 END) AS CountEUR
```

```
    FROM TabDokladyZbozi tdz
    WHERE
        tdz.DruhPohybuZbo = 13 AND
        tdz.DnuProdleniNazapl IS NOT NULL AND
        tdz.DnuProdleniNazapl <= 180
    GROUP BY
        CASE
            WHEN tdz.DnuProdleniNazapl <= 0 THEN tdz.DnuProdleniNazapl -
(tdz.DnuProdleniNazapl % 7)
            ELSE ((tdz.DnuProdleniNazapl - 1) / 7 + 1) * 7
        END
-- SELECT DaysRange, SumCZK AS 'Suma CZK', SumEUR AS 'Suma EUR' FROM
Tabx_Analyza_Fakturace_Vydane_Splatnost ORDER BY DaysRange ASC;

-- Přijaté faktury
CREATE VIEW Tabx_Analyza_Fakturace_Prijate AS
    SELECT
        YEAR(tdz.DUZP_X) AS Year,
        MONTH(tdz.DUZP_X) AS Month,
        SUM(CASE WHEN tdz.Mena = 'CZK' THEN tdz.SumaValBezDPH ELSE 0 END)
AS SumCZK,
        SUM(CASE WHEN tdz.Mena = 'EUR' THEN tdz.SumaValBezDPH ELSE 0 END)
AS SumEUR,
        SUM(CASE WHEN tdz.Mena = 'CZK' THEN 1 ELSE 0 END) AS CountCZK,
        SUM(CASE WHEN tdz.Mena = 'EUR' THEN 1 ELSE 0 END) AS CountEUR
    FROM TabDokladyZbozi tdz
    WHERE tdz.DruhPohybuZbo = 18
    GROUP BY YEAR(tdz.DUZP_X), MONTH(tdz.DUZP_X);
-- SELECT SumCZK AS 'Suma CZK - Přijaté', DATEFROMPARTS(Year, Month, 1) AS
Date FROM Tabx_Analyza_Fakturace_Prijate ORDER BY Year ASC, Month ASC;
-- SELECT SumEUR AS 'Suma EUR - Přijaté', DATEFROMPARTS(Year, Month, 1) AS
Date FROM Tabx_Analyza_Fakturace_Prijate ORDER BY Year ASC, Month ASC;

-- Podíl firem za poslední rok na přijatých fakturách
CREATE VIEW Tabx_Analyza_Fakturace_Podil_Firem AS
    SELECT
        tco.Nazev AS Company,
        SUM(tdz.SumaKcBezDPHDruh) AS TotalAmount
    FROM
        TabDokladyZbozi tdz
    INNER JOIN
        TabCisOrg tco ON tco.CisloOrg = tdz.CisloOrg
    WHERE
        tdz.DruhPohybuZbo = 13
        AND tdz.DUZP_X >= DATEADD(DAY, -365, GETDATE())
    GROUP BY
        tco.CisloOrg,
        tco.Nazev;
```

```
-- SELECT Company, TotalAmount FROM Tabx_Analyza_Fakturace_Podil_Firem
ORDER BY TotalAmount DESC

-- Notifikace
CREATE VIEW Tabx_Analyza_Prodejni_Nakupni_Cena
AS
WITH ExchangeRate AS (
    SELECT TOP 1 Kurz AS ExchangeRate
    FROM TabKurzList
    WHERE Mena = 'EUR'
    ORDER BY Datum DESC
)
SELECT
    ta.IDTabKmen,
    SUM(ta.AvgCostPerPieceWithDefective) AS TotalPriceWithDefective,
    SUM(
        CASE
            WHEN nc.CenaKC > 0 THEN nc.CenaKC
            ELSE nc.CenaVal1 * er.ExchangeRate
        END
    ) AS TotalSellPrice
FROM
    Tabx_Statistiky_Vyrobnich_Operaci ta
INNER JOIN
    TabNC nc ON nc.IDKmenZbozi = ta.IDTabKmen
CROSS JOIN
    ExchangeRate er
WHERE
    nc.CenovaUroven = 1 -- Prodej
    AND (nc.CenaKC > 0 OR nc.CenaVal1 > 0) -- Alespoň nějaká cana
GROUP BY
    ta.IDTabKmen;
-- Porovnání nákladů (Panel)
SELECT
  nc.IDTabKmen AS 'ID kmenové karty',
  CONCAT(tkz.Nazev1, ' ', tkz.Nazev2, ' ', tkz.Nazev3) AS 'Název',
  nc.TotalSellPrice AS 'Prodejní cena [Kč]',
  nc.TotalPriceWithDefective AS 'Náklady [Kč]'
FROM Tabx_Analyza_Prodejni_Nakupni_Cena nc
  INNER JOIN TabKmenZbozi tkz ON tkz.ID = nc.IDTabKmen
WHERE nc.TotalPriceWithDefective * 0.9 > nc.TotalSellPrice;
-- Alert
SELECT COUNT(*) FROM Tabx_Analyza_Prodejni_Nakupni_Cena nc
WHERE nc.TotalPriceWithDefective*0.9 >  nc.TotalSellPrice

-- Uživatel
CREATE LOGIN [GrafanaUser] WITH PASSWORD = 'ChangeMe';
CREATE USER [GrafanaUser] FOR LOGIN [GrafanaUser];
```

```
GRANT SELECT ON [Tabx_Analyza_Produkty_Vyvoj_prodeje] TO [GrafanaUser];
GRANT SELECT ON [Tabx_Analyza_Produkty_Vyvoj_ceny] TO [GrafanaUser];
GRANT SELECT ON [Tabx_Analyza_Materialy_Vyuziti] TO [GrafanaUser];
GRANT SELECT ON [Tabx_Statistiky_Vyrobnich_Operaci] TO [GrafanaUser];
GRANT SELECT ON [Tabx_Analyza_Materialy_Vyuziti_Operace] TO [GrafanaUser];
GRANT SELECT ON [Tabx_Analyza_Produkty_Statistiky_Vyrobku] TO
[GrafanaUser];
GRANT SELECT ON [Tabx_Analyza_Produkty_Statistiky_Operace] TO
[GrafanaUser];
GRANT SELECT ON [Tabx_Analyza_Materialy_Dodavatelske_firmy] TO
[GrafanaUser];
GRANT SELECT ON [Tabx_Analyza_Materialy_Vyvoj_ceny] TO [GrafanaUser];
GRANT SELECT ON [Tabx_Analyza_Materialy_Vyvoj_nakupu] TO [GrafanaUser];
GRANT SELECT ON [Tabx_Analyza_Materialy_Stav_skladu_prijem] TO
[GrafanaUser];
GRANT SELECT ON [Tabx_Analyza_Materialy_Stav_skladu_vydej] TO
[GrafanaUser];
GRANT SELECT ON [Tabx_Analyza_Materialy_Stav_skladu_souhrn] TO
[GrafanaUser];
GRANT SELECT ON [Tabx_Analyza_Zamestnanci_Operace] TO [GrafanaUser];
GRANT SELECT ON [Tabx_Analyza_Zamestnanci_Efektivita] TO [GrafanaUser];
GRANT SELECT ON [Tabx_Analyza_Fakturace_Vydane] TO [GrafanaUser];
GRANT SELECT ON [Tabx_Analyza_Fakturace_Prijate] TO [GrafanaUser];
GRANT SELECT ON [Tabx_Analyza_Fakturace_Vydane_Splatnost] TO [GrafanaUser];
GRANT SELECT ON [Tabx_Analyza_Fakturace_Podil_Firem] TO [GrafanaUser];
GRANT SELECT ON [Tabx_Analyza_Prodejni_Nakupni_Cena] TO [GrafanaUser];
```

Příloha E

Akceptační testy

Přihlášení pomocí čipu

- **Popis**: Přihlášení zaměstnance pomocí čipu z terminálu
- **Předpoklady**: Zaměstnanec není přihlášen a pokouší se k systému přistupovat z terminálu
- **Postup**:
  1. Zaměstnanec přiloží čip ke čtece kódu, anebo zadá jeho číslo ručně a odešle formulář
     Možné reakce:
     - Zobrazí se aplikace s výběrem modulů - Úspěch
     - Aplikace vyžaduje heslo - Chyba
     - Přihlášení selže s chybovou hláškou - Chyba

Přihlášení uživatele s rolí, která vyžaduje heslo

- **Popis**: Uživatel s rolí "Heslo vyžadováno"se pokouší přihlásit z libovolného zařízení
- **Předpoklady**: Uživatel není přihlášen
- **Postup**:
  1. Uživatel zadá své uživatelské jméno anebo číslo čipu a odešle formulář
     Možné reakce:
     - Aplikace vyžaduje heslo - Úspěch
     - Zobrazí se aplikace s výběrem modulů - Chyba
     - Přihlášení selže s chybovou hláškou - Chyba

Přihlášení mimo terminál

- **Popis**: Přihlášení zaměstnance mimo terminál
- **Předpoklady**: Zaměstnanec není přihlášen a pokouší se přihlásit z jiného zařízení, než je terminál, a má roli "Pouze terminál"
  1. Uživatel zadá své uživatelské jméno anebo číslo čipu a odešle formulář
     Možné reakce:

- Přihlášení selže s chybovou hláškou - Úspěch
- Aplikace vyžaduje heslo - Chyba
- Zobrazí se aplikace s výběrem modulů - Chyba

Vyhledávání pomocí čtečky kódů

- **Popis**: Rychlé vyhledávání výrobků nebo příkazů pomocí čtečky čárových kódů
- **Předpoklady**: Zaměstnanec je přihlášen a je na obrazovce umožňující vyhledávání
- **Postup**:
  1. Zaměstnanec naskenuje čárový kód výrobní operace nebo výrobku
     Možné reakce:
     - Výrobek nebo příkaz byl nalezen a uživatel je přesměrován na požadovaný detail - Úspěch
     - Vyhledávání se nespustí - Chyba
     - Příkaz nebo výrobek není nalezen - Chyba

Pozastavení výroby po odchodu zaměstnance

- **Popis**: Pozastavení automatického trasování času stráveného plněním výrobního příkazu
- **Předpoklady**: Zaměstnanec je přihlášen
- **Postup**:
  1. Zaměstnanec spustí automatické trasování času výroby
  2. Zaměstnanec libovolně pokračuje v práci, kromě operací ukončujících výrobu tohoto příkazu
  3. Zaměstnanec v modulu docházky přidá nový záznam typu odchod
- **Úspěch**: Výroba operace, nad kterou bylo spuštěno automatické trasování času, je pozastavena

Naskladnění materiálu

- **Popis**: Operace naskladnění materiálu včetně výběru konkrétní šarže
- **Předpoklady**: Zaměstnanec je přihlášen a má roli dovolující naskladňovat materiál
- **Postup**:
  1. Zaměstnanec vyhledá požadovaný materiál v modulu "Sklady a materiály"
  2. Zaměstnanec přejde na detail materiálu
     - Pokud se materiál vyskytuje ve více šaržích:
       * Systém vyžaduje výběr konkrétní šarže - Úspěch
       * Systém povolí další operace nad materiálem - Chyba
     - Pokud se materiál nevyskytuje ve více šaržích:
       * Systém povolí další operace nad materiálem - Úspěch

        * Systém vyžaduje výběr konkrétní šarže, nebo jinak zamezí pokračování - Chyba
    3. Ve formuláři "Naskladnění"zaměstnanec vyplní požadovaný počet jednotek materiálu a formulář odešle

- **Úspěch**: Operace je úspěšně dokončena a skladová zásoba materiálu se zvedne o požadovaný počet jednotek. V systému HELIOS vznikne příjemka s vybranou položkou (včetně šarže), která je realizována.

Vyskladnění materiálu

- **Popis**: Operace vyskladnění materiálu včetně výběru konkrétní šarže
- **Předpoklady**: Zaměstnanec je přihlášen a má roli dovolující vyskladňovat materiál
- **Postup**:
    1. Zaměstnanec vyhledá požadovaný materiál v modulu "Sklady a materiály"
    2. Zaměstnanec přejde na detail materiálu
        - Pokud se materiál vyskytuje ve více šaržích:
            * Systém vyžaduje výběr konkrétní šarže - Úspěch
            * Systém povolí další operace nad materiálem - Chyba
        - Pokud se materiál nevyskytuje ve více šaržích:
            * Systém povolí další operace nad materiálem - Úspěch
            * Systém vyžaduje výběr konkrétní šarže, nebo jinak zamezí pokračování - Chyba
    3. Ve formuláři "Vyskladnění"zaměstnanec vyplní požadovaný počet jednotek materiálu a formulář odešle

- **Úspěch**: Pokud je na skladě dostatek materiálu, operace je úspěšně dokončena a skladová zásoba materiálu se sníží o požadovaný počet jednotek. V systému HELIOS vznikne výdejka s vybranou položkou (včetně šarže), která je realizována. Pokud na skladě není dostatek materiálu, operace skončí chybou s chybovou hláškou vysvětlující důvod selhání operace.

Odvedení operace

- **Popis**: Odvedení jedné z operací příkazu k výrobě
- **Předpoklady**: Zaměstnanec je přihlášen a má roli dovolující odvádět výrobu. Výrobní příkaz je v rozpracovaném stavu (výroba povolena).
- **Postup**:
    1. Zaměstnanec vyhledá konkrétní příkaz a přejde na jeho detail
    2. Zaměstnanec vybere operaci, zadá čas strávený nad operací a počet vyrobených kusů
        - Pokud má výrobek navázaný materiál:
            * Systém automaticky vypočítá a zobrazí předpokládanou spotřebu materiálu, kterou je možné editovat - Úspěch

        * Spotřeba materiálu je vypočítána špatně nebo vůbec - Chyba
    3. Zaměstnanec odešle formulář

- **Úspěch**: Operace je úspěšně odvedena a v systému HELIOS vznikne zápis o výrobě, který je možné vidět na otevřeném příkaze v záložce "Historie". Je zvýšen počet realizovaných kusů konkrétní operace a snížen počet kusů nutných k výrobě. Pokud se jedná o operaci, která není první, a v předchozích operacích nebyl dostatečný počet vyrobených kusů, systém automaticky tyto kusy zpětně s nulovým časem výroby odvede pomocí fiktivního terminálového zaměstnance (jiného než přihlášený uživatel). Pokud byl na této operaci navázán materiál, vznikne v systému HELIOS výdejka do výroby.

Realizace operace

- **Popis**: Realizace finální operace výrobního příkazu
- **Předpoklady**: Zaměstnanec je přihlášen a má roli dovolující odvádět výrobu. Výrobní příkaz je v rozpracovaném stavu (výroba povolena).
- **Postup**:
    1. Zaměstnanec vyhledá konkrétní příkaz a přejde na jeho detail
    2. Zaměstnanec vybere poslední operaci, zadá čas strávený nad operací a počet vyrobených kusů
    3. Zaměstnanec odešle formulář
- **Úspěch**: V systému HELIOS vznikne jak zápis o výrobě, tak nerealizovaná příjemka výrobku na sklad. Počet kusů výrobku na skladě není zvýšen.

Odvedení zmetků

- **Popis**: Odvedení zmetkovitých kusů na konkrétní operaci výrobního příkazu
- **Předpoklady**: Zaměstnanec je přihlášen a má roli dovolující odvádět výrobu. Výrobní příkaz je v rozpracovaném stavu (výroba povolena).
- **Postup**:
    1. Zaměstnanec vyhledá konkrétní příkaz a přejde na jeho detail
    2. Zaměstnanec vybere libovolnou operaci a vybere možnost "Zmetky"
    3. Zaměstnanec zadá počet zmetkovitých kusů a odešle formulář
- **Úspěch**: V systému HELIOS vznikne zápis o výrobě zmetků. Pokud v předchozích operacích nebyl vyroben dostatečný počet kusů (zmetky by se generovaly z ničeho), systém automaticky zpětně vyrobí potřebný počet kusů nutných pro odvedení zmetků. Tyto kusy jsou vyrobeny s nulovým časem výroby pomocí fiktivního terminálového zaměstnance.