FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Modernizace informačního systému společnosti Workswell s.r.o.
Student:	Jiří Věneček
Vedoucí:	Ing. Jan Sova
Studijní program:	Informatika
Studijní obor:	Informační systémy a management
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce zimního semestru 2021/22

Pokyny pro vypracování

Cílem práce je analýza a modifikace informačního systému, který je aktuálně spol. Workswell s.r.o. využíván.
Popis:
1) Prozkoumejte stávající informační systém společnosti.
2) Po konzultaci s vedoucím a zaměstnanci společnosti proveďte sběr uživatelských požadavků, zmapujte firemní procesy a navrhněte vhodná rozšíření systému.
3) Proveďte analýzu funkčních a nefunkčních požadavků a navrhněte vhodnou funkcionalitu informačního systému.
4) Implementujte vámi navržené změny v souladu s metodami softwarového inženýrství.
5) Zhodnoťte ekonomicko-manažerské dopady a přínosy navrženého řešení pro společnost Workswell s.r.o.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 23. února 2020

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalářská práce

Modernizace informačního systému společnosti Workswell s.r.o.

Jiří Věneček

Katedra softwarového inženýrství
Vedoucí práce: Ing. Jan Sova

4. června 2020

Poděkování

Rád bych poděkoval svému vedoucímu práce Ing. Janu Sovovi a zaměstnancům společnosti Workswell Ing. Janu Kovářovi a Ing. Jánovi Markovi za pomoc při analýze systému společnosti.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů, zejména skutečnost, že České vysoké učení technické v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 citovaného zákona.

V Praze dne 4. června 2020 . . . . . .

Odkaz na tuto práci

Věneček, Jiří. *Modernizace informačního systému společnosti Workswell s.r.o.*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2020.

Abstrakt

Tato bakalářská práce se zabývá interním systémem společnosti Workswell, možnostmi jeho modernizace a rozšíření. Cílem práce bylo provést anazýzu stávají podoby systému, identifikovat jeho nedostatky a následně navrhnout a implementovat změny. Výsledkem analýzy bylo rozhodnutí vytvořit nový ERP systém místo dílčích úprav systému stávajícího. Všechny části firemních procesů prošly analýzou, na základě které byly identifikovány potřeby a požadavky společnosti. Klíčovými procesy jsou objednávání a naskladňování zboží, výrobní postupy a rezervace firemních zařízení. Pro vývoj nového systému byl použit framework webových aplikací Django. V závěru práce jsou navrženy další možnosti a směry rozvoje vytvořeného systému.

Klíčová slova Workswell, interní systém, ERP systém, Django framework, implementace webové aplikace, analýza a sběr požadavků

Abstract

This bachelor's thesis is about Workswell's internal system and its possible improvements and modifications. Main goal of this thesis was to analyse the system, identify its flaws, propose and implement agreed changes. As a result of the analysis a new ERP system was implemented instead of working on

improving the current system. All business processes were analyzed and their improvements were suggested. Key processes include ordering and stocking components, production and internal reservations. Django framework was used for developing the new system. At the end of this thesis the future of the new system is discussed.

Keywords Workswell, internal system, ERP system, Django framework, web application implementation, analysis and requirements engineering

Obsah

Seznam obrázků

Úvod

Informační systém je nepostradatelnou součástí většiny společností. Umožňuje vytvářet, ukládat, spravovat a zpracovávat data. Nároky na něj rostou spolu s růstem společnosti. Po nějaké době dojde k potřebě ukládat více dat, data třídit, kategorizovat, vyhledávat v nich, případně přidávat nové funkcionality systému. Workswell je česká firma vyrábějící termokamery a termovizní systémy, která byla založena v roce 2010. [1] Využívá interní systém, který se zabývá nejen logistikou a výrobním procesem produktů, ale umožňuje také zaznamenávat různé interní procesy firmy jako jsou například dovolené zaměstnanců, rezervace firemních automobilů apod. Systém je průběžně vyvíjen od roku 2012, ale nebyl nikdy modernizován, procházel jen drobnými úpravami. Historicky sloužil k mnoha dalším účelům, ke kterým nyní společnost používá externí nástroje a systémy. V současnosti obsahuje řadu bezpečnostních rizik, zastaralých technologií a minimální možnost rozšiřitelnosti. Práce popisuje přechod na nový interní systém společnosti. Řeší důvody, proč systém modernizovat, způsoby, jakými lze tuto modernizaci provést a další problematiku spojenou s tímto přechodem. Zabývá se otázkou, které části původního systému zachovat ve stávající podobě a které přepracovat. Zároveň práce řeší požadavek, aby přechod na nový systém nevyžadoval náročné zaškolení zaměstnanců. Implementace nového systému především eliminuje bezpečnostní rizika a urychlí firemní procesy hlavně v oblasti výroby a logistiky.

Praktická část bakalářské práce je rozdělena do několika částí. První popisuje analýzu stávajícího systému. Ta má ukázat a vysvětlit vedení společnosti, proč je výhodné přistoupit k vývoji modernizovaného systému místo provádění úprav systému stávajícího. Dále je popsáno, jakým způsobem probíhá sběr požadavků, analýza a návrh modernizovaného systému. Následuje popis vybraných technologií, časový rozsah a náklady na vytvoření nového modernizovaného systému. Na to navazuje podrobný popis implementace, který poslouží jako dokumentace pro kohokoliv, kdo bude mít zájem na systému dále pracovat. V poslední části jsou zhodnoceny výhody a přínosy nového systému.

Výsledkem práce je snadno rozšiřitelný systém vyvíjený v souladu s aktuálními metodami softwarového inženýrství.

Kapitola 1

Cíl práce

Prvním cílem bakalářské práce je analyzovat stávající informační systém společnosti Workswell. Na základě analýzy poté navrhnout úpravy systému, které budou diskutovány se zaměstnanci a vedením společnosti. Dalším dílčím cílem je zhodnotit, jaký přínos pro společnost budou navržené změny mít a vysvětlit, proč jednotlivé změny realizovat, jaká bude jejich efektivita. Z dohodnutých změn vyplyne stěžejní cíl práce, kterým je implementace firemního systému za použití moderního frameworku. Následným dílčím cílem je zaškolení zaměstnanců a jejich seznámení s úpravami a novými funkcionalitami. Neméně důležitým cílem je zajištění snadné rozšiřitelnosti systému dalšími studenty nebo zaměstnanci společnosti. Tato práce společnosti Workswell poslouží jako východisko pro provádění změn a přidávání nových částí systému.

Kapitola 2

Firemní systémy a webové aplikace

2.1 ERP systémy a jejich výhody

ERP je zkratkou pro Enterprise Resource Planning, česky plánování podnikových zdrojů. Podle [2] jsou ERP systémy softwarové aplikace, ve kterých je zakódována hluboká znalost podnikových procesů a praktik. Informační technologie dlouhodobě napomáhají ke snižování nákladů a zvyšování výkonu automatizací opakujících se podnikových činností. Výhody ERP systému jsou:

- operační
  - snížení nákladů
  - zvýšení produktivity
  - zkrácení cyklu firemních procesů
  - zvýšení kvality
  - zlepšení zákaznických služeb
- manažerské
  - vyšší úroveň řízení zdrojů
  - zefektivnění plánování a rozhodování
  - zvýšení výkonu
- strategické
  - podpora růstu společnosti
  - zlepšení týmové spolupráce ve společnosti
  - obchodní inovace

  - posílení cenové konkurenceschopnosti
  - diferenciace výrobků zahrnující výrobky na míru
  - vytváření externích vazeb (na zákazníky a dodavatele)
- IT infrastruktura
  - vybudování flexibility společnosti pro současné i budoucí změny
  - snížení IT nákladů
  - rozšíření schopností IT infrastruktury
- organizační
  - podpora smysluplných organizačních změn
  - snazší pochopení organizační struktury společnosti (například pro nové zaměstnance)
  - přenesení kompetencí na vybrané zaměstnance
  - budování společných vizí

2.2 MVC architektura a framework Django

MVC je architektonický vzor, který je součástí populárních webových frameworků jako je Nette pro PHP, Ruby on Rails pro Ruby nebo Django pro Python. Odděluje od sebe datový model, řídící logiku a výstup aplikace. Řeší problém prolínání logických operací a renderování výstupu v jednom souboru zdrojového kódu. Takový soubor obsahuje databázové dotazy a logiku v programovacím jazyku (například PHP nebo Python) a dále HTML tagy, které jsou s programovacím jazykem míseny. Zdrojový kód se z tohoto důvodu špatně čte, udržuje a rozšiřuje. Při použití MVC architektury obsahuje zdrojový kód s řídící logikou pouze jeden programovací jazyk (například PHP nebo Python) a zdrojový kód stránky zobrazené uživateli vypadá jako HTML stránka. [3]

„Celá aplikace je rozdělena na komponenty 3 typů, hovoříme o Modelech, View (pohledech) a Controllerech (kontrolerech), od toho MVC.“ [3] Typ Model obsahuje pouze business logiku aplikace, to znamená databázové dotazy, validaci, výpočty, vztahy mezi objekty a podobně. Model se nezajímá o to, odkud data přichází a ani jak budou zobrazena koncovému uživateli. Ve frameworku Django je využíváno ORM. ORM je zkratka pro objektově relační mapování a podle [4] je to technika, při níž jednoduše řečeno dochází ke konverzi mezi objekty v paměti a tabulkami databáze. Například bez ORM by musel vývojář napsat SQL dotaz `SELECT * FROM tasks WHERE priority=5`, který vrací úkoly s prioritou 5. Ekvivalentní dotaz v Djangu pomocí ORM napsaný v jazyku Python je `Tasks.objects.filter(priority=5)`. To je pro

vývojáře méně časově náročné a ve zdrojovém kódu je použit pouze jeden jazyk.

Část View zobrazuje výstup webové aplikace uživateli. Zobrazení dat uživateli probíhá pomocí HTML šablony s tagy značkovacího jazyka (poskytován frameworkem, například Djangem). Značkovací jazyk „umožňuje do šablony vkládat proměnné, případně provádět iterace (cykly) a podmínky“. Stejně jako je tomu u Modelu, View se nezajímá o to, odkud data přichází, pouze je zobrazuje. [3]

Controller je zprostředkovatel mezi částmi Model a View a zároveň koncovým uživatelem. „Drží tedy celý systém pohromadě a komponenty propojuje.“ [3]

Framework Django využívá MVC architektury, ale nazývá komponenty trochu jinak. Modely jsou stále nazývány Models, na druhou stranu Views se říká Templates a pro Controllers se používá výraz Views. Podle [5] je Model nezbytným zdrojem informací o datech v aplikaci. Obsahuje základní atributy a chování dat, která ukládáme. Obecně je každý Model přiřazen jedné tabulce v databázi. View je funkce nebo třída, která přijímá požadavek, anglicky request a vrací odpověď, anglicky response. [6] Migrace je potom funkcionalitou Djanga, pomocí které lze promítat změny provedené na modelech (například přidání atributu, smazání modelu a podobně) do databázového schématu. Jsou navrženy tak, aby fungovaly převážně automaticky. S frameworkem Django nejlépe funguje Postgres databáze. Podle dokumentace je PostgreSQL nejlepší ze všech databází z hlediska podpory schématu. [7]

2.3 Application Programming Interface

API je zkratka anglického názvu Application Programming Interface, který v češtině znamená aplikační programové rozhraní. Programy klienta používají API ke komunikaci s webovými službami. Obecně řečeno API odkrývá množinu dat a funkcí pro usnadnění interakce mezi počítačovými programy a umožňuje jim výměnu informací. [8] API rozhraní využijeme pro AJAX, což je „technologie výměny informací mezi klientem a serverem přes HTTP protokol bez nutnosti znovunačtení celé stránky při každém požadavku.“ [9] Jednou z funkcí webového API je serializace dat. Ta umožňuje převést komplexní datové struktury jako je instance modelu nebo seznam instancí modelů na datové typy jazyku Python, které mohou být snadno dále převedeny na formáty JSON, XML nebo další formáty dat. Serializace dále umožňuje tzv. „deserializaci“, při níž jsou formáty dat JSON, XML apod. převedeny zpátky na komplexní datové struktury poté, co jsou příchozí data validována. [10]

2.4 Zabezpečení systému ve frameworku Django

Cross-site scripting (zkráceně XSS) je útok, ve kterém se útočníkovi podaří spustit škodlivý skript v prohlížeči ostatních uživatelů (tedy na straně klienta). Toho je většinou docíleno uložením škodlivého skriptu v databázi, odkud je stažen a zobrazen uživatelem webové aplikace. Útočník také může přesvědčit uživatele, aby kliknul na odkaz, který spustí JavaScript kód útočníka v prohlížeči uživatele. Použitím Django šablon je webová aplikace chráněna před většinou XSS útoků. [11]

Cross site request forgery (zkráceně CSRF) útoky umožňují útočníkovi provádět akce za jiné uživatele bez jejich vědomí a souhlasu. Django nabízí ochranu proti většině typů CSRF útoků za předpokladu, že je ochrana povolena a používána na správném místě. CSRF ochrana funguje na základě kontroly tajného kódu při každém požadavku metody POST. Ten zajišťuje, že útočník nemůže odeslat formulář s daty do webové aplikace vydávající se za jiného uživatele, který o odeslání dat neví. Útočník by musel znát tajný kód, který je specifický pro každého uživatele a obtížně prolomitelný. [11]

SQL injection je typ útoku, ve kterém je útočník schopen spustit libovolný SQL příkaz v databázi aplikace. To může vést ke ztrátě nebo úniku dat společnosti. Pro přístup k databázi Django nabízí rozhraní Querysets, které je chráněno proti SQL injection útokům, protože jejich databázové dotazy jsou poskládány pomocí parametrizace. SQL kód dotazu je oddělen od parametrů dotazu. Navíc parametry mohou být dodány uživatelem systému, z toho důvodu mohou být nebezpečné, proto jsou escapovány databázovým ovladačem. [11] „Escapování je převod znaků majících v daném kontextu speciální význam na jiné odpovídající sekvence“. [12]

Kapitola 3

Analýza a zhodnocení současného systému

V následující kapitole je popsán informační systém, který společnost Workswell dosud využívá, analýza jeho současného stavu a popis jeho nedostatků. Dále kapitola předkládá a zdůvodňuje argumenty pro změny a nově navržená řešení.

3.1 Analýza databáze

V databázi chybí kaskádové mazání a aktualizace podle cizích klíčů, to znamená, že nejsou zachyceny vztahy mezi objekty. Existence některých tabulek v databázi dává smysl pouze v relaci s jinými objekty, bez nich nemá cenu je ukládat. Například pro modely `Zákazník` a `Adresa doručení` by bylo žádoucí, aby při smazání účtu zákazníka došlo i ke smazání adresy doručení, tím k smazání jeho osobních údajů. Jiným příkladem, tentokrát ze systému společnosti, je vztah mezi modely `Zásoba` a `Produkt`. Zásoba představuje konkrétní kus na skladě, zatímco produkt je pouze popis výrobku. Je nežádoucí, aby bylo možné smazat produkt, na který je odkazována nějaká zásoba. Dále chybí unikátní a tzv. „not null“ restrikce. Unikátní by mělo být například sériové číslo zásoby. Not null omezení očekáváme v tabulce Zásoby u cizího klíče na Produkt. Zásoba bez produktu nemůže existovat a dvě zásoby nemohou mít stejné sériové číslo. Tato omezení jsou důležitou funkcionalitou databáze a celého systému, ale v aktuálním systému chybí. Pojmenování tabulek je matoucí, většina z nich navíc již není používána. Dochází k opakování dat, například atributy adresy se vyskytují v několika tabulkách. Správné řešení je mít zvláštní tabulku `Adresa` a na ní odkazovat cizím klíčem. Některé tabulky jsou zbytečné, příkladem je tabulka spojující sestavu a zásobu s názvem `set_stock`. Ta by vůbec nemusela existovat. Zásoba může být maximálně v jedné sestavě, stačí v ní proto ukládat cizí klíč sestavy. Potom jsou vztahy v databázi jednodušší a přehlednější.

3.2 Analýza webové aplikace

Pro vývoj systému byl použit programovací jazyk PHP, skriptovací jazyk Javascript, CSS a markup jazyk HTML. Není použit žádný framework ani MVC architektura, business logika není oddělena od prezentační vrstvy. To má za následek malé možnosti rozšíření. Chybí objektově relační mapování (zkráceně ORM). Z toho důvodu při úpravách a přidávání atributů objektům systému je třeba takové úpravy provést ručně na dvou místech, nejprve v databázi, poté i v kódu. Ve zdrojovém kódu nejsou dodržovány zásady a osvědčené postupy programování. Objevuje se zde míchání českého a anglického jazyka v názvech proměnných, funkcí, tříd, komentářů a konstant. Dále dochází ke střídání jmenných konvencí, konkrétně velbloudí notace a podtržítkové notace. V několika případech se vyskytuje toto střídání v rámci jednoho názvu, například `getMaterial_number`. Převážná část kódu je duplikována, to je nejlépe vidět na funkci ověřující přihlašovací údaje nebo při psaní SQL dotazů. Při změně je potřeba dělat úpravy na několika místech najednou, což zbytečně zatěžuje zaměstnance, který tráví čas něčím, co nepřináší do systému nic nového. Pokud bychom část systému, která se stará o rezervace, chtěli zpřístupnit nové skupině uživatelů, museli bychom projít každý soubor v modulu a na správném místě provést úpravu. Projekt je strukturován pouze na úrovni hlavních částí systému. Konkrétněji už se nijak nedělí, složka modulu tedy obsahuje pouze soubory, ne adresářovou strukturu. Lepším přístupem je mít další rozdělení podle funkčnosti, oddělit třídy s databázovými modely, šablony webových stránek, pomocné funkce a podobně. V projektu rozhodně platí princip rozbitých oken, který ve významu softwarového inženýrství říká, že v zanedbaném kódu nemá cenu nebo dokonce ani nejde přidávat nové věci správným způsobem. Navíc neexistuje žádná forma dokumentace, ve společnosti se v systému vyzná pouze jeden člověk. Z názvosloví není jasné, k čemu jaké části systému slouží a jak fungují. Pomocí metody s názvem `getNote` (v češtině „získej poznámku“) je možné získat informaci o tom, pro koho je určena sestava. Způsob, jakým je přistupováno k datům v databázi, je zastaralý a nebezpečný. Chybí ochrana i proti těm nejzákladnějším útokům jako je SQL injection nebo CSRF. Mohlo by tedy dojít k úniku dat uživatelů, která jsou v systému uložena.

3.3 Analýza uživatelského prostředí

Uživatelské prostředí není intuitivní, nemá jednotný jazyk, některé části jsou v češtině, jiné v angličtině, další ve slovenštině. Pokud by se firma rozhodla expandovat nebo přijmout zaměstnance nemluvícího česky, nedokázal by se v systému orientovat. V některých případech chybí popis chybových hlášek a není jasné, co uživatel udělal špatně. Dále chybí logování errorů, vývojář neví o chybách, na které uživatel narazil, pokud mu je sám neoznámí. Při plánování

dovolených jsou kontrolovány pouze státní svátky z roku 2017, protože ty jsou tam předdefinované. Pro uživatele toto prostředí rozhodně není komfortní. Při přijímání nových zaměstnanců firma jejich zaškolováním ztrácí zbytečně moc času. Pokud by systém byl více intuitivní a jednodušší, nový zaměstnanci by o něm získali přehled za daleko kratší dobu.

3.4 Údržba zdrojového kódu

Při vývoji se nepoužívá správa verzí, nové verze se nasazují překopírováním změněných částí kódu na produkční server. I přesto, že existuje vývojářské a produkční prostředí, po zkopírování nového kódu a tím přepsání kódu starého v produkčním prostředí, neexistuje možnost vrátit se ke starší verzi systému. Tento návrat se využívá v momentě, kdy se na produkční verzi objeví chyba, kterou je potřeba neprodleně opravit. Nástroje pro správu verzí umožňují nejen uchování historie vývoje kódu, ale také souběžný vývoj systému několika osobami najednou. To by bylo využito, pokud by firma v budoucnosti vypsala téma bakalářské práce pro více studentů. Každý potom může vyvíjet systém na své vývojové větvi a kód po dokončení spojit.

3.5 Shrnutí

Systém je zastaralý, minimálně rozšiřitelný a se spoustou bezpečnostních rizik. Na systému pracovali různí vývojáři, kteří se s programováním seznamovali až v průběhu jeho vývoje a nedodržovali metody a osvědčené postupy vývoje software, což se negativně projevilo na kvalitě tohoto systému, jenž je v současnosti potřeba modernizovat. Pokud by byla prováděna úprava kódu stávajícího systému, byla by časově natolik náročná, že by přestala být smysluplná. Jako výhodnější způsob se tedy jeví přepsání celého systému do moderního frameworku, což bude časově mnohem efektivnější. Po prezentaci výše uvedených argumentů a diskusi s vedením firmy byl navržený postup odsouhlasen.

Kapitola 4

Návrh nového systému

Následující kapitola přináší informace o výběru technologií použitých pro vývoj nového systému, stanovení funkčních a nefunkčních požadavků, výpočet nákladů jednak na vývoj a na provoz aplikace. Závěr kapitoly se zabývá návratností investice do navrhovaných změn systému.

4.1 Výběr technologií

Nový systém bude napsán v open source frameworku pro vývoj webových aplikací s názvem Django. Framework je napsaný v programovacím jazyce Python a využívá MVC architektury. Pro vytvoření frontend části aplikace bude použit Bootstrap a JQuery společně s HTML, CSS a jazykem Javascript. Bootstrap je CSS framework, který je výhodné použít u projektů, kde není vyžadován vlastní, originální vzhled. Dalším řešením by bylo použít například Javascriptovou knihovnu React, ale vzhledem k tomu, že systém nevyžaduje složité uživatelské rozhraní, budou pro vývoj systému navržené technologie dostačující. Web není používán na mobilních zařízeních, proto není potřeba aby byl responzivní.

Python a Django jsou populární nástroje pro tvorbu webových aplikací, které mohou být atraktivní pro někoho dalšího, kdo se na vývoji bude chtít v budoucnu podílet. Při vývoji bude kladen důraz na business logiku aplikace více než na prezentační vrstvu. Systém není natolik rozsáhlý, aby bylo potřeba rozdělení na na frontend a backend části. Pokud by se však takový požadavek v budoucnu objevil, z Django frameworku lze jednoduše udělat API pomocí REST knihovny, tedy funkční backend aplikaci. Django navíc nabízí prostředí Django admin, kde mohou pověření uživatelé spravovat obsah dat nad rámec běžných funkcí systému bez znalosti databází nebo zdrojového kódu systému. Aktuálně ve firmě jakoukoliv takovou úpravu může provést pouze jeden člověk. Navíc pomocí SQL dotazu. Při komplexnosti požadavku může být napsání SQL dotazu časově náročné.

4.2 Proces vývoje aplikace

Při vývoji aplikace bude použit systém pro správu verzí Git se vzdáleným repozitářem na Gitlabu, kde bude také založena Wiki projektu, obsahující dokumentaci projektu a krátké návody na spuštění a údržbu. Aplikace bude průběžně nahrávána na vzdálený server, konzultována a testována vedením a zaměstnanci společnosti. Před zavedením do produkce bude potřeba provést migraci reálných dat z aktuálního systému a systém otestovat.

4.3 Funkční požadavky

- Po zadání emailu a hesla bude uživatel přihlášen do systému.
- Povinná pole ve formulářích musí být viditelně označena.
- Frontend systému nemusí mít responzivní design, protože všichni přistupující uživatelé budou používat stolní počítač.
- Nepřihlášený uživatel musí být vždy přesměrován na přihlašovací obrazovku.
- Přihlášený uživatel může být při přístupu na přihlašovací obrazovku přesměrován na domovskou stránku.
- Přihlášený uživatel se může odhlásit.
- Uživatel musí mít zakázán přístup do částí systému, do kterých nemá přístupové právo.
- Uživatel musí mít přístup do částí systému, do kterých má právo vstupovat nebo je mu přiřazena skupina práv, která požadované právo obsahuje.
- Systém musí zkontrolovat, zda položka, kterou se snaží uživatel rezervovat, není v systému vedena jako rezervovaná.
- Systém by měl mít funkci vyhledávání, která umožní uživatelům rychlý přístup k informacím uloženým v systému.
- Systém by měl mít domovskou obrazovku, která uživatele informuje o aktuálních událostech, rozhodnutích a novinkách ve společnosti.
- Administrátor musí mít možnost zakázat uživateli přístup do systému (bez nutnosti smazání jeho účtu).
- Stránka „Absence“ by měla zobrazovat přehled absencí ve firmě v podobě kalendáře. Záznamy kalendáře by měly být barevně odlišeny podle dostupnosti zaměstnance na telefonu.

- Stránka „Rezervace“ by měla zobrazovat přehled rezervací ve firmě v podobě tabulky. Řádky tabulky by měly být barevně odlišeny podle aktuálnosti rezervace.
- Stránka „Výrobky“ umožní přiřazovat komponenty k výrobkům pomocí 2D čtečky nebo manuálním zadáním hodnoty QR kódu.
- Stránka „Úkoly“ by měla zobrazovat úkoly ke splnění přihlášeného uživatele.

4.4 Nefunkční požadavky

- Systém bude k dispozici v režimu 24x7.
- Systém musí umožnit souběžnou práci až 50 zaměstnanců v jeden okamžik.
- Uživatel si musí zvolit heslo, které obsahuje alespoň 8 znaků, není podobné ostatním osobním údajům uživatele, není běžně zvolené heslo a neskládá se pouze z číslic.
- Databáze systému musí být zálohována tak, že po obnovení po případném výpadku budou ztracena pouze data vytvořená v posledních 24 hodinách.
- Databáze umožní alespoň 50 připojení najednou.
- Údržba databáze bude probíhat maximálně 1 hodinu do měsíce.
- Databázi musí být možné obnovit ze zálohy provedené v posledních 4 dnech.
- Velikost úložiště databáze musí být alespoň 50 GB.
- Doba přesunu z jedné stránky na druhou nebude delší než 3 sekundy.
- Datum ve formulářích nebude zadáváno ručně, ale pomocí výběru z nabídky kalendáře.
- Pro výběr ve formuláři z existujících položek uložených v systému bude použit našeptávač s funkcí autocomplete.

4.5 Náklady na vývoj

Pro stanovení odhadu nákladů na vývoj aplikace byla práce na projektu rozdělena do dílčích úkolů, u kterých byla časová náročnost odhadnuta podle předchozích pracovních zkušeností. Takovému postupu se říká Work Breakdown Structure (zkráceně WBS).

- Příprava (54 hodin)
  - Analýza a sběr požadavků (30 hodin)
  - Příprava projektu, instalace potřebných balíčků (8 hodin)
  - Nasazení na server (4 hodiny)
  - Multijazyčnost, české překlady (8 hodiny)
  - Middleware (4 hodiny)
- Testování (30 hodin)
- Uživatelé (32 hodin)
  - Přihlašování pomocí emailu (4 hodiny)
  - Model uživatele a pobočky (4 hodiny)
  - Přiřazování práv, vytváření práv, vytváření skupin práv (8 hodin)
  - Vytvoření šablony a pohledu pro domovskou stránku (8 hodiny)
  - Zobrazení a zasílání notifikací (8 hodin)
- Absence (34 hodin)
  - Vytvoření modelů (6 hodin)
  - Vytvoření šablon (4 hodiny)
  - Vytvoření formuláře pro podání absence (2 hodiny)
  - Kalendář absencí (6 hodin)
  - Počítání odpracovaných dnů, nároky na dovolenou, přesun dní do nového roku, čerpání „sick days“ apod. (8 hodin)
  - Vyřizování dovolených (8 hodin)
- Komponenty (8 hodin)
  - Vytvoření modelů (4 hodiny)
  - Vytvoření šablon (4 hodiny)
- Rezervace (28 hodin)
  - Vytvoření modelů (6 hodin)
  - Vytvoření šablon (6 hodin)
  - Vytvoření formuláře rezervování položky (8 hodin)
  - Kontrola dostupnosti položky před jejím rezervováním (8 hodin)
- Výrobky (56 hodin)
  - Vytvoření modelů (12 hodin)

  - Vytvoření šablon (6 hodin)
  - Formulář pro vytváření šablon výrobků (6 hodin)
  - Načítání komponent a produktů pomocí 2D čtečky (32 hodin)
    * Přiřazení komponenty k instanci produktu (6 hodin)
    * Konfigurace 2D čteček (6 hodin)
    * Vytvoření dat štítku komponenty a jejich export do programu Dymo Label (4 hodiny)
    * Vytvoření šablony pro štítek pomocí programu Dymo Label (12 hodin)
    * Tisknutí štítků pomocí tiskárny (4 hodiny)

- Úkoly (20 hodin)
  - Vytvoření modelů (6 hodin)
  - Vytvoření šablon (6 hodin)
  - Řešitel úkolů (8 hodin)

- Objednávky (18 hodin)
  - Vytvoření modelů (6 hodin)
  - Vytvoření šablon (6 hodin)
  - Uložení šablony objednávky a nahrání dat objednávky z šablony (6 hodin)

Celková předpokládaná časová náročnost na analýzu, vývoj a testování činí 280 hodin včetně rezervy.

4.6 Náklady na provoz a návratnost investice

Pro provoz aplikace společnosti bude použita služba Digital Ocean, ve které provoz serveru hostujícího Django webovou aplikaci stojí $5 měsíčně, to je v přepočtu asi 125 Kč. Dále bude využívána databáze, za kterou si Digital Ocean účtuje $15 dolarů měsíčně, přibližně 375 Kč. Roční provoz aplikace tak činí cca 6,000 Kč s DPH. Zakoupení 2D čtečky pro snadnější a efektivnější skládání výrobků vyjde přibližně na 2,200 Kč. Společnost se rozhodla zakoupit 3 čtečky různých značek, v celkové hodnotě cena cca 6,500 Kč s DPH.

Aktuálně si firma nevede metriky, podle kterých by se dala přesně spočítat návratnost investice. Zavedením nového systému dojde k úspoře času a tím zvýšení produktivity práce jednotlivých zaměstnanců. Při označování přijatých komponent číslem a názvem komponenty dojde k odbourání ručního přepisování dat ze systému. Bylo odhadnuto, že pro 100 přijatých položek je v průměru potřeba vynaložit více než hodinu práce, stažení a vytisknutí štítků

trvá maximálně několik minut. Díky tomu bude možné přijímat větší objednávky. Dříve rostl s počtem přijatých dílů čas potřebný k jejich přepsání na štítek. To u nového způsobu neplatí, v případě větší objednávky se pouze vygeneruje větší soubor s více daty, rozdíl mezi přijetím 1 dílu a 100 dílů je minimální. Zvýšený počet objednávek firma pokryje stávajícím počtem zaměstnanců, nebude muset přijímat a financovat nové zaměstnance, může se soustředit na vývoj lepších produktů. K další časové úspoře dojde při sestavování výrobku. Použitím 2D čtečky se výrazně zrychlí výběr komponentů a tím i sestavení výrobků. V původním systému zaměstnanec složil průměrně 20 výrobků za týden, zatímco nyní je to o 4–5 výrobků více. Návratnost investice je v tomto případě vidět okamžitě.

Při vývoji nového systému také máme za úkol minimalizovat pravděpodobnost vzniku škody, například únikem nebo ztrátou dat. V neposlední řadě nový systém eliminuje škody způsobené ztrátou dat nebo napadením systému. Za únik dat zákazníků by firma mohla zaplatit pokutu až několik tisíc korun.

Kapitola 5

Realizace

Následuje kapitola zabývající se implementací webové aplikace. V úvodu je popsána struktura projektu a nástroje, které jsou při vývoji systému využity. Dále se zabývá vysvětlením různých uživatelských práv, která v systému existují, popisem nových funkcionalit, které byly nově do systému přidány. Součástí kapitoly je také popis jednotlivých modulů v systému.

5.1 Struktura projektu

Při vytvoření projektu vygeneruje Django základní adresářovou strukturu, konfiguraci databáze, nastavení a podobně. Ve složce pojmenované stejně jako název aplikace najdeme soubory `manage.py` (nástroj příkazové řádky, který umožňuje interakci s Django aplikací různými způsoby), `settings.py` (soubor s konfigurací projektu) a `urls.py` (deklarace URL adres). [13]

Projekt je rozdělen do tzv. aplikací, anglicky „apps“. Aplikace je jeden modul systému zajišťující jednu určitou funkci systému. Příkladem je `Reservations`, která se stará o rezervace, `Orders`, která se stará o objednávky apod. Každá jedna aplikace obsahuje několik různých souborů, které zajišťují různé funkce v systému. `Models.py` obsahuje modely systému, které představují tabulky databáze, vztahy mezi nimi a různá logická omezení. Tato omezení jsou definována v souboru s názvem `validators.py`. Ve `views.py` jsou pohledy, které zachází s http požadavky. V souboru `admin.py` jsou registrovány modely, které se zobrazí v administračním prostředí Djanga. `Urls.py` se starají o navigaci požadavků aplikací. Mohou je přesměrovávat do jiných částí aplikace nebo rovnou odkazovat na konkrétní pohled. Různé pomocné funkce, které lze využít většinou v modelech nebo pohledech, jsou uloženy v souboru `utils.py`. `Forms.py` potom ukládají formuláře, které slouží k získání dat od uživatele a jejich následnému převedení na Python slovník (datová struktura pro ukládání dat, více na [14]). Podobnou funkci mají serializery uložené v `serializers.py`. Převádí data z formátu JSON do Python slovníku a naopak.

5.2 Použití middleware

Middleware je systém, který umožňuje pozměnit vstupy a výstupy webové aplikace Django. Každý middleware komponent je zodpovědný za provádění specifické funkce. Django například zahrnuje middleware s názvem `AuthenticationMiddleware`, který asociuje uživatele s požadavkem za použití sessions. [15]. Pro lepší fungování aplikace byly do systému přidány následující middleware komponenty.

5.2.1 Login required middleware

Tento middleware přesměruje každého nepřihlášeného uživatele na přihlašovací stránku. Vyjímkou jsou url cesty definované v nastavení aplikace. Toho lze využít například pro zpřístupnění dotazníku spokojenosti zákazníkům. Taková stránka nebude vyžadovat přihlášení. Zároveň middleware přesměruje přihlášeného uživatele na domovskou stránku, pokud se snaží přistoupit na přihlašovací obrazovku.

5.2.2 Set language middleware

Při použití základního nastavení Django aplikace je automaticky vybrán jazyk, který používá prohlížeč. To zajišťuje `LocaleMiddleware`. V systému požadujeme, aby měl uživatel možnost zvolit si svůj jazyk. K tomu je v systému určený middleware s názvem `SetLanguageMiddleware`. Ten zvolí jazyk stránky zobrazené uživateli podle atributu `language` (jazyk) v modelu uživatele. `AuthenticationMiddleware` automaticky zjistí, který uživatel si chce stránku zobrazit. Podle [15] přidá atribut uživatele (což je objekt reprezentující aktuálně přihlášeného uživatele) do každého příchozího objektu `HttpRequest`, který reprezentuje HTTP požadavek.

5.3 Uživatelská práva v systému

Následuje přehled a vysvětlení firemních přístupových práv v systému. Přidáním práva uživateli je mu umožněna zmíněná akce. Práva lze seskupit do skupiny práv, například práva pro určitou skupinu zaměstnanců, práva pro určitou pobočku.

Pro každý objekt v systému existují práva povolující nebo zakazující manipulaci s ním. Jsou to práva „přidat“, „upravit“, „zobrazit“ a „smazat“. Kromě těchto základních práv jsou v systému přidána navíc práva

- `resolve_vacation` (rozhodnutí o poskytnutí dovolené),
- `request_vacation` (žádost o dovolenou),
- `use_sick_day` (žádost o sick day),

- `view_reservation_history` (prohlížení historie rezervací),
- `assign_task` (přiřazení úkolu jinému uživateli) a
- `assemble_product` (sestavování výrobků z komponent).

5.4 Sestavování výrobků pomocí šablon

Současný technologický postup sestavování výrobků je zbytečně náročný. Zaměstnanec si u sebe v kanceláři vybere, jaké položky pro sestavení použije, následně jde do skladu, kde si je může vyzvednout. V praxi nastávají situace, kdy vybraná položka na skladě není, protože v danou chvíli není ještě dodána. Zaměstnanec se v tom případě musí vrátit do kanceláře, vybrat položku jinou a proces opakovat. Další zbytečně namáhavý proces je u přijímání objednávek. V současnosti pokud firma obdržela díly na sestavení produktu, bylo zboží označováno vytisknutím štítku, na který bylo ručně dopsáno sériové číslo z databáze. V začátcích fungování firmy byl tento postup přijatelný, počet dodaných dílů byl v řádech jednotek až dvou desítek. S postupným růstem firmy se objednávky zvětšují a současný systém nemá šanci držet krok s reálnými potřebami. Po zhodnocení a konzultacích došlo k implementaci systému nového.

Nově je vygenerován štítek s unikátním QR kódem pro každou naskladněnou položku. Pro tisk štítků je využívám software výrobce tiskárny s názvem Dymo Label. Ten nabízí několik možností importu dat. Nejjednodušším a nejvhodnějším pro naše účely je textový soubor, který obsahuje hlavičku dat na prvním řádku a jednotlivá data na řádcích dalších. Každý záznam je oddělen středníkem. Uživatelé tak mohou jednoduše zkontrolovat, zda jsou data v souboru správná, případně provést potřebné úpravy. Do systému bylo přidáno generování datových souborů pro jednotlivou položku i celý dodací list. Při přijetí objednávky stačí jedním kliknutím stáhnout soubor, nahrát ho do programu Dymo Label, vytisknout štítky a nalepit je na zboží. Na štítku je kromě QR kódu dále pro lepší přehled i jméno produktu, číslo materiálu, sériové číslo a hodnota QR kódu pro případ, že QR kód z jakéhokoliv důvodu nelze načíst.

Při výběru zboží je pro sestavení použita 2D čtečka a počítač, zaměstnanci stačí vybrat komponent a naskenovat ho, tím se přidá v systému do sestavy. Stránku pro načítání komponent v novém systému ukazuje obrázek 6.7. Nemůže se tedy stát, že by si vybral to zboží, které ještě není na skladě. Další výhodou je grafické zobrazení šablony.

Šablona je předpis, který udává, z kolika a jakých komponent se produkt skládá. Jedna položka šablony může být splněna několika různými součástkami. Například šablona nám říká, jak sestavit bezkontaktní kameru za použití jedné kamery a jednoho dronu. Dron i kamera potom mohou být různých značek a typů, které jsou přesně definovány pro každou položku šablony. Použití šablon znamená méně starostí a chyb, což vede k větší produktivitě práce.

Kapitola 6

Moduly v systému

6.1 Společné

Modul `Commons` (společné) poskytuje hlavně programátorům systému pomocné funkce, modely a číselníky. Číselník představuje výčet pevně daných hodnot. Například pro pohlaví je to muž a žena. Důležitou částí je model tzv. „měkkého smazání“ objektu. Tento model lze z uživatelského pohledu smazat, ale v databázi nadále zůstává. Toho se využívá pro pročištění uživatelského prostředí od nepoužívaných dat, ale ponechání informací, které by se z účetních nebo podobných dalších důvodů mohly hodit.

Obecný modul má také funkci vyhledávání v celém systému, data se snaží najít v modulech dovolené, produkty, komponenty a rezervace. Funkce vyhledávání výrazně usnadní přístup k informacím, které jsou uloženy v databázi systému. Uživatel tak neztrácí čas prohledáváním irelevantních částí systému.

Validátor je funkce, jehož jediným vstupem je hodnota atributu modelu. Vrací hodnotu pravda, pokud hodnota odpovídá požadované podmínce a nepravda, pokud ne. Atributům modelů je ve frameworku Django možné přiřadit jeden nebo více validátorů. Například validátor s názvem `only_hours` zkontroluje z pohledu backendu, zda uživatel zadal opravdu jenom hodinu a ne navíc i minuty nebo vteřiny. Dalším příkladem by byl validátor formátu telefonního čísla využívající regulárních výrazů. Validátory jsou uloženy v obecném modulu, aby je bylo možné použít u jakéhokoliv modelu v systému.

Objekty `Address` (adresa) a `Country` (stát) slouží k zaznamenání fyzické adresy, adresa má atributy

- `street` (ulice),
- `house_number` (číslo domu),
- `zip` (poštovní směrovací číslo),
- `city` (město nebo obec),

- `state` (stát),
- `region` (kraj nebo oblast),
- `country` (cizí klíč na zemi)

a stát má atribut

- `name`.

`Currency` (měna) je jednoduchý objekt zaznamenávající

- `value` (zkratka měny, například „EUR“, „CZK“ nebo „USD“).

Je použit samostatný model pro možnost dynamického přidání nových měn.

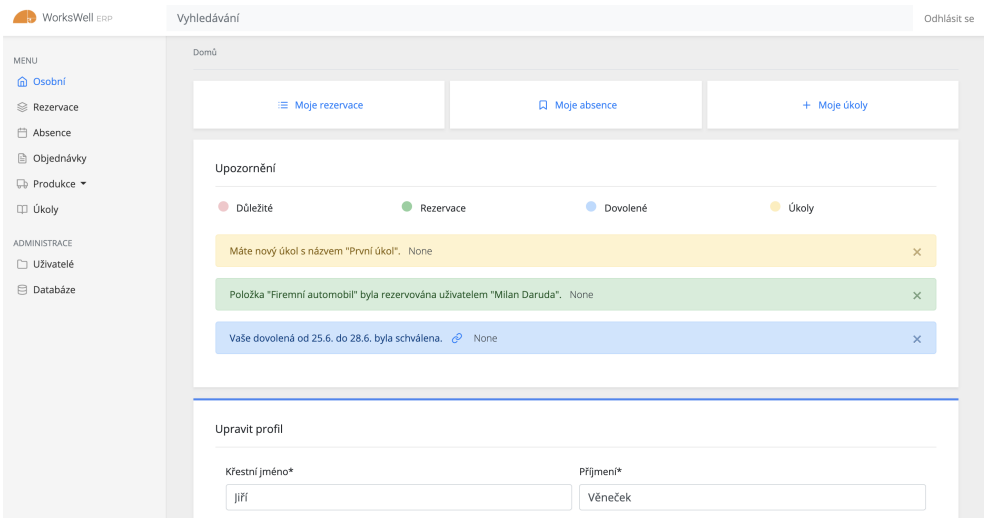
V neposlední řadě se modul stará o notifikace uživatelů. `Notification` (notifikace) je v systému objekt s atributy

- `message` (zpráva notifikace),
- `recipient` (cizí klíč příjemce zprávy),
- `created` (datum a čas zaslání notifikace),
- `link` (odkaz notifikace),
- `type` (typ notifikace).

Notifikace vzniká při přijetí nebo zamítnutí dovolené uživatele, při rezervaci položky zaměstnancem, udělení nového úkolu nebo i z jiného důvodu. Notifikaci může například vytvořit administrátor pro upozornění zaměstnance zadáním zprávy „Nezapomeňte na schůzku 19.5. v 10:00“. Seznam notifikací se nachází na hlavní stránce, obrázek 6.1, a je zobrazen podle přihlášeného uživatele. V administračním prostředí je možné zobrazit si notifikace všech zaměstnanců. Notifikace budou zároveň využívány pro informování zaměstnanců o změnách a novinkách v systému.

6.2 API

Modul API poskytuje rozhraní schopné komunikovat nejen s ostatními systémy, ale i uvnitř firemního systému. Například při přidávání zboží do sestavy je po naskenování kódu odeslána žádost z modulu produktů na endpoint modulu API, který zboží do sestavy přidá na pozadí. Uživatel tak nemusí čekat na načtení stránky a může pohodlně skenovat další položky. Další využití API modulu je při vyplňování formuláře rezervace. Pokud si uživatel vybere



Obrázek 6.1: Hlavní stránka uživatele s nejnovějšími upozorněními a formulářem pro úpravu osobních údajů.

položku z formuláře, na pozadí proběhne dotaz API modulu na aktuální dostupnost položky. Pokud položka dostupná není, uživatel je informován ještě před odesláním formuláře.

Modul využívá pohledů a serializace dat poskytnuté balíčkem Django REST framework. API je možné rozšířit tak, aby systém mohl komunikovat s jinými užívanými systémy ve společnosti jako je ZOHO Task Management. Systém potom může například po přihlášení uživatele zažádat ZOHO API o úkoly uživatele a upozornit ho na ty, u kterých se blíží lhůta pro jejich splnění.

6.3 Uživatelé

V systému bude možné přidat nové uživatele s firemním emailem, jménem, příjmením a telefonním číslem. Uživatelům budou přiřazována práva nebo skupiny práv. Právo je objekt, který popisuje, jaké právo uživatel má. Různé pohledy vyžadují různá práva. Příkladem práva je právo s názvem `request_vacation`. Každý uživatel s takovým právem může zažádat o dovolenou. Pokud právo nemá, systém mu zablokuje přístup na stránku a ve většině případů ani nezobrazí odkaz směřující na danou stránku. Zaměstnanec s právem `resolve_vacation` má potom na detailu dovolené možnost schválit nebo zamítnout dovolenou. Aby práva nemusela být přiřazována každému zaměstnanci jednotlivě, existují také skupiny práv. Vytvoření skupin práv bude v rukou vedení firmy, což jim umožní maximální volnost a případné změny. Uživatel typu superuživatel má všechna možná práva. Superuživatelé budou vlastníci společnosti a správci systému. Zároveň má superuživatel pří-

stup do prostředí Django admin, zobrazené na 6.2, které slouží jako systém pro správu obsahu. Jde v něm změnit data na databázové úrovni. Django admin je jednou z výhod frameworku Django, nabízí uživatelsky přívětivé administrační prostředí. Data budou mít možnost změnit i vedoucí firmy bez znalosti SQL, což do této doby nebylo možné.

Objekt `User` (uživatel) zaznamenává

- `email`,
- `first_name` (křestní jméno),
- `last_name` (příjmení),
- `phone_number` (telefonní číslo),
- `branch` (pobočka, na které zaměstnanec pracuje),
- `last_login` (datum a čas posledního přihlášení),
- `date_joined` (datum přidání uživatele),
- `is_active` (indikace toho, zda je uživatel ve společnosti stále zaměstnaný).

Pokud má atribut `is_active` hodnotu nepravda, uživateli není umožněn přístup do systému.

Firma má několik poboček, moduly dovolených a rezervací se používají na všech z nich, modul produkce pouze na některých. Ke které pobočce patří uživatel je zaznamenáno pomocí objektu `Branch` (podnik), ten zaznamenává

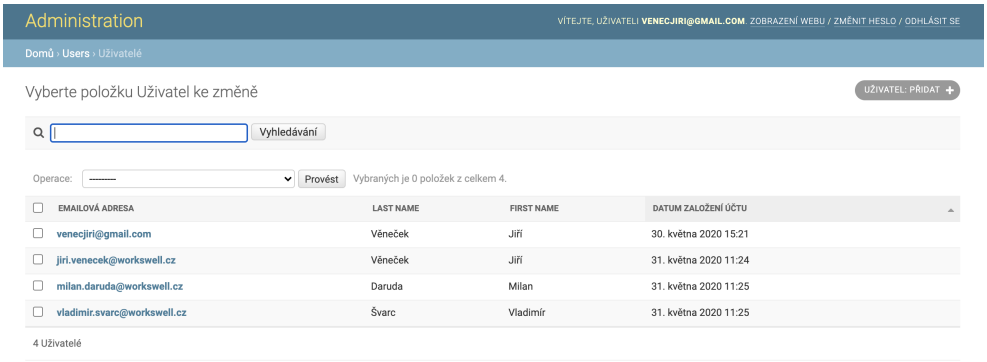
- `name` (název pobočky),
- `address` (adresa pobočky).

Přístup do částí nesouvisejících s pobočkou bude v kompetenci vedení firmy. Tento přístup je možné realizovat například pomocí skupiny práv nazvané podle pobočky.

6.4 Úkoly

Modul slouží k zadávání a přehledu úkolů zaměstnanců společnosti. O uchování informací o úkolech se stará objekt `Task` (úkol). Ten zaznamenává

- `status` (stav úkolu, může být „čeká“, „zadán“, „dokončen“ nebo „zrušen“),
- `priority` (priorita na stupnici od jedné do pěti),
- `name` (název),
- `description` (popis),



Obrázek 6.2: Administrační prostředí pro správu uživatelských účtů v systému.

- `assignee` (komu je přiřazen),
- `assigned_by` (od koho je přiřazen),
- `due_date` (deadline) a
- `assignment_date` (datum zadání).

Úkolu je možné přiřadit také štítky. `Tag` (štítek) je jednoduché označení povahy úkolu, například „spěchá“ nebo „nízká priorita“. Objekt štítku ukládá tedy jeho

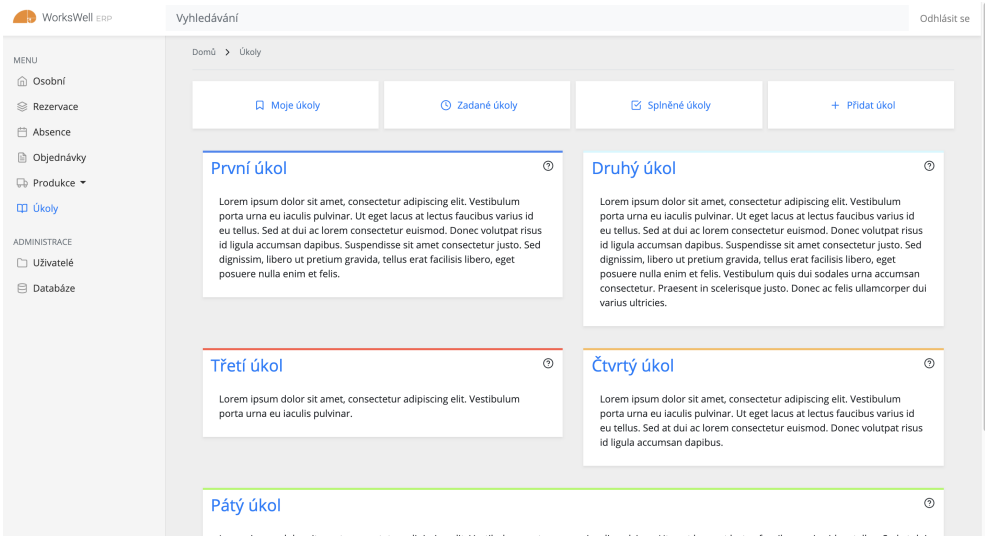
- `text` a
- `color` (barva pro lepší rozlišení).

Přehledu úkolů přihlášeného uživatele ukazuje obrázek 6.3.

O připomínání úkolů se stará objekt `TaskReminder` (připomenutí úkolu). Jeho vytvoření zajistí upozornění zaměstnance plnícího úkol ve zvolený čas.

Oproti běžným funkcím úkolníčku je v systému objekt `TaskResolver` (řešitel úkolu). Ten zadaný úkol označí za vyřešený, pokud nastane určitá akce. Například schválení dovolené nebo uzavření sestavy. Funkcionalita je implementována pomocí Django signálů, to je systém upozornění po uskutečnění nějaké události v systému, například editace dovolené. Systém řešitelů úkolů je rozšiřitelný i pro další případy užití. Vztahy mezi zmíněnými objekty jsou zobrazeny na diagramu 6.5.

Tento modul by mohl nahradit používání ZOHO Task Managementu a posunout společnost směrem k využívání jednotného systému pro všechny její činnosti.



Obrázek 6.3: Přehled přiřazených úkolů zaměstnance.

6.5 Rezervace

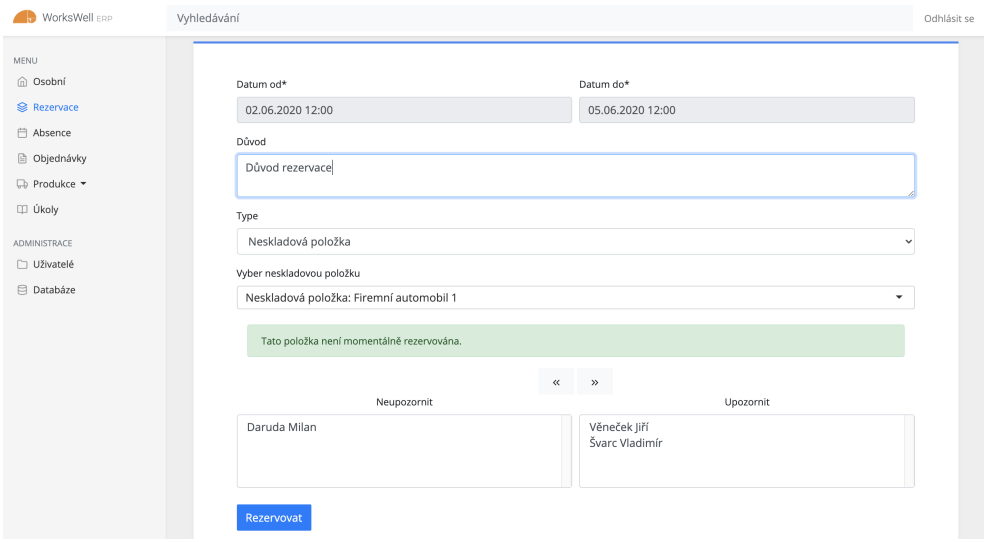
Modul rezervací má za úkol spravovat interní zápůjčky. Zaměstnanci si mohou vypůjčit tři druhy věcí. Produkty, komponenty nebo neskladové položky (např. firemní automobil). Výpůjčku zaznamenají do formuláře zobrazeného na obrázku 6.4. Výpůjčku zaznamenává model `Reservation` (Rezervace). Rezervace obsahuje

- `date_from` (od jakého data rezervace probíhá),
- `date_to` (do jakého data rezervace probíhá),
- `reason` (důvod rezervace),
- `made_by` (kým byla rezervace provedena),
- `reservation_item` nebo `product` nebo `component` (co bylo rezervováno).

Datum od musí být dřívější než datum do, jiná omezení nejsou. Atributy `product` nebo `component` jsou cizí klíče na stejnojmenné modely, `reservation_item` je cizí klíč na model speciální pro neskladové věci. `ReservationItem` model má pouze

- `name` (název rezervované položky).

Pro rezervování položky využije zaměstnanec k tomu určený formulář. Při vybrání požadované položky k rezervaci ho systém upozorní, zda již položka není rezervována někým jiným. Po dohodě s vedením firmy systém umožní



Obrázek 6.4: Formulář pro rezervování položek.

uživateli zarezervovat i položku, která je v systému vedena jako momentálně rezervována. Odpovědnost tedy zůstává na uživateli systému.

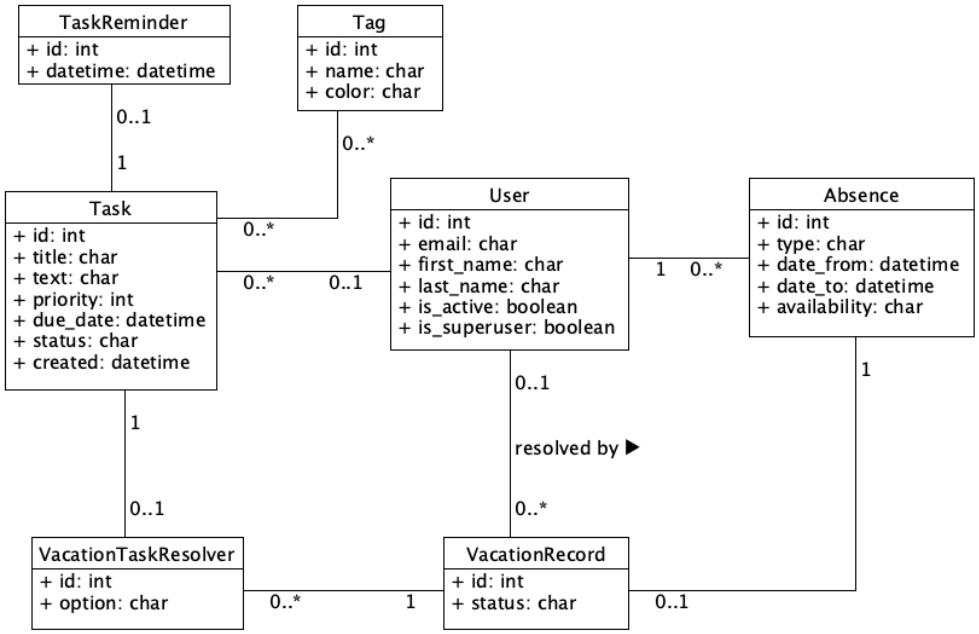
6.6 Absence

Jedním z požadavků na systém bylo zaznamenávání a kategorizace všech absencí zaměstnanců firmy. Existuje několik typů absencí. Jsou to

- dovolená,
- pracovní neschopnost,
- sick days,
- návštěva lékaře a
- služební cesta.

V systému je pro zaznamenání absence použit objekt Absence, který ukládá

- `type` (typ absence),
- `user` (cizí klíč zaměstnance, ke kterému absence patří),
- `date_from` (datum a čas začátku absence),
- `date_to` (datum a čas konce absence),



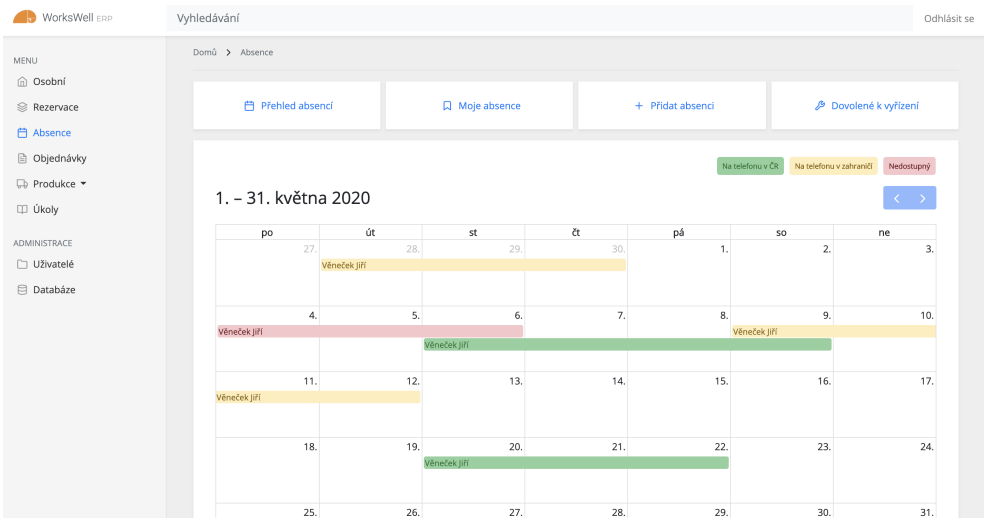
Obrázek 6.5: Doménový diagram zachycující vztahy mezi objekty související s úkoly, rezervacemi a absencemi.

- `availability` (dostupnost během absence - může být buď dostupný na telefonu v České republice, dostupný na telefonu v zahraničí nebo nedostupný).

Při vytvoření objektu `Absence` s typem dovolená je zároveň vytvořen další objekt s názvem `VacationRecord` (záznam dovolené). Vytvořit žádost o dovolenou může každý uživatel s příslušným právem, schválit nebo zamítnout dovolenou potom mohou uživatelé systému se zvláštním oprávněním. Systém rozpozná, pokud uživatel zadá dovolenou v době státního svátku a nebude ji počítat do dnů vyčerpané dovolené. Protože tento typ absence podléhá schválení, zaznamenává navíc

- `absence` (cizí klíč na absenci),
- `resolved_by` (kým byla žádost o dovolenou vyřízena),
- `status` (stav žádosti).

Žádost se po jejím založení nachází ve stavu `requested` (podána). Dále je potřeba žádost zkontrolovat personalistkou ve společnosti, která zjistí, zda má žádající zaměstnanec na dovolenou nárok. Pokud nárok nemá, žádost přechází do stavu `declined` (zamítnuta). Pokud nárok na dovolenou má, přechází do stavu `waiting` (čeká). V této fázi čeká na schválení nebo zamítnutí jedním ze



Obrázek 6.6: Přehled absencí zobrazený v kalendáři aktuálního měsíce.

tří členů vedení firmy. Pokud je schválena, skončí záznam dovolené ve stavu `accepted` (schválena) a uživatel je upozorněn emailem a notifikací ve firemním systému. V opačném případě přechází do stavu `declined` (zamítnuta), na tuto skutečnost je uživatel rovněž upozorněn emailem a notifikací.

Počet dní dovolené je dohodnutý ve smlouvě. Standardně podle zákoníku práce je to 20 dní ročně. Pracovní neschopnost je nepřítomnost zaměstnance na pracovišti na základě vystavení neschopenky lékařem. Pokud zaměstnanec lékaře nenavštíví, má nárok na tzv. „sick days“, při kterých zaměstnanec pobírá plat a které se mu nezapočítávají do vybrané dovolené. Ve firmě tento nárok činí 2 dny. Zaměstnanec má také nárok na návštěvu lékaře, tuto návštěvu mu musí lékař potvrdit na propustce. Absence zaměstnance na pracovišti může vzniknout také při různě dlouhých pracovních cestách, které jsou součástí pracovního procesu, započítávají se do pracovní doby.

Každý zaměstnanec má standardně nárok na jednu stravenku za každý odpracovaný den. Stravenky za uplynulý měsíc obdrží zaměstnanec vždy první pracovní den následujícího měsíce. Zaměstnanec nemá nárok na stravenku v době, kdy čerpá dovolenou, dále v době, kdy je v pracovní neschopnosti, v době, kdy jeho absence z důvodu návštěvy lékaře trvala více než polovinu pracovní doby, za den, kdy jeho pracovní cesta trvala více než polovinu pracovní doby. Naopak zaměstnanec má nárok na stravenku při využití sick days. Všechna tato fakta bylo nutné ve firemním systému zohlednit.

Na obrázku 6.6 je zobrazena stránka s přehledem absencí ve společnosti pro aktuální měsíc.

6.7 Objednávky

Součástí každého výrobního procesu je příprava a zabezpečení dostatečného množství potřebného materiálu. Společnost tedy musí objednat komponenty na sestavení jednotlivých výrobků v dostatečném časovém předstihu a v dostatečném množství. K hladkému průběhu tohoto procesu přispívá způsob evidence a objednávání jednotlivých dílů. Při prvním nákupu (u nového výrobku) nebo při nedostatku komponent na skladě je vytvořena objednávka. Pro opakující se objednávky se používají šablony objednávek. Společné atributy objektu `Order` (objednávka) a objektu `OrderTemplate` (šablona objednávky) jsou uloženy v objektu `BaseOrder` (základ objednávky), který zaznamenává

- `supplier` (cizí klíč na objekt společnosti typu dodavatel),
- `business` address (obchodní adresa),
- `delivery` address (dodací adresa),
- `currency` (měna objednávky),
- `order_confirmation_email` (email, na který je zasláno potvrzení objednávky),
- `pick_up_confirmation_email` (email, na který je zasláno potvrzení o vyzvednutí dodávky),
- `delivery_conditions` (podmínky dodávky),
- `shipment_conditions` (podmínky přepravy),
- `payment_conditions` (podmínky platby),
- `special_delivery_instructions` (zvláštní pokyny k doručení).

Tyto atributy jsou děděny objednávkou a šablonou objednávky. Objednávka má navíc

- `invoice_number` (číslo faktury),
- `watch_order_confirmation` (možnost sledovat potvrzení zásilky),
- `watch_delivery_time` (možnost sledovat čas doručení).

Šablona objednávky má také atribut pro pojmenování

- `name` (pojmenování šablony).

Po vytvoření objednávky je možné přidávat položky objednávky. Lze vybrat pouze ty komponenty, které jsou od daného dodavatele nabízeny. Ty jsou zaznamenávány pomocí objektu `OrderItem` (položka objednávky), který obsahuje

- `order` (cizí klíč na objednávku),
- `component` (cizí klíč na komponentu),
- `count` (požadovaný počet komponent).

Ve firemním systému objekt `Company` (Společnost) identifikuje firmy dodavatelů, výrobců a zákazníků (odběratelů). Jeho údaje jsou importovány z účetního systému S5 Money. Zaznamenává

- `type` (typ společnosti, může být dodavatel, výrobce nebo zákazník),
- `code` (kód společnosti),
- `name` (název společnosti),
- `shortcut` (zkratka názvu společnosti),
- `business_address` (obchodní adresa),
- `billing_address` (fakturační adresa),
- `branch_address` (adresa provozovny),
- `bank` (banka),
- `bank_account_number` (číslo bankovního účtu),
- `bank_country` (země sídla banky),
- `currency` (preferovaná měna),
- `ICO` (IČO),
- `DIC` (DIČ),
- `vat_payer` (indikace, zda je plátce DPH),
- `natural_person` (fyzická osoba),
- `email`,
- `phone_number` (telefonní číslo).

6.8 Komponenty

Ve stávajícím systému existovaly objekty `Stock` (zboží), `Set` (sestava) a `product` (produkt). Objekt `Product` se ukázal po analýze být zbytečný, v novém systému se proto vůbec používat nebude. `Stock` je přejmenováno na `Component` (komponenta) a `Set` je nyní `Product` (výrobek). Ve firemní praxi se pojmy produkt a výrobek často ztotožňují.

Výrobky společnosti jsou skládány z různých komponent. Příkladem komponent jsou SSD disk, USB kabel, teploměr nebo iPad. V systému jsou pro zaznamenání komponent použity dva objekty, jednak „Model komponenty“, dále pak „Komponenta“. První objekt představuje předpis komponenty, druhý potom značí konkrétní kus na skladě. Například SSD disk Samsung 120GB je tzv. „Model komponenty“ a „SSD disk Samsung 120 GB“ s konkrétním výrobním sériovým číslem „ABC-123“ je konkrétní komponenta nazvaná „Komponenta“.

`ComponentModel` (model komponenty) obsahuje

- `manufacturer` (cizí klíč výrobce),
- `supplier` (cizí klíč dodavatele),
- `name` (jméno),
- `areas` (oblasti, do kterých model komponenty spadá).

Pro lepší kategorizaci lze modely komponent řadit do oblastí. `Area` (oblast) je jednoduchý objekt zaznamenávající

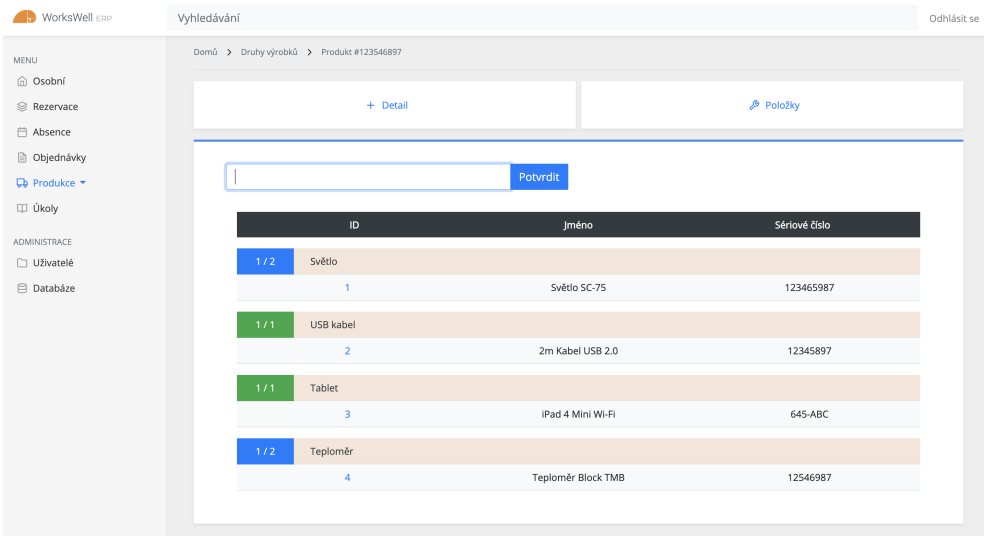
- `name` (název oblasti).

Příkladem oblasti jsou kamery, software, termokamery, osvětlení, kryty pro kamery, kalibrační zdroje. `Component` (komponenta) je objekt představující instanci předpisu komponenty, který zaznamenává

- `component_model` (cizí klíč na Model komponenty),
- `delivery_list` (cizí klíč na dodací list),
- `warranty_until` (v záruce do) a
- `serial_number` (sériové číslo).

V případě, že objednávka nepřijde kompletní, zaměstnanec zaznamená a potvrdí pouze komponenty, které byly dodány. Seznam takových komponent je nazýván dodací list. V systému je to objekt s názvem `DeliveryList` (dodací list), který zaznamenává

- `name` (automaticky přiřazen podle objednávky a pořadí, ve kterém přišly na něm označené komponenty).



Obrázek 6.7: Stránka pro načítání komponent výrobku ručně nebo pomocí 2D čtečky.

- `order` (cizí klíč na objednávku) a
- `delivery_date` (datum dodání).

Pro každý dodací list lze vygenerovat štítky pro fyzické označení dodaných komponent.

6.9 Výrobky

Výrobky představují hlavní produkty a zdroj příjmů společnosti. Stejně jako u komponenty je v systému potřeba zaznamenat dva druhy výrobku. Prvním je tzv. „Model produktu“ představující předpis produktu a druhým tzv. „Produkt“, který reprezentuje konkrétní kus na skladě. `ProductModel` (model produktu) je objekt zaznamenávající

- `manufacturer` (cizí klíč výrobce),
- `buyer` (cizí klíč odběratele),
- `name` (jméno),
- `template` (šablona, podle které byl výrobek sestaven, pokud existuje),
- `material_number` (číslo materiálu) a
- `description` (popis).

`Product` (produkt) je potom objekt, který má

- `product_model` (cizí klíč Modelu produktu),
- `stock_number` (skladovací číslo),
- `serial_number` (sériové číslo),
- `intended_for` (určeno pro) a
- `note` (poznámka).

Objekt produktu se vytvoří na začátku výrobního procesu. V této fázi se stanoví cílová podoba výrobku. Při jeho sestavování jsou k němu postupně přiřazovány jednotlivé komponenty. Jako indikátor, zda je produkt připravený k prodeji, slouží objekt `ProductClosure` (uzávěrka produktu), který zaznamenává

- `product` (cizí klíč produktu),
- `date` (datum provedení výstupní kontroly),
- `is_closed` (stav výstupní kontroly) a
- `closed_by` (kým byla uzávěrka vytvořena).

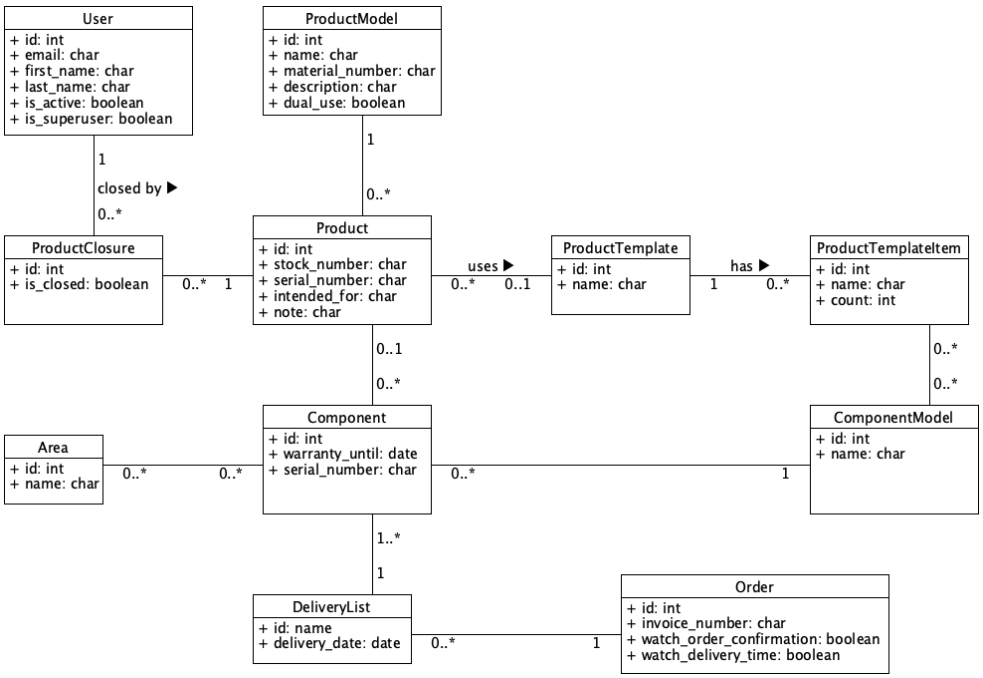
Protože je proces sestavování různých produktů repetitivní, pro snazší a rychlejší průběh jsou používány šablony produktů. Šablona má název a položky šablony. Položka šablony poté představuje počet a typ komponent, které je potřeba přidat. Můžeme mít například šablonu s názvem „Termokamera Medicas“, která má různé položky, jednou z nich je „USB kabel“ s počtem 1 a možnostmi „OEM USB 2.0“ a „PremiumCord USB 2.0“. To znamená, že při sestavování výrobku může být použit jeden z těchto dvou kabelů. `ProductTemplate` (šablona produktu) je v systému objekt zaznamenávající

- `name` (název šablony).

`ProductTemplateItem` (položka šablony produktu) je potom objekt s

- `template` (cizí klíč šablony),
- `name` (jméno),
- `count` (počet) a
- `options` (konkrétní nabídka modelů komponent, které lze použít).

Obrázek 6.8 zobrazuje vztahy mezi objekty ve výrobním procesu.



Obrázek 6.8: Doménový diagram zachycující vztahy mezi objekty související s výrobním procesem.

Kapitola 7

Možnosti rozšíření systému

Jedním z cílů bakalářské práce bylo vytvořit snadno rozšiřitelný systém. V této kapitole jsou popsány možnosti jeho rozšíření a dalších úprav.

Díky spuštění aplikace na webovém serveru je možné instalovat rozšiřující moduly nabízené ostatními systémy nebo vývojáři. Příkladem takového rozšíření je využití spolupráce se Zoho Projects. Pro jazyk Python nabízí Zoho svůj SDK (software development kit). Propojení se Zoho je využitelné v řadě případů. Jedním z nich může být modul přehledů, díky kterému firemní vedení může získávat různá statistická data ve formě tabulek, grafů a diagramů. Například kolik bylo sestavených produktů, jaká byla nemocnost v daném měsíci, jak dlouhá je průměrná doba dodání komponent apod. Zpracování statistik by umožnilo měřit firemní procesy a vliv zavádění změn na produktivitu práce. U nejnověji přidané funkce „sestavování za pomoci skenování QR kódů“ v současnosti neexistuje způsob změření její efektivity, po přidání modulu přehledů by bylo možné změřit úsporu času při použití 2D čtečky QR kódů.

Další možností rozšíření je přidávat úkoly do Zoho Projects podle stavu skladu. Pokud nastane moment, kdy počet komponent na skladě bude menší než určitá hodnota, je odpovědné osobě vytvořen úkol objednat tyto komponenty a doplnit tak jejich skladové zásoby.

Pro lepší přehled časové náročnosti jednotlivých úkonů, které zaměstnanci firmy provádějí, by mohl být systém rozšířen o modul, který bude měřit dobu, za kterou zaměstnanci daný úkol splnili. To pomůže vedení lépe plánovat práci a lépe odhadnout délku výrobního procesu například u nového produktu nebo je může vedení firmy využít jako podklad pro hodnocení zaměstnanců. Pro přehlednost bude možné data zobrazovat v podobě grafů a tabulek.

Závěr

Bakalářská práce si kladla za cíl analyzovat informační systém společnosti Workswell. Po konzultaci s firemními zaměstnanci, seznámení s organizační strukturou, činností a potřebami firmy byl prozkoumán zdrojový kód a funkcionalita systému. Byly odhaleny zastaralé postupy, bezpečnostní rizika a systémové chyby, které snižují produktivitu práce ve firmě.

Nedostatky a nefunkcionality byly konzultovány s vedením firmy, detailně a na konkrétních případech bylo doloženo, proč je důležitá modernizace systému, proč nestačí pouze dílčí úpravy. Bylo vysvětleno, jakým způsobem se zvýší efektivita výrobního procesu firmy, jakým způsobem bude vylepšena komunikace mezi jednotlivými zaměstnanci firmy a firemními pobočkami. Firemní systém byl implementován za použití moderního frameworku Django a dalších frontendových technologií jako je HTML, CSS, Bootstrap, JQuery a Javascript. S novými funkcionalitami byli seznámeni zaměstnanci, kteří budou systém využívat.

Již od počátku bylo stěžejním cílem vytvořit takový systém, který by byl snadno rozšiřitelný, intuitivní pro nové zaměstnance a dokázal držet krok s vývojem firmy a novými technologiemi. Tohoto cíle se podařilo dosáhnout a v budoucnu nebude problém přidat potřebné vylepšení systému. Práce zároveň slouží jako dokumentace systému pro nové zaměstnance nebo jako východisko pro další studentskou práci nebo stáž. Jak již bylo zmíněno, společnost se zabývá výrobou termokamer a v současné situaci pandemie nebyl vývoj systému prioritou, proto se nenachází ve finální fázi zpracování, ale bude ještě dále upravován, testován a vylepšován.

Literatura

[1] *O nás* [online]. Workswell [cit. 2020-06-03]. Dostupné z: https://workswell.cz/o-spolecnosti/

[2] SHANG, Shari a Peter B. SEDDON. A Comprehensive Framework for Classifying the Benefits of ERP Systems [online]. AMCIS 2000 Proceedings, 2000, , 1006 [cit. 2020-06-03]. Dostupné z: https://aisel.aisnet.org/cgi/viewcontent.cgi?article=1452&context=amcis2000

[3] ČÁPKA, David. *Lekce 3 - Představení MVC a MVT architektury v Django* [online]. itnetwork.cz [cit. 2020-06-03]. Dostupné z: https://www.itnetwork.cz/python/django/predstaveni-mvc-a-mvt-architektury-v-django

[4] FOWLER, Martin. Mapping to Relational Databases. *Patterns of enterprise application architecture*. Boston: Addison-Wesley, 2003, s. 33-53. ISBN 0-321-12742-0.

[5] *Models* [online]. Django Software Foundation [cit. 2020-06-03]. Dostupné z: https://docs.djangoproject.com/en/3.0/topics/db/models/

[6] *Writing views* [online]. Django Software Foundation [cit. 2020-06-03]. Dostupné z: https://docs.djangoproject.com/en/3.0/topics/http/views/

[7] *Migrations* [online]. Django Software Foundation [cit. 2020-06-03]. Dostupné z: https://docs.djangoproject.com/en/3.0/topics/migrations/

[8] MASSE, Mark. *REST API Design Rulebook: Designing Consistent RESTful Web Service Interfaces*. Sebastopol: O'Reilly Media, 2011, s. 5. ISBN 978-1449310509.

[9] *Slovníček pojmů*: *AJAX* [online]. Nette Foundation [cit. 2020-06-03]. Dostupné z: https://doc.nette.org/cs/3.0/glossary

[10] *Serializers* [online]. [cit. 2020-06-03]. Dostupné z: https://www.django-rest-framework.org/api-guide/serializers/

[11] *Security in Django* [online]. Django Software Foundation [cit. 2020-06-03]. Dostupné z: https://docs.djangoproject.com/en/3.0/topics/security/

[12] *Slovníček pojmů*: *Escapování* [online]. Nette Foundation [cit. 2020-06-03]. Dostupné z: https://doc.nette.org/cs/3.0/glossary

[13] *Writing your first Django app, part 1*: *Creating a project* [online]. Django Software Foundation [cit. 2020-06-03]. Dostupné z: https://docs.djangoproject.com/en/3.0/intro/tutorial01/

[14] *Data Structures*: *Dictionaries* [online]. Python Software Foundation [cit. 2020-06-03]. Dostupné z: https://docs.python.org/3/tutorial/datastructures.html

[15] *Middleware* [online]. Django Software Foundation [cit. 2020-06-03]. Dostupné z: https://docs.djangoproject.com/en/3.0/ref/middleware/

Příloha **A**

Seznam použitých zkratek

ERP Enterprise Resource Planning

API Application Programming Interface

IT Informační technologie

HTML Hypertext Markup Language

CSS Cascading Style Sheets

ORM Objektově relační mapování

HTTP Hypertext Transfer Protocol

XML Extensible Markup Language

AJAX Asynchronous JavaScript and XML

JSON JavaScript Object Notation

XSS Cross–site scripting

CSRF Cross site request forgery

SQL Structured Query Language

REST Representational State Transfer

WBS Work Breakdown Structure

URL Uniform Resource Locator

Příloha B

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF