|  |
| --- |
| OSTRAVSKÁ UNIVERZITA  PŘÍRODOVĚDECKÁ FAKULTA  KATEDRA INFORMATIKY A POČÍTAČŮ |
| Návrh a implementace skladového systému na platformě .NET s využitím Blazoru  BAKALÁŘSKÁ PRÁCE |
| Autor práce: Tomáš  Vedoucí práce: Mgr. Robert Jarušek, Ph.D. |
| 2023 |

|  |
| --- |
| UNIVERSITY OF OSTRAVA  FACULTY OF SCIENCE  DEPARTMENT OF INFORMATICS AND COMPUTERS |
| Design and implementation of warehouse system on the .NET platform using Blazor  BACHELOR THESIS |
| Author:  Tomáš Kocůrek  Supervisor:  Mgr. Robert Jarušek, Ph.D. |
| 2023 |

(Zadání vysokoškolské kvalifikační práce)

ABSTRAKT

Český text abstraktu

*Klíčová slova:*

*(klíčová slova vypsaná na řádku, oddělená od sebe čárkami)*

**ABSTRACT**

The text of the abstract.

*Keywords:*

čestné prohlášení

Já, níže podepsaný/á student/ka, tímto čestně prohlašuji, že text mnou odevzdané závěrečné práce v písemné podobě je totožný s textem závěrečné práce vloženým v databázi DIPL2.

Ostrava dne

………………………………

podpis studenta/ky

|  |
| --- |
| Poděkování |
| Prohlašuji, že předložená práce je mým původním autorským dílem, které jsem vypracoval/a samostatně. Veškerou literaturu a další zdroje, z nichž jsem při zpracování čerpal/a, v práci řádně cituji a jsou uvedeny v seznamu použité literatury.  V Ostravě dne . . . . . . . . . . . .  . . . . . . . . . . . . . . . . . .  (podpis) |

OBSAH

[ÚVOD 3](#_Toc440839236)

[1 NADPIS 3](#_Toc440839237)

[1.1 Podnadpis 3](#_Toc440839238)

[1.2 Podnadpis 3](#_Toc440839239)

[1.2.1 Podpodnadpis 3](#_Toc440839240)

[2 NADPIS 3](#_Toc440839241)

[2.1 Podnadpis 3](#_Toc440839242)

[2.1.1 Podpodnadpis 3](#_Toc440839243)

[3 NADPIS 3](#_Toc440839244)

[3.1 Podnadpis 3](#_Toc440839245)

[3.1.1 Podpodnadpis 3](#_Toc440839246)

[4 NADPIS 3](#_Toc440839247)

[4.1 Podnadpis 3](#_Toc440839248)

[4.1.1 Podpodnadpis 3](#_Toc440839249)

[ZÁVĚR 3](#_Toc440839250)

[RESUMÉ 3](#_Toc440839251)

[SUMMARY 3](#_Toc440839252)

[SEZNAM POUŽITÉ LITERATURY 3](#_Toc440839253)

[SEZNAM POUŽITÝCH SYMBOLŮ 3](#_Toc440839254)

[SEZNAM OBRÁZKŮ 3](#_Toc440839255)

[SEZNAM TABULEK 3](#_Toc440839256)

[SEZNAM PŘÍLOH 3](#_Toc440839257)

ÚVOD

Text

1. Rešerše

Skladový systém je druh informačního systému, jehož cílem je usnadnění práce se skladovou zásobou typicky ve firemním prostředí. Na trhu se v dnešní době nachází mnoho různých komerčních řešeních, které poskytují různé funkcionality.

Touto problematikou se již zabýval ve své bakalářské práci Toman Marek [1]. Ten ve své práci vytvářel skladový systém pro obchod s oblečením. K řešení využil metodu RUP (Rational Unified Process). Autor v závěru své popisuje přínosy této metodiky, konkrétně jmenuje například výhodu v podobě odhalení rizik již na začátku vývoje a možnost dodávat zákazníkovi malé části aplikace již v průběhu vývoje. Pro samotnou implementaci poté zvolil Javu a 2 vrstvou architekturu.

Jana Svobodová [2] se v části své práce zaměřila na možnosti automatické evidence zboží na skladě. V ní porovná různé metody jako například. čtečky čárových kódů, magnetické proužky a další. U jednotlivých metod porovnává primárně jejich ekonomickou nákladnost, ale také zda jsou vhodné pro její doménu problému.

Jiří Altschmied [3] se věnuje implementaci již existujícího komerčního skladového systému do expedičního skladu. Popisuje důvody, proč společnost chce zavést skladový systém a následně jednotlivé fáze implementace systému SAP.

* 1. Komerční řešení

Na trhu se také nachází velké množství komerčního softwaru určeného ke správě skladu. Tyto systémy jsou často přímo spojeny s dalšími produkty daného výrobce. Například skladový systém od společnosti Dotykačka [4] je součástí balíčku softwaru pro pokladny. Skladový systém v tomto případě nabízí integraci s mnoha dalšími systémy (platby, tržby, prodej atd.), ale nelze používat samostatně což může být značná nevýhoda. Díky těmto integracím je systém využitelný pouze pro gastronomické podniky (restaurace, rychlé občerstvení). Dalším příkladem je ERP Karat [5]. Jedná se o kompletní Enterprise resource planning software, jehož součástí je také skladový systém. V tomto případě se jedná o obecný systém, který je využitelný pro jakýkoliv druh podniku. Na rozdíl od předchozího softwaru je Karat starší, což se projevuje například na uživatelském rozhraní. Součástí Karatu je také řízený sklad, který automatizuje část skladových procesů (automatické objednávání, výběr skladových pozic atd.). Kromě těchto obecných systémů, existují také systémy, které jsou navrhovány speciálně pro určitou firmu a její specifické potřeby. Příkladem takového systému je WMS navržený pro společnost Polabské mlékárny, který řeší problémy týkající se konkrétně skladování a expedice mléčných výrobků. Systém opět obsahuje modul řízeného skladu, který například radí, jak skládat palety s produkty pro jednotlivé objednávky [6].

* 1. Shrnutí

I přes velké množství komerčních řešení se dá zjistit velice málo o vnitřním fungování samotných skladových systémů. Oproti prvním popisované práci budu svůj systém vyvíjet na platformě .NET a využiji n-vrstvé architektury. Dále se také budu zaměřovat čistě na implementaci skladového systému s prvky řízeného skladu a nebudu se zabývat jeho integrací na další vnitropodnikové systémy.

1. Cíle práce
   1. Hlavní cíl

Hlavním cílem práce je vytvoření funkčního skladového systému s prvky řízeného skladu. Implementace bude provedena na platformě .NET v jazyce C#.

* 1. Dílčí cíle

Dílčími cíli práce jsou provést analýzu již existujícího skladového systému, zanalyzovat a porovnat možná technologická řešení (platforma, architektura) a zvolit to nejoptimálnější, a dále popsat vývojový cyklus softwaru pomocí agile metody Scrum.

1. Analýza problému
   1. Požadavky

Skladový systém by měl být navržen pro obecný sklad tzn. měl by být použitelný v různých odvětvích, aniž by bylo potřeba zásadních úprav samotného systému. Systém musí být schopný zobrazovat aktuální zásobu produktů, jejich umístění na skladě a také umožňovat administraci jednotlivých položek v katalogu.

Systém by také měl obsahovat prvky řízeného skladu. Systém bude doporučovat nejvhodnější místa ve skladu, kde uložit přijaté zboží. Systém bude také asistovat při nakládání objednávek na export tzn. určuje optimální pořadí nakládání předmětů. Dále také sleduje skladovou zásobu a upozorňuje pokud klesne pod danou hranici.

1. Technologické možnosti implementace

Důležitým krokem při vývoji softwaru je výběr architektury a technologií, které budou použity pro následnou implementaci. Je nutné zvážit výhody a nevýhody různých řešení a poté zvolit to nejvhodnější pro naši doménu. Výběr nesprávného Tech Stacku (soubor technologií, které slouží pro vývoj a běh softwaru) může vést k velkým problémů během vývoje systému.

* 1. Architektura

Softwarová architektura určuje strukturu aplikace. Většinou dělí celý systém na jednotlivé části a říká, jak spolu můžou komunikovat a spolupracovat. Základní architektura většinou neurčuje samotné technologie, které budou použity například nedefinuje, pokud mám použít Oracle nebo MySQL databázi, pouze říká že v systému bude využita relační databáze.

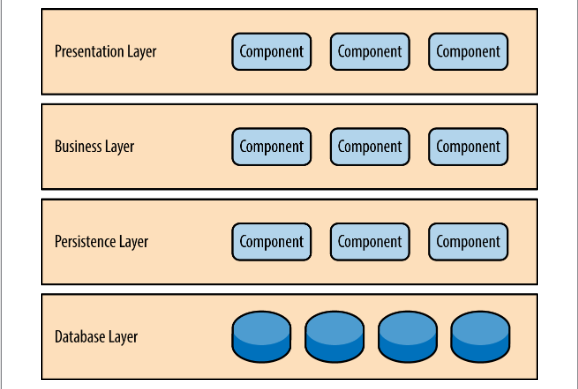
Mnoho vývojářů začne programovat aplikaci, aniž by konkrétně definovali její architekturu. Tento přístup často vede ke shluku neorganizovaného kódu. Takové to aplikace jsou velmi náročné na údržbu, těžko se mění a většinou jsou velmi obtížně škálovatelné.

Existuje mnoho různý vzorů pro architektury, tyto vzory se často kombinují a vytváří nové architektury. Při výběru správné architektury je nutné se zvážit mnoho elementů. Například při vývoji menší aplikace, kterou vyvíjí pouze pár jedinců není potřeba využít složitější architektury, která je zaměřena na výkon a škálovatelnost, ale je obtížná na vývoj a testování.

* + 1. Vrstvená architektura

Jednou z nejpoužívanějších architekturou je právě vrstvená architektura. Program je rozdělen na jednotlivé vrstvy, které spolu komunikují. Každá vrstva má svůj vlastní účel a ostatní vrstvy by na ni neměli být závislé. Díky tomuto dělení je možné jednu vrstvu upravit dokonce i naprosto nahradit (například jiná technologie), aniž by to ovlivnilo fungování ostatních vrstev.

Za normálních okolností by vrstvy měli komunikovat pouze s vrstvami, které s ní přímo sousedí. Vrstva ovšem může být označená jako otevřená což znamená, že při komunikaci může být přeskočena.

Počet vrstev není pevně stanovat. Nejčastěji se objevují 4 vrstvy, ale podle potřeby je možné přidávat další vrstvy, které mají svůj specifický účel.

Nevýhoda této architektury spočívá ve výkonu a škálovatelnosti. Když se na prezentační vrstvě vytvoří požadavek musí projít všemi vrstvami, aby mohl být zpracován a mohl být zobrazen výsledek. Takovéto řešení požadavků může být zdlouhavé zejména v případě, že některé vrstvy s požadavkem vůbec nepracují a pouze ho předávají dále. Z hlediska škálovatelnosti nastává problém u databázové vrstvy. Všechny ostatní vrstvy lze do určité míry škálovat, při velkém množství požadavků se ale databáze stává úzkým hrdlem.

* + 1. Event-driven architektura
  1. Platforma
  2. Programovací jazyk (backend)

Po výběru vhodné architektury je nutné zvolit programovací jazyk, který bude systém pohánět. Hlavním kritériem je cílová platforma, na které bude systém běžet. Tento požadavek může značně omezovat spektrum jazyků, mezi kterými můžeme volit. Při tomto výběru může hrát velkou roli také osobní preference vývojářské týmů a jejich zkušeností s jednotlivými programovacími jazyk.

* + 1. C# (.NET)

Moderní objektově orientovaný jazyk vyvíjený společnostní Microsoft. C# funguje na platformě .NET, původně .NET Framework, který byl určen pouze pro systémy Windows a byl close-source. Microsoft později vytvořil .NET Core (dnes již pouze .NET), což je open-source platforma, která umožňuje vývoj na většinu dnešních systémů a zařízení. C# si bere z hlediska syntaxe inspiraci z jazyků C, C++ a také Java, k poslednímu zmiňovanému je často přirovnáván, ovšem vydává se svojí cestou a často implementuje nové funkce rychleji než jeho předchůdce.

* + 1. Java

Objektově orientovaný jazyk původně vytvořen společnostní Sun Microsystems dnes již pod správou firmy Oracle. Java umožňuje přístup *write once, run anywhere*, toto je umožněno díky tomu, že zdrojový kód je zkompilovaný do bytecodu a ten je poté spuštěn na JVM (Java Virtual Machine). JVM je poté schopno kód spustit nehledě na platformu. Díky tomu se Java využívá hojně u serverový a mobilních aplikací.

* 1. Programovací jazyk (frontend)
  2. Databáze
  3. Nástroje pro vývoj

ZÁVĚR

RESUMÉ

SUMMARY

SEZNAM POUŽITÉ LITERATURY

|  |  |
| --- | --- |
| [1] | TOMAN, Marek. *Návrh a implementace informačního systému pro řízení prodeje a skladu prodejny*. Ostrava, 2009. Bakalářská práce. Ostravská univerzita. Vedoucí práce Doc. Ing. František Huňka, CSc. |
| [2] | SVOBODOVÁ, Jana. *Skladovací systém*. Praha, 2017. Diplomová práce. České vysoké učení technické v Praze. Vedoucí práce Prof. akad. arch. Jan Fišer. |
| [3] | ALTSCHMIED, Jiří. *Implementace systému skladového hospodářství ve výrobní společnosti*. Liberec, 2017. Bakalářská práce. Technická univerzita v Liberci. |
| [4] | *Dotykačka* [online]. Dotykačka s. r. o., 2021 [cit. 2022-12-14]. Dostupné z: https://dotykacka.cz |
| [5] | *Skladový systém Karat* [online]. KARAT Software a. s., 2022 [cit. 2022-12-14]. Dostupné z: https://www.karatsoftware.cz/erp-karat/skladovy-system |
| [6] | *WMS Polabské mlékárny* [online]. Ostrava: Itixo s.r.o, 2022 [cit. 2022-12-15]. Dostupné z: https://itixo.com/cs/projects/ |

SEZNAM POUŽITÝCH SYMBOLŮ

|  |  |  |
| --- | --- | --- |
| ABC |  | Význam první zkratky. |
| B |  | Význam druhé zkratky. |
| C  ERP |  | Význam třetí zkratky. |
| WMS |  |  |
|  |  |  |

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH