Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačního inženýrství

Diplomová práce

Návrh informačního systému v UML

Bc. Evgeniya Gorina

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ DIPLOMOVÉ PRÁCE

Bc. Evgeniya Gorina

Systémové inženýrství a informatika
Informatika

Název práce

Návrh informačního systému v UML

Název anglicky

Design of Information System in UML

Cíle práce

Cílem diplomové práce je návrh informačního systému „Your takeaway“, který se zaměřuje na online správu a úpravu objednávek. Navržený systém zaznamenává objednávky zákazníka ku obchodníkovi. Z toho důvodu bude obsažen proces registrace jak obchodu, tak účtu zákazníka. Dále systém obsahuje správu a úpravu objednávek. Základní funkcí systému je online objednávání kávy a zákusků od různorodých kavárenských podniků.

Metodika

Metodika diplomové práce je zaměřena na studium odborných literárních zdrojů v souladu s požadavky, kladené na systém. Bude provedena analýza požadavků. Návrh pro informační systém bude realizován v jazyce UML. Rešerše metod určených k realizaci systému budou doplněny vlastním návrhem designu. Pro implementace navrženého systému bude vytvořen design uživatelského rozhraní.

Doporučený rozsah práce

60 – 80

Klíčová slova

UML, návrh, informační systém, analýza, design, webová aplikace, prototyp

---

Doporučené zdroje informací

BASL, Josef a Roman BLAŽÍČEK. Podnikové informační systémy: Podnik v informační společnosti. Praha: Grada, 2012. ISBN 978-80-247-4307-3.

PROCHÁZKA, JAROSLAV a CYRIL KLIMEŠ. SOFTWAROVÉ INŽENÝRSTVÍ. 2., aktualiz. a dopl. vyd. Ostrava: Ostravská univerzita v Ostravě, 2009.

ŘEPA, Václav. Alayza a návrch informačních systémů. Praha : Ekopress, 1999. ISBN 80-86119-13-0.

Václav. Procesně řízená organizace. Praha: Grada, 2012. Management v informační společnosti. ISBN 9788024741284.

VRANA, I. – ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE. PROVOZNĚ EKONOMICKÁ FAKULTA, KATEDRA INFORMAČNÍHO INŽENÝRSTVÍ. Projektování informačních systémů s UML. V Praze: Česká zemědělská univerzita, Provozně ekonomická fakulta, 2008. ISBN 978-80-213-1817-5.

Předběžný termín obhajoby

2021/22 LS – PEF

Vedoucí práce

Ing. Jakub Konopásek, Ph.D.

Garantující pracoviště

Katedra informačního inženýrství

Elektronicky schváleno dne 31. 10. 2022

Ing. Martin Pelikán, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 28. 11. 2022

doc. Ing. Tomáš Šubrt, Ph.D.

Děkan

V Praze dne 28. 03. 2023

Čestné prohlášení

Prohlašuji, že svou diplomovou práci „Návrh informačního systému v UML“ jsem vypracovala samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autorka uvedené diplomové práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 31.03.2023 _____

Poděkování

Ráda bych touto cestou poděkovala Ing. Jakubovi Konopáskovi, Ph.D. za metodické vedení diplomové práce, cenné rady a odborný dohled. Srdečně jemu děkují za poskytnuté konzultace, návrhy a výstižná doporučení.

Návrh informačního systému v UML

Abstrakt

Diplomová práce se zabývá návrhem informačního systému zaměřeného na vývoj aplikace určené pro vytváření a zpracování objednávek. Návrh je realizován pomocí modelovacího jazyka UML. Teoretická část diplomové práce spojuje poznatky o informačních systémech, jejich moderních metodikách vývoje, prototypování, problematice jazyka UML a systémovém modelování.

Praktická část diplomové práce znázorňuje návrh a principy fungování informačního systému. Jednotné modely jsou vytvořeny v jazyce UML na základě vypracovaných požadavků. Dále je návrh realizován prostřednictvím grafického prototypu na základě vlastního designu navrženého s ohledem na přívětivé uživatelské rozhraní, které poskytuje kvalitní uživatelský zážitek ze systému.

Výsledkem diplomové práce je prototyp elementární verze informačního systému, který může posloužit jako základ při realizaci start-upu.

Klíčová slova: informační systém, design, analýza, návrh, webová aplikace, UML, prototyp

Design of information System in UML

Abstract

This diploma thesis deals with the design of an information system focused on the development of an application intended for the creation and processing of orders. The proposal is implemented using the UML modeling language. The theoretical part of the thesis combines knowledge about information systems, their modern development methodologies, prototyping, UML language issues and system modeling.

The practical part of the diploma thesis shows the design and principles of the functioning of the information system. Unified models are created in the UML language on the basis of developed requirements. Furthermore, the proposal is implemented through a graphic prototype based on a custom design designed with a friendly user interface that provides a quality user experience from the system.

The result of the thesis is a prototype of an elementary version of the information system, which can serve as a basis for the implementation of a start-up.

Keywords: information system, design, analysis, design, web application, UML, prototype

Obsah

Seznam obrázků

Seznam tabulek

Seznam použitých zkratek

IS Informační systém
OOP Objektově orientované programování
BPM Workflow Business Process Management
CMS Content management system
CPM Corporate Performance Management
CRM Customer relationship management
DFD Data flow diagram
DSS Decision support system
EAM Enterprise Asset Management
ECM Enterprise Content Management
EDMS Electronic Document Management
EDP Electronic data processing
ERD Entity Relationship Diagram
ERP Enterprise Resource Planning
HRM Human Resource Management
MIS Management information system
MRP Material requirement planning
SADT Structured Analysis & Design Technique
TDD Test-Driven Development
UI User Interface design
UX User Experience design

1 Úvod

Informační systémy jsou nedílnou součástí moderního světa, jejich popularita roste, stejně tak i jejich nezaměnitelnost, která je podpořená neustálým rozvojem informačních a internetových technologií. Tyto systémy jsou cíleny na usnadnění lidských životů, zjednodušení pracovních činností, úspory času a zefektivnění kvality poskytovaných služeb a produktů.

Různorodé podniky je integrují v rámci odlišných sektorů a odvětví s ohledem na trendy současnosti a zaměřením na konkurenceschopnost, což vyvolává potřebu strukturovaného a plnohodnotného řešení informačních systémů. Východiskem podobných návrhů je snadný a rychlý vývoj softwaru nebo úpravy stávajících řešení, které umožňují zjednodušení podnikových procesů, rozšíření škály zákazníků či snížení nákladů na výrobu. Výjimkou není vývoj webových či mobilních aplikací, v rámci, kterého jsou informační systémy podkladem k jejich návrhu a realizaci. Tyto aplikace jsou často využívané malými a středními podniky pro interakce se zákazníky, za účelem prodeje rozmanitého zboží a služeb. Zájmena webové služby dnešní doby disponují rozsáhlou nabídkou online obchodů, v rámci, kterých nabízí zákazníkům nejen nákup online, ale i donáškové služby nebo možnosti objednaní potřebných produktů či služeb na určitý čas s možností osobního vyzvednutí. Podobné služby často nabízí malé podniky či start-upy v rámci vlastních webových stránek či elektronických obchodů. Nákup podobných IT řešení, jejich vývoj a dále údržba na internetu vyžaduje vynaložení nemalých finančních prostředků a přidává tedy další nákladové položky, kterými začínající podnikatelé ve většině případů nedisponují. Tato problematika se týká nejaktuálnějších webových technologií v současnosti a internetových služeb, v rámci, kterých jsou společnosti za cílem většího a stabilnějšího postavení na trhu nuceny k využití podobných služeb, které nabízí možnosti pronájmu elektronických obchodů, webových stránek či aplikací za poplatky. V tomto směru je snaha o vytvoření návrhu IS, který by umožnil malým a středním podnikům a start-upům provádět jejich činnost snadněji a provázat jejich možnosti s ohledem na menší riziko bankrotu, za poskytnutí příležitosti prostřednictvím využívání aplikací k prodeji jejich vlastních nápojů a zákusků, s možností přilákání většího množství zákazníků. Pro konečné uživatele by aplikace mohla být jednodušším řešením a nabízela by se možnost osobního odběru a pestré palety výběru občerstvení za menší ceny.

2 Cíl práce a metodika

2.1 Cíl práce

Tato diplomová práce si klade za cíl návrh informačního systému potřebného k řešení problematiky vývoje mobilní aplikace, která se zabývá prodejem rozmanitých nápojů a občerstvení v rámci vztahu dodavatel-odběratel. Informační systém umožní zefektivnění a optimalizace činností a procesů probíhajících v rámci vztahu odběratel-dodavatel, a zároveň umožní zjednodušení vývoje aplikace. Informační systém je zaměřen na malé podniky a vytvořen v jazyce UML a je modelován na základě vlastních zkušeností autora a zaměřen na správu objednávek. Dílčím cílem práce je analýza potřebných požadavků kladených na systém. Navržený prototyp je zaměřen na uspokojení potřeb uživatelů systému jak zákazníků, tak podnikatelů.

2.2 Metodika

Metodika diplomové práce vychází ze stanoveného cíle a analýz potřebných k jeho dosažení. Východiskem ke stanovení hranic navrhovaného systému bude využita metoda *„MoSCoW“*. Požadavky určené k fungování systému budou stanoveny podle rozdělení na funkční a nefunkční. Dále v souladu se získanými poznatky na základě provedených analýz bude navržen informační systém. Řešení návrhu je realizováno za pomoci jazyka UML a je zaměřeno na důležité činnosti, které se vztahují k možnosti založení objednávek zákazníkem, registrace různých typů uživatelských účtů, zakládání menu apod. Návrh uživatelsky přívětivého rozhraní bude realizován v rámci grafického prototypu vytvořeného za pomoci nástroje Figma. Vybraná část navrženého informačního systému bude implementována pomocí technologií JavaScript, CSS, HTML.

3 Teoretická východiska

3.1 Informační systémy

Informační systémy jsou velice populární a využívané v současnosti jak velkými, středními, tak malými podniky. Tyto systémy jsou používané organizacemi ke zlepšování, zkvalitnění a automatizaci podnikových procesů, služeb a produktů, zároveň umožnují správu důležitých funkcí, mezi které se může zařadit nákup, prodej, plánování, logistika apod. Na základě těchto funkcí vznikla potřeba pro rozvoj a růst, respektive pro celkové zdokonalení informačních systémů, které jsou využívány v dnešní době [5].

3.1.1 Vývoj informačních systémů

Historická myšlenka vzniku informačních systémů se vztahuje k 50. letům minulého století, kdy byla implementována první podoba IS současnosti. Tento prototyp byl určen ke zpracovávání faktur a mezd a byl implementován v elektromechanických účetních počítacích strojích, což vedlo k určitému snížení nákladů a času na přípravu papírových dokumentů. Dále 60. léta se vyznačují změnou postoje k IS. Informace získávané z informačních systémů se začaly využívat pro reportování, na základě, čeho vznikla potřeba pro počítačové vybavení, které by umožnilo zpracovávat velké objemy dat [17]. Podniky začaly využívat celopodnikové systémy nejen za účelem kontroly zásob ve skladech. Navzdory nedostatkům ve sféře komunikačních technologií se společnost IBM postarala o další průlom ve vývoji informačních systémů a vyvinula nejmodernější software té doby, který řešil materiálové plánování výroby (MRP). Pak počátkem 70. let zásluhou Olivera Wighta, který se věnoval mnoho let problematice výrobních činností, plánování a řízení v podnicích, byl software rozšířen o funkce prodeje a provozu [5]. Na pomezí 70. a 80. let se informační systémy začaly široce využívat jako prostředek řízení managementu, který podporoval a urychloval procesy rozhodování. Dále tato koncepce expandovala na další druhy informací, což přineslo možnost poskytování informací v předem stanovené době. Podnikům tyto změny pomohly k dosažení úspěchu v jejich činnostech, například při vytváření nových produktů a služeb, otevírání nových prodejen, zajišťování vhodných partnerů apod. [17].

Informační systémy v současnosti dosáhly vysokého stupně rozvoje a rozšířily se do dalších sektorů a úrovní podnikání. Zpřístupněním veřejnosti se dosáhlo de facto nekonečných

možností v počítačových, komunikačních, internetových technologiích a informační systémy se staly nedílnou součástí moderního světa.

3.1.2 Systém a informační systém

Systém je základním pojmem, ze kterého IS vychází. Ještě Aristoteles definoval systém jako: *"Mnohé složité věci jako celek více než jen souhrn částí, ze kterých se skládají [1]."*

Informační systémy se neustále rozvíjí, rychle se mění, přizpůsobují se a ovlivňují různorodé odvětví lidských činností, umožnují šetřit čas, peníze a optimalizovat procesy neboli činnosti podniků. Pojem informačního systému již existuje mnoho let, kterému ve společnosti rozumí leckdo především jako komplexnímu technologickému nástroji. Existuje mnoho definic a to:

„Informační systém (IS) je celek složený z počítačového hardwaru a souvisejícího softwaru, k němuž patří také lidé, využívající tento hardware a software, a procesy (činnosti), které přitom vykonávají za účelem sběru, zpracování a šíření informací potřebných k plánování, rozhodování a řízení [2]."

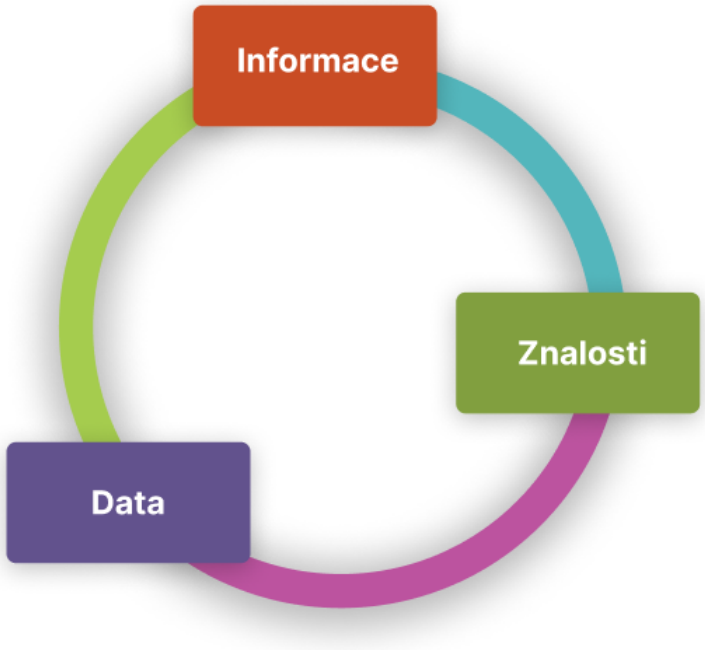
„Informační systém organizace je systém informačních technologií, dat a lidí, jehož cílem je efektivní podpora informačních a rozhodovacích procesů na všech úrovních řízení organizace (firmy) [3]."

„Informační systém je vzájemně propojený soubor informačních, technických, softwarových, matematických, organizačních, právních, ergonomických, jazykových, technologických a jiných prostředků, jakož i personálních, určených ke shromažďování, zpracovávání, uchovávání a šíření informací za účelem stanoveného cíle [17]."

Uvedené definice IS výše jsou odlišné, ale obsahují společné pojmy, které je potřeba více vysvětlit. Jedná se o následující pojmy jako data, informace a znalosti.

3.1.3 Data, informace a znalosti

Pojmy data, informace a znalosti se protínají navzájem a vytváří cyklus souvislostí. Jednotná data ve formě znaků, faktů, měření apod. nemají vypovídající hodnotu. Data nevyjadřují žádný význam, pokud nejsou organizovaná v jakýkoliv ucelený kontextový soubor, který sděluje skutečnost formálním způsobem. Tyto jednotlivé elementy se zároveň dají využívat jako vstupní hodnoty a je lze ukládat, přenášet, zpracovávat a sdílet. Sama data v tom této formě představují pouze soubor neuspořádaných prvků, ale proměňují se v informace za pomoci přidávání jim smyslu, myšlenkového pochodu, klasifikace či strukturování [18].
Informace jsou množinou sdělitelných poznatků neboli jednotlivá fakta, která mají význam a jsou snadno pochopitelná. Tyto údaje jsou proměnlivé a vypovídající prvky o zákonitostech a trendech reálného světa. Jedná se o soubor dat zabalených do kontextu, který obsahuje časový rámec a je relevantní. Data, aby tvořila informace, by měla být interpretovatelná, shrnutelná a zároveň závislá na tom, kdo je potřebuje [19].



Obrázek 1: Cyklus data, informace a znalosti (Zdroj: [51]- upraveno)

Znalosti jsou relevantní informace, které vyplývají z porozumění zákonitostem, vznikajícím na základě zkušeností. Jedná se o kognitivní procesy, které jsou východiskem k zpracovávání a interpretaci informace. Znalosti jsou závislé na inteligenčních schopnostech, procesech porozumění a chápaní, schopnosti dávat věci do souvislostí. Ty jsou získávané studiem, zkušenostmi, uvědoměním či porozuměním [20].

3.2 Klasifikace informačních systémů

Informační systémy se dají zařadit do mnoha různých klasifikací v závislosti na požadavcích kladených na systém, funkce či vlastnosti, které systém obsahuje, službách, které poskytuje, rovněž na základě daného stupně automatizace, podle způsobu použití atd., tudíž neexistuje jednoznačně daná klasifikace informačních systémů, neboť jeden a tentýž IS lze zařadit do více kategorií. Například existuje následující klasifikace IS v souladu se způsobem zpracování informací na různých úrovních řízení podnikání a to:

- Systém (EDP) určený ke zpracování dat za účetními a provozními účely. Jedná se o regulace obchodních transakcí, přípravu platebních dokumentů jako faktury pro externí prostředí. Horizont operativního řízení podnikových procesů se pohybuje od jednoho do několika dnů. V rámci tohoto typu IS se provádí zpracování a evidence událostí, např. zadávání a sledování realizace objednávek, příjem a spotřeba zásob ve skladě, vedení časového rozvrhu atd. Tyto úkoly jsou iterativní a vykonávají se za pomoci účastníků obchodních procesů, jako jsou skladníci a správci. Výsledky obchodních operací jsou zadávány do databáze prostřednictvím formulářů [21].
- Manažerský systém (MIS) je zaměřen na strategickou úroveň řízení, plánování, analýzu a organizaci práce na několik týdnů či měsíců, např. analýzu a plánování zásob, prodeje, výroby. Tyto úlohy jsou pravidelné, periodické a opakovatelné. Výstupem je stanovení potřeb na základě specifikací požadavku na produkt. Řešením této problematiky se zabývají manažeři či vedoucí různých sektorů podniků jako logistiky, prodeje [21].
- Systém (DSS) využíván pro podporu rozhodování se používá především na nejvyšší úrovni řízení (podnikové řízení), které má strategický dlouhodobý význam na rok až několik let. Mezi hlavní úlohy, které se řeší v rámci tohoto systému, patří vytváření strategických cílů, plánování, získávání zdrojů financování, výběr umístění podniku a zároveň se řeší méně globální problematika jako výběr dodavatelů nebo uzavírání smluv s klienty[21].

Existuje klasifikace systémů na základě využití dat jako informační systémy veřejné správy, soukromé či podnikové.

Podnikové informační systémy tvoří největší složku nejvyužívanějších a nejvíce potřebných informačních systémů na trhu. Tyto systémy jsou v současnosti využívané pro různorodé potřeby podniku. V rámci podnikových systémů lze přidělovat různá oprávnění na základě

přidělené role pracovníka, tedy v závislosti na typu pracovní pozice. Existují garanti role, kteří se starají o schvalování úprav, ale o přidělení práv se stará technický administrátor, o správu systému se starají poskytovatelé dalších služeb, jakými může být outsourcované či interní IT oddělení [3].

V rámci podnikových IS lze rozlišit níže následující kategorie, které tvoří vlastní klasifikace subsystémů, rozlišují se podle provádění potřebných akcí pro podnik s informační podporou podniku.

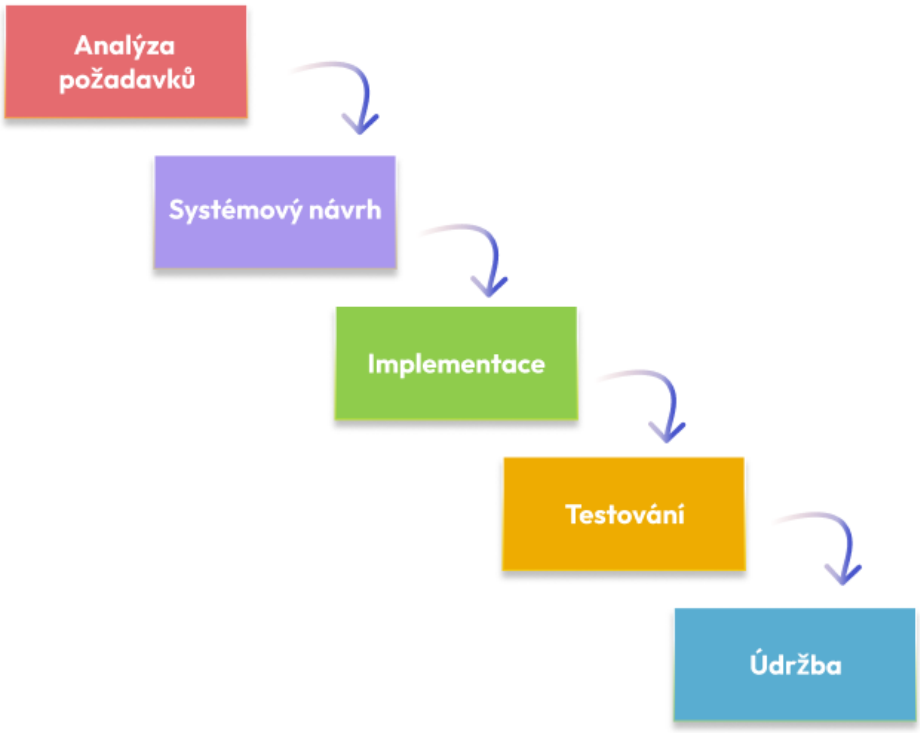
- ERP je systém pro plánování a řízení podnikových zdrojů. Software se zabývá problematikou komunikace a sdílení informací v rámci jedné organizace. Dále umožnuje plánování celého logistického řetězce od nákupu po výdej, expedici a přijetí materiálů, zároveň jejich správu na skladě. Systém zároveň zahrnuje účetnictví a řízení lidských zdrojů a zakládá se na modulárním přístupu, v rámci, kterého se dají přidávat jednotlivé komponenty dle potřeby podniku [5].
- CRM je interakční model, základem tohoto typu je obchodní filozofie, ve které zákazník je středem pozornosti a hlavní cíl je založen na podpoře efektivního marketingu, prodeje a poskytovaní služeb v rámci online podpory určené pro zákazníka. Systém je využíván k vytváření pevného vztahu mezi prodejcem a zákazníkem, prostřednictvím zjištění míry uspokojení, na základě shromažďování a vyhodnocování velkého množství informací o zákazníkovi, identifikace klíčových procesů a zvolení správné marketingové strategie [5].
- ECM systém se vztahuje ke strategické infrastruktuře a technické architektuře pro podporu jednotného životního cyklu nestrukturovaných informací různých typů.
- CPM systém dodržuje koncept měření výkonnosti podniku, který pokrývá celou škálu úloh v oblasti strategického a finančního řízení podniku, veškeré metodiky, procesy a metriky potřebné k měření. Mezi nejrozšířenější patří metodiky řešení a řízení projektu za pomoci stanovených legislativních standardů a norem.
- HRM systém, který je vázán na oblast znalostí a praxe, jehož cílem je zajistit a zorganizovat pracovníky a docílit jejich optimálního využití. Software je využíván pro řízení lidských zdrojů, zaměřen na automatizaci funkcí personálních činností ve společnosti [5].
- EAM je informační systém určený především k automatizaci procesů spojených s údržbou zařízení a jeho opravami a také poskytováním správy a možností úpravy v rámci prodeje.

- EDMS je podnikový systém správy dokumentů. Cílem systému je přenášení strukturovaných souborů, archivace, kontrola dokumentace, centralizované ukládání a příprava dokumentů na jednom uložišti, příkladem může být cloudová řešení [22].
- Workflow (BPM) je komplexní informační systém zodpovědný za workflow podniku, od jednoduché objednávky až po konečné verze používaných dokumentů. Současně se jedná o rychlé a spolehlivé sdílení dokumentů či sběr dat v rámci pracovních skupin.
- Collaboration neboli spolupráce je systém odpovědný za elektronickou interakci lidí, ale není formalizovaný jako pracovní tok a najedná se pouze o „archiv" jako EDMS. Informační systém je používaný k usnadnění sdílení dat, dokumentů, souborů, informací a znalostí mezi týmy a zaměstnanci v organizaci.

3.2.1 Životní cyklus informačních systémů

Životní cyklus informačního systému popsán jako celek fází, kterými produkt prochází od objevení nápadu až po jeho implementaci v kódu a následně až po nasazení v podnicích s další podporou. Fáze životního cyklu systému do značné míry předurčují metodiky vývoje softwaru. Obvykle model zahrnuje analýzu požadavků, modelování, programování, testování apod. [24].
Jednotlivé informační systémy, které byly představeny výše, byly vyvíjeny na základě různých modelů životního cyklu, které v rámci definovaných fází rovněž navazují na časové uspořádání vývoje. Mezi základními modely se rozlišují vodopádový, iterativní, spirálový a prototypový model.

Vodopádový model je vývojový proces softwaru, ve kterém vývoj vypadá jako lineárně sekvenční tok, který postupně prochází fázemi shromažďování a analýzy požadavků, navrhování a prototypování, implementování, testování, integrování a podpory.
Podle vodopádového modelu se vývojář přesouvá z jedné fáze do druhé přísně sekvenčně. Nejprve by měla být zcela dokončena základní fáze vývoje neboli definování a sběr požadavků, jejímž výsledkem je seznam požadavků kladených na systém. Po úplném nadefinování požadavků následuje přechod k modelování, kvůli čemu vznikají podrobné dokumentace, které popisují způsob a plán implementace těchto požadavků.



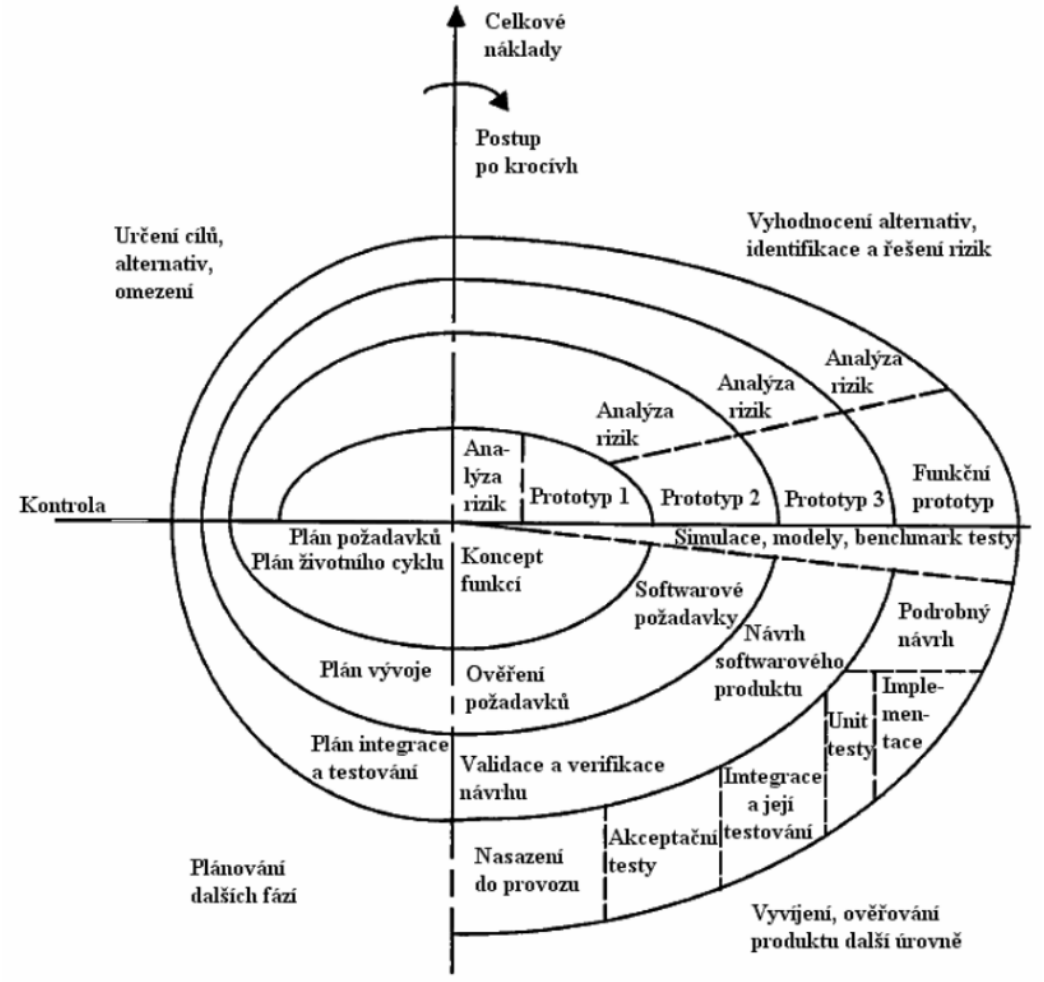
Obrázek 2: Fáze sekvenčního modelování (Zdroj: [23]- upraveno)

Po dokončení návrhu dále programátoři realizují výsledný projekt. Další fáze je integrace jednotlivých komponent, která obsahuje prezentační vrstvu neboli uživatelské rozhraní, aplikační logiku neboli přístupy k datům a jejich správu. Po dokončení implementace a integrace produktu je software testován a laděn. Základní myšlenkou této fáze je odstranění veškerých nedostatků, které se objevily v předchozích fázích vývoje. Poté implementovaný softwarový produkt přechází do finální fáze, v rámci, které je zajištěna jeho podpora, vývoj nové funkcionality a odstranění chyb [23].

Tento model má jak kladné, tak záporné strany. Některé z negativních stránek modelu jsou nepružnost modelu, vyžadovaná pouze aktuální dokumentace, zákazník nemá možnost se seznámit se systémem předem a uživatel nemá možnost se s produktem postupně sžít. Všechny požadavky musí být známy na začátku životního cyklu projektu. Kladné strany modelu jsou vysoká transparentnost vývojových a projektových fází, jasně daná sekvence vývoje, přesně kladené požadavky na stabilitu, přísná kontrola projektového řízení. Ve své podstatě model právě usnadňuje práci při sestavování plánu projektu a shromáždění projektového týmu a zároveň dobře definuje postup kontroly kvality [25].

Iterativní model vychází z vodopádového modelu, zahrnuje rozdělení životního cyklu systému na části, které obsahují fáze a iterace, kdy v každé jednotlivé iteraci se aplikuje vodopádový model jako celek. Iterace připomíná zmenšenou verzi projektu, který zahrnuje všechny fáze životního cyklu. Každá etapa je úplně sama o sobě, ale sada etap nadále realizuje konečný výsledek. Při každé iteraci se jedná o vývoj stejného softwarového produktu. Požadavky kladené na výsledek jsou specifikovány v průběhu vývoje, a proto každá aktuální iterace má možnost být poslední nebo další na cestě k dokončení. Výsledkem po ukončení každé iteraci je použitelný produkt, ale s ohraničenou funkcionalitou ve srovnání s projektem jako celkem. Mimo to v rámci každé následující iterace produkt získává funkcionalitu všech předchozích iterací a zároveň i aktuální, ve které se nachází. Konečná verze systému je výsledkem poslední iterace, která splňuje všechny požadavky z hlediska struktury životního cyklu modelu.
Tento přístup umožňuje se vypořádat s nejistotou a postupně ji odstraňovat, zároveň umožňuje včasné ověření správnosti technického nebo jakéhokoli jiného rozhodnutí.
Použití iterativního modelu snižuje riziko globálního selhání a dává možnost dokončení vývoje na konci jakékoli iterace na rozdíl od vodopádového modelu, kde nejprve je potřeba dokončit všechny fáze [26].

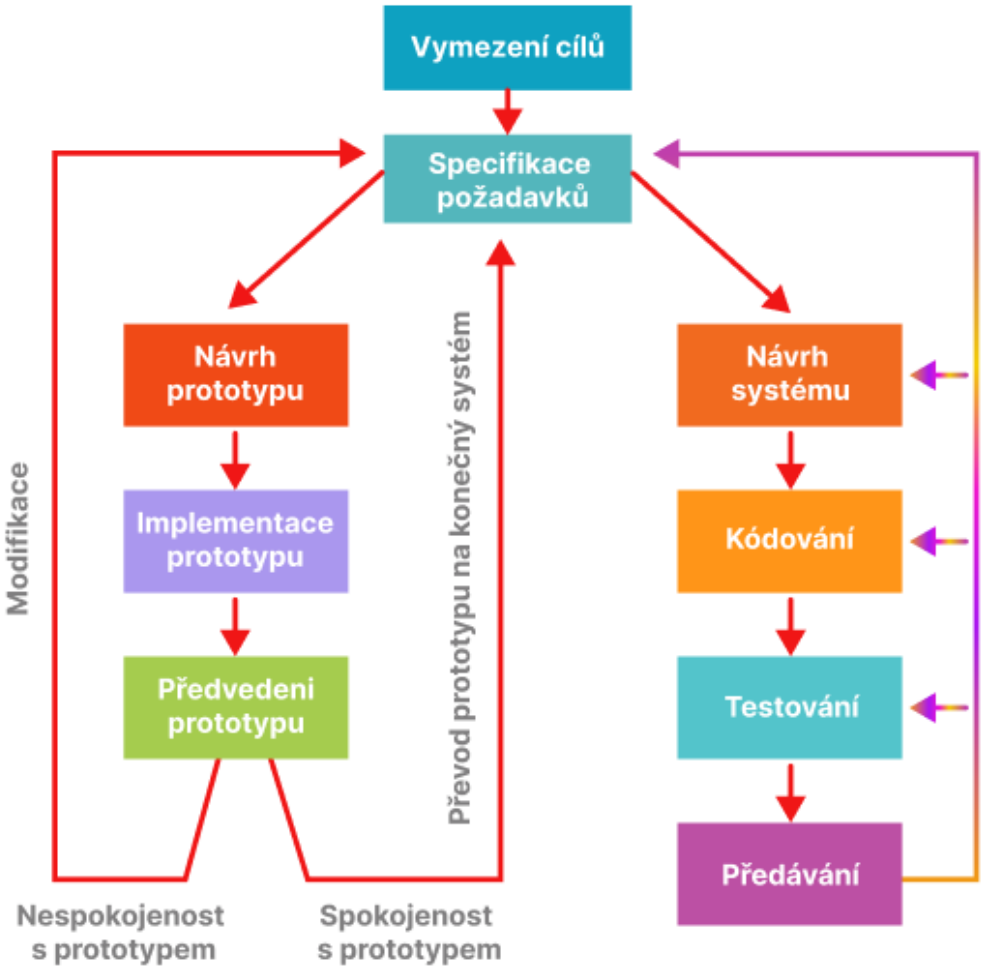
Spirálový model je kombinací iterativního a vodopádového modelu. Tento model byl poprvé definován Barry Boehmem ve svém článku „*A Spiral Model of Software Development and Enhancement*“ v roce 1986, ve kterém byly popsány největší nedostatky vodopádového modelu pokrývané spirálovým modelem [8], [9]. Ve svém schematickém znázornění vypadá jako spirála s mnoha smyčkami. Fáze životního cyklu modelu se střídají a v každé z nich probíhá návrh, kódování, testování atd. Spirálové smyčky představují jednotlivé fáze procesu vývoje a jejich počet se liší v závislosti na projektu, v rámci kterého se aplikuje metodika. Takový proces odráží podstatu už z názvu: stoupání, jedna otáčka cyklu spirály prochází k dosažení konečného výsledku. Navíc není nutné, aby se stejná sada procesů opakovala od cívky k cívce. Výsledky každé z cívek vedou k hlavnímu cíli [27].



Obrázek 3: Fáze spirálového modelu (Zdroj: [28])

Prototypový model je založen na prototypování produktu. Model je využíván v počátečních fázích životního cyklu softwaru. Zjištění požadavků je prezentováno prototypem uživatelského rozhraní. Dále se jedná o výběr jednoho z řady koncepčních řešení neboli implementování scénářů. „*Prototyp je částečnou implementací produktu (nebo jeho částí) v logické, nebo fyzické formě, která prezentuje všechna vnější rozhraní. Jeho hlavním cílem je urychlení vývoje IS využitím prototypů a seznámení zákazníka s prvními verzemi systému v co nejkratší době [29].“* Model taktéž obsahuje analýzu proveditelností a vlastní klasifikaci, která má následující prototypy a to:

- Horizontální, který je využíván výhradně ke tvorbě UI bez ovlivnění logiky zpracování a databáze.
- Vertikální je založen na ověřování architektonických řešení.
- Jednorázové prototypy umožňují rychlý vývoj.
- Evoluční prototypy jsou první aproximací evolučního systému[29].



Obrázek 4: Fáze prototypového modelu (Zdroj: [29] - upraveno)

Moderní metodiky vývoje IS

Metodiky vývoje jsou v současnosti zaměřené na minimalizaci rizik a neúspěchu projektu, na rozdíl od tradičních metodik založených na modelech životního cyklu systému popsaných výše, stanovené požadavky na začátku vývoje lze měnit a upravovat v jeho průběhu.

Agilní metodiky jsou založené na flexibilním a iterativním přístupu vývoje softwaru, kde je kladen důraz na realizace systémových požadavků, jež určuje zákazník. Dále jedním z důležitých a odlišných rysů metodiky je spolupráce a přímá komunikace se zadavatelem, který je chápán jako člen pracovního týmu. Díky tomu se následně lze vyhnout psaní rozsáhlé dokumentace.

Většina agilních metodologií má za cíl minimalizovat riziko pomocí série krátkých vývojových cyklů tzv. iterací. Každá iterace obvykle trvá dva až tři týdny a má všechny fáze životního cyklu systému:

- Plánování.
- Analýza požadavků.
- Modelovaní.
- Programování.
- Testování.
- Dokumentace.

V praxi bývá, že jedna iterace obvykle nestačí k vydání nové verze produktu, čímž se předpokládá nadále přehodnocování rozvojových priorit koncem iterací a jejich realizace v následné iteraci [34].

Obsah Agilní metodiky je přesně definován Agilním Manifestem, který byl vyvíjen a přijat v roce 2001 [31]. Tento manifest nezahrnuje praktická doporučení, ale definuje hodnoty a principy, kterými se řídí úspěšné týmy.

Agilní Manifest obsahuje čtyři hlavní myšlenky a dvanáct principů.

Hlavní body Agilního Manifestu jsou:

1) *„Lidé a interakce jsou důležitější než procesy a nástroje.*
2) *Funkční produkt je důležitější než komplexní dokumentace.*
3) *Spolupráce se zákazníkem je důležitější než dohodnutí podmínek smlouvy.*
4) *Připravenost na změnu je důležitější než dodržování původního plánu [32].“*

Dále principy manifestu si zakládají na uspokojení potřeb zákazníka, včasným a častým dodáváním funkčního softwaru v nejkratší době. Dalšími neméně důležitými položkami metodiky jsou dostačující motivace pracovníků a neustálé zdokonalování kvality produktu, jednoduchost řešení a analýza způsobu zlepšení efektivity [33].

Existuje několik metod, které patří do třídy flexibilní vývojové metodiky, zejména Extrémní programování, SCRUM apod.

Extrémní programování se zakládá na Spirálovém modelu životního cyklu. Název metodiky je spojen s touhou autorů C. Becka, W. Cunninghama a M. Fowlera, aby zlepšili stávající metody vývoje IS a softwaru obecně přesměrováním na novou, „extrémní“ úroveň [33]. Pro hodnocení projektů z hlediska použitelnosti metodiky jsou využívané ukazatele kritičností a rozsahu. Kritičnost je měřena ve směru od nejmenší po největší, zjišťuje následky způsobené vadami softwaru:

1) Vady způsobují ztrátu pohodlí.
2) Závady způsobují ztrátu obnovitelných zdrojů.

3) Vady způsobují ztrátu nenahraditelných zdrojů.
4) Vady mohou ohrozit lidský život.

Rozsah je určen počtem zapojených vývojářů na projektu:

- od 1 do 6 osob – malý tým.
- od 6 do 20 osob – střední tým.
- více než 20 lidí – velký tým.

Tato metodika je nejčastěji využívaná a přizpůsobena k řešení projektů s malým či středním rozsahem a nízkou kritičností.

Obrázek 5: Extrémní programování, proces vývoje (Zdroj: [40] - upraveno)

Vývoj probíhá v malých třítýdenních iteracích, během kterých jsou specifikovány a realizovány požadavky kladené na systém. Ve vývoji se používá neustálá refaktorizace kódu, která slouží k vylepšení čitelnosti bez funkcionálních změn v softwaru. Kód je opakovaně předěláván, a to i v pozdějších fázích vývoje projektu. Kromě výše uvedeného vzniká potřeba v dodržení obecních programovacích standardů, což značně usnadňuje jeho předělávání. Aby se zajistilo, že tyto změny nepovedou k nefunkčnosti systému, je k tomu využívaná kontrola za pomoci testovací metodiky TDD. Metodika extrémního programování zahrnuje automatické testy, které se neustále vyvíjí od počátku vývoje a

výrazně snižují riziko nefunkčnosti systému v důsledku refaktorizace. [40]V procesu vývoje systému se používají způsoby párového programování, kontinuální integrace a zjednodušené modelování.

Párové programování předpokládá, že software je vytvářen dvojicemi programátorů pracujících na stejném počítači. Jeden z nich programuje a druhý hodnotí jeho práci, což umožňuje neustálou kontrolu zdrojového kódu. Během práce na projektu nejsou dvojice pevně dané, a to s cílem, aby každý programátor v týmu dobře porozuměl celému systému.

Průběžná integrace systému umožňuje zachovat jeho integritu po celou dobu vývoje. V tradičních metodikách na rozdíl od moderních se integrace provádí až na konci vývoje, kdy jsou všechny komponenty vyvíjeného systému kompletně připraveny. Integrační problémy mají schopnost se hromadit a vzájemně se překrývat, což může vést i k selhání projektu.

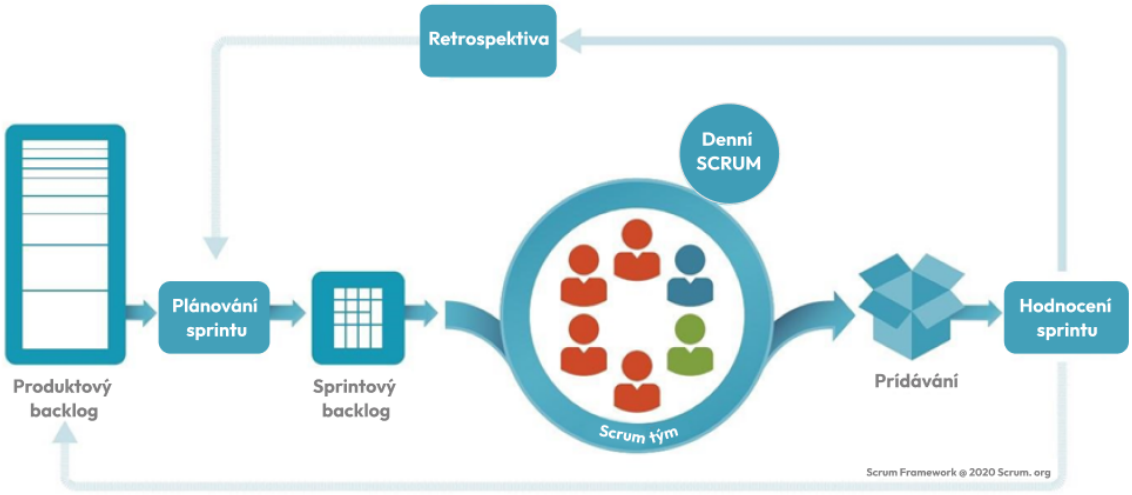
Zjednodušené modelovaní je používáno z důvodu, že systémové požadavky se mohou v průběhu práce opakovaně měnit, což snižuje hodnotu projektu. Modelování se provádí po celou dobu životnosti projektu a jeho zjednodušení zajišťuje odstranění duplicit a následné správné provedení testů[40].

SCRUM

Metodika je často využívaná při vývoji informačních systémů, nachází se v základu flexibilního vývoje software [43]. Scrum klade důraz na kontrolu kvality vývojového procesu. Tento postup byl poprvé popsán pány Hirotaka Takeuchi a Ikujiro Nonaka v roce 1986 [42], [43].

Základními principy využívanými k řízení projektu v rámci této metodiky jsou:

- Transparentnost předpokládá, že všechny důležité aspekty procesu jsou k dispozici všem odpovědným osobám. Transparentnost vyžaduje, aby tyto aspekty byly definovány společnými standardy, které by měly být dodrženy všemi členy týmu.
- Kontrola je způsob sloužící k včasnému odhalení chyb či nežádoucích odchylek od definovaného cíle. Pokud výsledkem kontroly jsou vady systému, je třeba problém produktu opravit, aby nevznikaly defekty v dalších částech systému.
- Přizpůsobení umožňuje přijmout veškeré změny požadavků a přizpůsobit jejich vývoj [41], [42].



Obrázek 6: SCRUM, proces vývoje (Zdroj: [42] - upraveno)

Metodika Scrum umožňuje v pevně fixovaných a krátkodobých iteracích zvaných sprinty poskytnout koncovému uživateli fungující software s novými funkcemi mající nejvyšší prioritu. Úkoly pro další sprint jsou určeny na jeho začátku ve fázi plánování a nelze je měnit po celou dobu jeho trvání. Zároveň je striktně daná doba trvání sprintu, která dává vývojovému procesu předvídatelnost a flexibilitu [42].

Sprint je iterace, ve které dochází k růstu nových funkcionalit softwaru a jeho délka obvykle trvá od 2 až do 6 týdnů. Délka sprintu určuje flexibilitu vývojového procesu, neboli čím sprint je kratší, tím častěji je vydána nová verze a zároveň rychleji vzniká zpětná vazba od uživatele. Na druhou stranu u delších sprintů vývojový tým má více času na řešení, které je stanoveno na počátku sprintu problematiky, či vzniklých chyb během vývoje. Výhodou delších iterací je snížení nákladů na předvádění produktu či další porady, vyžadované v rámci vývoje.

Délka sprintu se volí podle složení týmu, specifikací vykonávané práce a požadavků kladených na systém v rámci procesu vývoje [34].

Dále tato metoda využívá seznam požadavků ohledně funkcionality, seřazené podle stupně priority. Nadále uživatelské přání či prvky propsané v seznamu zvaném „Backlog" jsou implementovány do systému v pořadí, které naznačuje jejich prioritu. Tento seznam v rámci sprintu obsahuje plán a předběžnou předpověď k dosažení nárůstu funkcionality softwaru. Přehlednosti probíhajících činností jednoho sprintu je dosaženo pomocí tabulky úkolů, která zobrazí množství vykonané práce a množství zbývající. Dále tato metodika obsahuje klíčové role, které jsou přidělené jednotlivým osobám v rámci projektové skupiny a to:

- Scrum Master je osoba odpovědná za vedení porad a dodržení všech principů Scrumu, zároveň řeší konflikty a chrání tým před rozptýlením.
- Product Owner – zastupuje zájmy koncových uživatelů a další zainteresované strany[42].
- Scrum tým (Scrum Team) je tým vývojářů projektu velikostí 5 až 9 lidí, skládající se obvykle z testerů, backend a frontend programátorů, software architektů a analytiků [43], [34].

V současnosti existuje široká škála metodik vývoje IS, některé z nich byly zmíněny výše, což dává možnost výběru nejvhodnější metodiky pro řešení každého určitého případu. V rámci výběru se aplikují další kritéria, jako je velikost a složitost projektu, počet vývojářů a požadavky zákazníka systému. Metodiky popsané výše se zakládají na získaných informacích a požadavcích, předběžně provedených analýzách, výstupem, na kterých jsou informace potřebné k vývoji softwaru.

3.2.2 Návrh požadavků na IS

Sběr informací potřebných k realizaci a představě toho, co systém má dělat, je základním a počátečním krokem všech následujících analýz. Jedná se o návrh očekávaných výsledků projektu. Seznam požadavků je přesný kvantitativní popis, který zahrnuje očekávání zákazníka a funkcionalitu systému jako celku, načasovaní a rizika. Požadavky určují předpokládaný výsledek projektu. Dobře definované požadavky jsou základem úspěšného dokončení projektů a minimalizace rizik v průběhu jeho realizace. Tyto informace specifikují, co by se mělo dělat, jak dobře a za jakých omezení. Požadavky, omezení a předpoklady rozdělují problém, který má být řešen, na jeho jednotlivé části, což v konečném důsledku určuje úspěch projektu jako celku.
Existují následující typy požadavků:

- Funkční požadavky definují, co musí daný prvek systému dělat.
- Požadavky na přesnost určují kvantitativní parametry příslušných funkcí prováděných navrženým systémem.
- Omezující požadavky jsou podmínky prostředí, bezpečnosti nebo souladu s předpisy.
- Požadavky na ověření poskytují záruku, že systém má požadovaný výkon ve skutečném provozním prostředí.

- Požadavky na rozhraní definují funkce, charakteristiky, tolerance a omezení všech rozhraní.
- Provozní požadavky určují provozní schopnosti systému a jeho interakci s uživateli.

K definování požadavků existuje mnoho různých metod, ale asi nejvíce známá je metoda *„MoSCoW“*. Tento způsob definování a analýzy požadavku umožnuje zefektivnit řešení stanovených úkolů rozdělením podle priority neboli důležitosti.

„Tvůrce metody, konzultant společnosti Oracle Dai Clegg, navrhuje rozdělit věci do čtyř skupin. Název metody je akronym založený na prvních písmenech názvů kategorií, ke kterým autor přidal „o“, aby se slovo dalo snadno vyslovit.“[37]

Skupiny metody *„MoSCoW“*:

- *„Must have“* - tato kategorie se skládá z požadavků, které jsou pro systém „nezbytné“. Představují nesmlouvatelné potřeby pro systém. V této kategorie se jedná o základní požadavky neboli funkce kladené na systém, bez kterých systém nebude funkčním či jeho realizace nebude dávat smysl.
- *„Should have“* - Požadavky, které by měly být v této kategorii, jsou jen o krok pod nezbytností. Jsou nezbytné pro produkt, systém, ale nejsou životně důležité. Pokud požadavek je vynechán, systém nebo projekt stále funguje, ale požadavky této kategorie však mohou přidat významnou hodnotu systému.

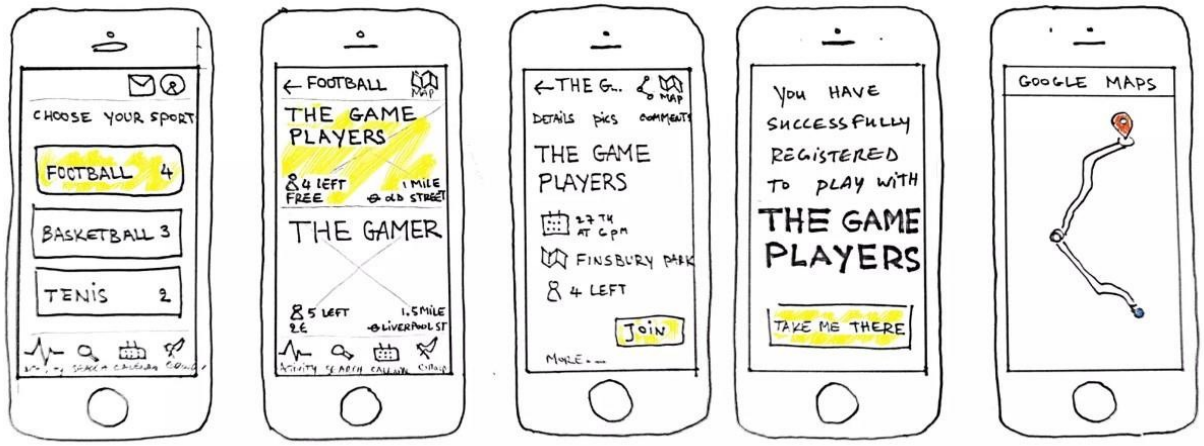
  Požadavky *„Should have“* se liší od *„Must have“* v tom, že mohou být naplánovány pro další „*release“*, aniž by to ovlivnilo to aktuální. Například vylepšení výkonu, drobné opravy chyb nebo nové funkce mohou být iniciativy „měly by být“, ale bez nich produkt stále funguje.
- *„Could have“* - dalším způsobem, jak popsat požadavky této kategorie, je *„mohl mít“*, neboli tyto požadavky nejsou pro základní funkci produktu nezbytné. Ve srovnání s požadavky *„Should have“* však mají mnohem menší dopad na výsledek, pokud jsou vynechány. Požadavky zařazené do této kategorie jsou tedy často prvními, kterým se odebere priorita, pokud požadavky systému v předchozích kategoriích potřebují k realizace větší potřeby, než se očekávalo při odhadu pracnosti.
- *„Will not have“* - jednou z výhod metody *„MoSCoW“* je, že několik typů požadavků řadí do kategorie se záporným významem. Kategorie může spravovat

očekávání ohledně toho, co nebude zahrnuté do konkrétního releasu nebo jiného časového rámce, který je upřednostněn k návrhu a realizaci systému.

Zařazení požadavku do této kategorie je jedním ze způsobů, jak zabránit zvětšovaní rozsahu. Některé požadavky budou v budoucnu upřednostňovány, zatímco k jiným pravděpodobně nedojde [38].

Obrázek 7: Metoda „MoSCoW", priority (Zdroj: [38] - upraveno)

Uživatelské scénáře jsou důležitým krokem při vytváření rozhraní, popisuje interakce uživatele se systémem. Scénáře využívané k testování rozhraní v rámci návrhu systému jsou základním podkladem k určení seznamu entit. Tyto scénáře ilustrují pořadí akcí, které uživatel podnikne k vyřešení stanovené problematiky. Scénář je kostrou rozhraní, skládá se z kroků, které k nejlepšímu pochopení z tabulkového tvaru přenáší k zobrazení ve „*wireframech*", neboli samostatných obrazovek s obsahem hrubého návrhu rozložení elementu. Uživatelsky přívětivé rozhraní začíná právě vývojem uživatelského průchodu neboli toku. Dalším povinným obsahem je přesný popis procesu prováděné akce a informace o rolích entit, které se zúčastní tohoto procesu, následnou neméně důležitou částí scénáře je popis očekávaného výsledku či reakce systému na prováděný děj [47].



Obrázek 8: Testování uživatelských scénáře (Zdroj:[46] - upraveno)

Prototypování je způsob jednoznačného a zjednodušeného zobrazení struktury a hlavních funkcionálních záměrů navrhované aplikace. Jedná se o experimentální proces, při kterém se provádí implementace nápadu do hmatatelných forem od papírové až do digitální verze. Prototyp umožňuje rychle a schematicky ukázat zákazníkovi koncept produktu. Prototypem lze ověřit životaschopnost navrhovaného systému pomocí grafického funkčního typu, či papírového návrhu [46] .

Další cíle prototypování jsou:

- Snaha o nejlepší nápad, myšlenku. Prototyp umožňuje přípravu několika možností a nadále i výběr nejúspěšnějšího nápadu.
- Detekce chyb. Ve fázi vytváření prototypu lze zjistit klíčové vady a nedokonalosti budoucího systému.
- Posouzení použitelnosti. Způsob prototypování a testování za pomoci uživatelských scénářů umožňuje zjistit, na kolik je navrhované rozhraní uživatelsky přívětivé [46].

Prototypování je součástí UX/UI designu, kde designéři často testují své návrhy a provádí výzkum s potenciálními uživateli digitálních produktů, a to buď webových stránek, či informačních systémů apod. Podobné způsoby testování jsou zaměřeny na ověřování uživatelského rozhraní jeho použitelnosti a vnímání ze strany uživatele, dále zkoumání grafických a technických aspektů systémů a jejich jednotlivých elementů [46].

Obrázek 9: Vývoj interaktivního prototypu, Figma (Zdroj: Vlastní tvorba)

3.2.3 Přístup k analýze IS

V softwarovém inženýrství v současnosti existují dva nejvyužívanější přístupy k analýze IS a jsou to strukturovaný a objektově orientovaný způsob dekompozice. První z uvedených je založen na principu funkční dekompozice, kdy je popsána struktura systému z hlediska hierarchie jeho funkcí a přenosu informací mezi jednotlivými funkčními prvky. Druhý objektově orientovaný přístup se zakládá na objektech. V tomto případě je struktura systému popsána z hlediska objektů a vztahů mezi nimi a chování systému je popsáno z hlediska výměny zpráv mezi objekty [16],[36].

Zásadní rozdíl mezi strukturálním a objektově orientovaným přístupem spočívá ve způsobu dekompozice systému. OOP přístup využívá dekompozici objektů, přičemž statická struktura systému je popsána pomocí objektů a vztahů mezi nimi a chování systému je popsáno pomocí výměny zpráv mezi objekty.

Podstata strukturálního přístupu k vývoji softwaru IS tedy spočívá v jeho dekompozici neboli rozdělení na automatizované funkce: systém je rozdělen na funkční subsystémy, které jsou zase rozděleny na subfunkce, ty na úlohy atd. až po konkrétní postupy. Automatizovaný systém si zároveň zachovává celkový pohled, ve kterém jsou všechny

komponenty propojeny. Při vývoji systému „zdola nahoru“, od jednotlivých úkolů až po celý systém vznikají problémy při popisu informační interakce jednotlivých komponent [15], [16].

Strukturovaný přístup je založen na řadě obecných principů, jako je řešení složitých problémů jejich rozdělením na mnoho menších samostatných úloh, které jsou snadno pochopitelné a řešitelné. Další je možnost hierarchického uspořádání jednotlivých částí problému do stromových struktur s přidáním nových detailů ke každé úrovni.
Základní principy strukturovaného postupu:

- Abstrakce je zdůraznění podstatných aspektů systému a abstrakce od nepodstatného.
- Formalizace je založena na potřebě striktního metodického přístupu k řešení problému.
- Konzistence je základem prvků systému.
- Strukturování dat se vztahuje k potřebě organizace dat na hierarchické úrovni [15].

Z hlediska úrovně abstrakce se rozlišuje konceptuální model, logický a fyzický datový model. První se zakládá na tom, co je obsahem systému, jako jsou objekty a vztahy mezi nimi. Další model je relační schéma, které popisuje data z hlediska jejich struktury. Fyzický datový model je zobecněním implementace v rámci databázového produktu [45].
Ve strukturální analýze se používají především nástroje, které ilustrují funkce prováděné systémem a vztahy mezi daty. Každý způsob odpovídá určitým typům diagramů, například:

- ***SADT*** odrážejí funkční strukturu objektu.
- ***DFD*** diagramy toku dat.
- ***ERD*** diagramy „vztah entity“ [50].

Výběr přístupu strukturální analýzy záleží na stanovených podmínkách a typu systému. Modely jsou založeny na zobrazení funkcionality toků dat a vztahu mezi entitami v rámci systému. Modely jasně odrážejí funkční strukturu objektu, prováděné akce, vazby mezi těmito akcemi, popisují logiku a interakce procesů. Další popisují datové zdroje, logické funkce a toky nejen uvnitř, ale i mimo systém.
Hlavní výhodou strukturální analýzy je možnost získat kompletní informace o každé části přísně regulované struktury. Analýza umožňuje identifikovat například nedostatky související s duplicitou apod.

Objektově orientovaný přístup je založen na objektech, které obsahují definované vlastnosti neboli atributy a určité chovaní. *„Každý objekt je schopen reagovat na události. Informační systém z pohledu objektově orientovaného přístupu je chápán jako množina spolupracujících objektů. Objektový přístup lépe odpovídá chování reálného světa* [41]*."*

Základní vlastností objektově orientovaného přístupu, která se vztahuje k návrhu informačních systému, je dědičnost, zapouzdření a polymorfismus, občas se přidává ještě další vlastnost jako genericita.

Tyto mechanismy jsou plně implementovány v objektově orientovaném přístupu než ve strukturovaném.

- ***Dědičnost*** je způsob, podle kterého lze obecné znalosti aplikovat na užší. Ve vztahu ke třídám to znamená, že podřízená třída kompletně zahrnuje neboli dědí všechny atributy a metody definované v nadřazené třídě. V tomto případě v podřízené třídě lze definovat další atributy a metody apod.
- ***Zapouzdření*** neboli utajení informací. Obsah vnitřní struktury prvků systému je skryt. Tento princip předepisuje výměnu informací mezi objekty systému pouze v minimální nutné míře, omezení přístupu k atributům a metodám objektů neboli třídám z jiných objektů a úplné skrytí algoritmické implementace metod z jiných objektů.
- ***Polymorfismus*** je konstrukce prvků modelu, která nabývá různých vnějších forem nebo funkčnosti neboli chování v závislosti na okolnostech.
- ***Genericita*** umožňuje vytváření vlastních kolekcí [41].

S přihlédnutím k výše uvedeným vlastnostem je podstatou objektově orientovaného přístupu k analýze a návrhu informačních systémů rozložit systém na třídy, které odpovídají objektům zkoumané oblasti a vytvořit z nich hierarchii v podobě orientovaného grafu využívajícího vztahy kompozice a dědičnosti.

Základními součástmi objektově orientovaného přístupu jsou jednotné procesy, UML a návrhy [39],[44].

Jednotné procesy jsou procesy vývoje softwaru, které poskytují efektivní přístup k rozdělení úkolů a odpovědností v rámci vývojové organizace. Jednotný proces pokrývá celý životní cyklus informačního systému od definice požadavků až po údržbu a je zobecněným rámcem, který lze aplikovat na vývoj a údržbu široké škály systémů.

Nedílnou součástí je jazyk UML sloužící k definování, vizualizaci, tvorbě návrhu a konstrukci modelů systému ve formě diagramů a dokumentů založených na objektově orientovaném přístupu.

Hodnocení kvality IS je založeno na mezinárodních normách, které hodnotí software především podle jakostí výrobku a požadavků. V rámci hodnocení systému se často využívá metoda „*FURPS*“, která obsahuje následující kritéria

- Funkčnost.
- Použitelnost.
- Spolehlivost.
- Účinnost.
- Udržitelnost [14].

3.3 UML

Jazyk UML neboli Unified Modeling Langu je sémanticky a syntakticky bohatý vizuální modelovací jazyk, který je základem pro návrh, architekturu a implementaci softwaru, zároveň slouží k zobrazení procesních toků a primárně je využíván k modelování rozsáhlých informačních systémů v různých aplikačních oblastech. Jazyk je univerzální a jednoduchý, skládá se z digramů popisujících hranice, strukturu a chovaní systému a objektu v něm. Nejedná se o programovací jazyk, ale na základě digramů a za pomoci dalších nástrojů se dá vygenerovat kód v různých programovacích jazycích, vzhledem k tomu, že UML má přímý vztah k objektově orientované analýze a návrhu [39],[35].

„Jazyk UML se poprvé objevil na scéně v 90. letech, když tři softwaroví inženýři, Grady Booch, Ivar Jacobson a James Rumbaugh, chtěli přijít s méně chaotickým popisem stále složitějšího vývoje softwaru, a zároveň oddělit metodiku od samotného procesu [36].“
Tento standard objektově orientovaného jazyka byl přijat sdružením OMG (Object Management Group), které se zabývá regulováním cílů, způsobů vývoje a funkcionality jazyka [35].

3.3.1 Základní charakteristiky jazyka

Mechanismy UML jsou konzistentně aplikovány v celém modelovacím jazyce. Existují čtyři základní mechanismy:

- Specifikace neboli popis „pozadí“ modelu.
- Ornamenty či možnost rozšířeného popisu libovolného symbolu UML.
- Podskupiny se zakládají na popisu určitých způsobů reprezentace objektů reálného světa v modelu.
- Rozšíření je mechanismus aplikovaný, když základní možnosti UML nesplňují předložené požadavky.

Specifikace je textový popis modelu, který odráží jeho podstatu. Libovolný model lze prezentovat jak graficky, tak textově. Specifikace se také používají pro usnadnění modelování, protože řada prvků modelu může být přítomna ve specifikaci a chybět v diagramu.

Tento jazyk není jen o diagramech, tyto naopak slouží k vizuálnímu zobrazení modelu IS a obsahují komplexní specifikace, které zapadají do celkového modelu.

Ornamenty se používají, pokud vizualizace jako taková není dostačující k reprezentaci libovolného prvku diagramu. Tento mechanismus odráží důležité charakteristiky modelu, jako například přetížení. Doplňky se zpravidla zobrazují ve formě obdélníku, který obsahuje informace o prvku modelu [50],[13].

Podskupiny mají následující mechanismy:

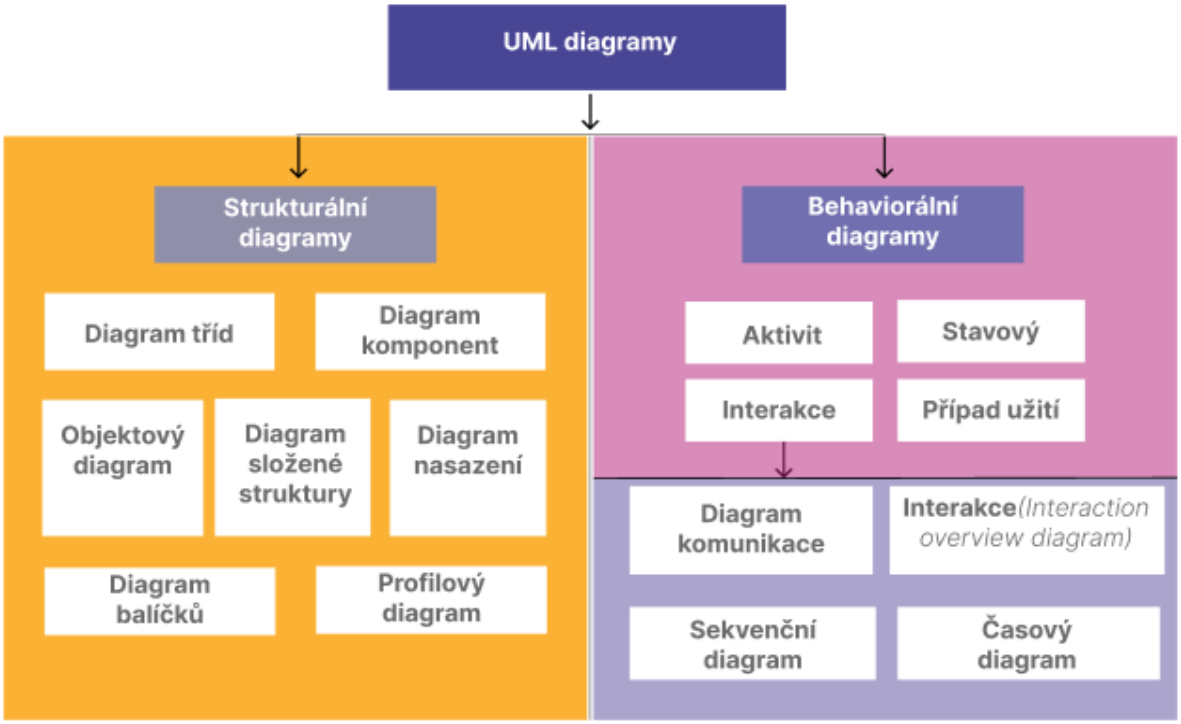
- Klasifikátory a instance neboli klasifikátorem je abstraktní pohled na objekt a v souladu s tím instance je konkrétní objekt.
- Rozhraní a implementace umožňují oddělit to, co se dělá, od toho, jak se to dělá neboli předmět vykonávaní a způsob.

Rozšíření existuje třemi hlavními mechanismy, které podporují flexibilitu jazyka a přizpůsobení specifickým podmínkám.

- Omezení rozšiřují sémantiku prvku, a tím umožňují přidávat nová pravidla.
- Stereotypy mají schopnost definovat nové prvky modelu na základě stávajících.
- Označené hodnoty dávají možnost přidat nové speciální informace do specifikace položky [13].

3.3.2 Diagramy

Diagramy jsou základním způsobem reprezentace sady prvků jazyka ve tvaru souvislého grafu s vrcholy (entitami) a hranami (vztahy), sloužící k vizualizaci systému z různých perspektiv.



Obrázek 10: UML diagramy (Zdroj: [50] - upraveno)

V UML existuje 8 typů diagramů a to:

Diagram tříd neboli „Class diagram“, který zobrazuje třídy, rozhraní, objekty a kooperace, jakož i jejich vztahy. Používá se v simulaci objektově orientovaného systému.

Diagram případu užití ukazuje „Use case diagram“ precedenty a aktéry, je zvláštním případem tříd, jakož i vztah mezi nimi. Používají se při modelování chování systému.

Sekvenční diagramy jsou speciální případy interakčních diagramů. Tyto diagramy ukazují vztahy mezi objekty, zprávy, které si objekty mohou vyměňovat.

Diagramy interakce odkazují na dynamický pohled systému. Ve stejné době diagramy sekvence odrážejí časové pořadí zpráv a schémata spolupráce – strukturální organizace výměny objektové zprávy [11].

Stavové diagramy představují automat, včetně stavů, přechodů, událostí a typů akcí. Tyto se používají při modelování chování rozhraní, třídy, popřípadě spolupráce v závislosti na sledu událostí.

Diagram aktivity představuje přechody toků mezi objekty z jedné činnosti do druhé uvnitř systému.

Diagram komponent představuje závislosti mezi jednotnými komponenty. Schémata komponent mapují na jednu, či více tříd, rozhraní nebo spolupráce.

Existuje další druh diagramu **„Deployment diagram“**, který reprezentuje konfigurace procesních uzlů systému a v nich hostovaných komponent [10].

3.4 Systémové modelování

Modelování systému je podmíněné počátečním postupem a je vytvořením návrhu, což je abstraktní model navrhovaného systému. Tento návrh umožnuje zobrazení stavěné problematiky z různých úhlů pohledu. Jazyk UML obsahuje následující, základní sadů diagramů, které graficky zobrazí jednotlivé modely systému v závislosti na již existujících podmínkách. Model tříd se zobrazí, systém ze statického, objektového pohledu a odpoví na otázku „co“ je proměnlivým obsahem. Dále modely interakce a stavů se zobrazí dynamické chovaní a popíše, jak a kdy se mění navrhovaný systém [15].

3.4.1 Model tříd

Diagramy tříd odráží různé vztahy mezi jednotlivými entitami zkoumané oblasti, jako jsou objekty a subsystémy, ale tyto diagramy neposkytují informace o časových aspektech provozu systému.

Základními komponenty diagramu jsou třídy, které sdělují atributy a chovaní mezi jednotnými třídami, zároveň existují určité případy, ve kterých třídy mají abstraktní charakter a jsou instancí [16][16]. Třídy mají vlastní název „*Třída objektů popisuje skupinu objektů s podobnými vlastnostmi (atributy), společným chováním (operace), stejnými vztahy k jiným objektům*“ [16].

Třídy obsahují informace o vlastnostech objektu neboli atributu, který zahrnuje popis množiny hodnot, které mohou instance této třídy nabývat. *„Atributy popisují datové vlastnosti objektu. Každý název atributu může být následován volitelnými detaily jako typ a počáteční hodnota“* [16] . Třída může mít libovolný počet atributů nebo vůbec žádné. V programovacích jazycích jako je C++, Java atributy odpovídají proměnným, deklarované ve třídě.

Dále třídy popisují chování objektu neboli operace. *„Každá operace má cílový objekt jako implicitní argument. Operace může mít argumenty (navíc k cílovému objektu). Metoda je implementace operace na třídu.“* Třída může obsahovat libovolný počet operací nebo vůbec je neobsahuje. V programovacích jazycích operace odpovídají funkcím deklarovaným ve třídě.

Dále diagramy tříd popisují vazby mezi třídami objektu. Mezi základní patří asociace, generalizace, kompozice, agregace, dědičnost.

Asociace popisuje vztah mezi jednotnými třídami, či instancemi objektů, například zaměstnanec pracuje v síti lékáren, síť má lékárnu ve městě. Popisuje vztah v rámci přímé vazby mezi instancemi tříd objektů.

Agregace je formou objektové kompozice v objektově orientovaném přístupu, tento vztah je také využíván k reprezentaci systému, v rámci kterého se jedná o vztahy typu „součást-celek“. *„Symbolem agregace je čára vztahu s „diamantem” na straně celku. Agregace může mít libovolný počet úrovní“[16].* Například rozdělení osobního počítače do komponent: systémová jednotka, monitor, klávesnice a myš.

Kompozice je speciálním případem agregace, ve kterém jednotlivé části jsou v určitém smyslu v rámci celku. Specifičnost vztahu mezi nimi spočívá v tom, že části nemohou existovat odděleně od celku.

Generalizace znázorňuje vztah, ve kterém se sdílí podobnosti mezi třídami při zachovaní jejich odlišností, jedná se o nadtřídu a podtřídu neboli jeden prvek specializuje, rozšiřuje či upravuje jiný [16].

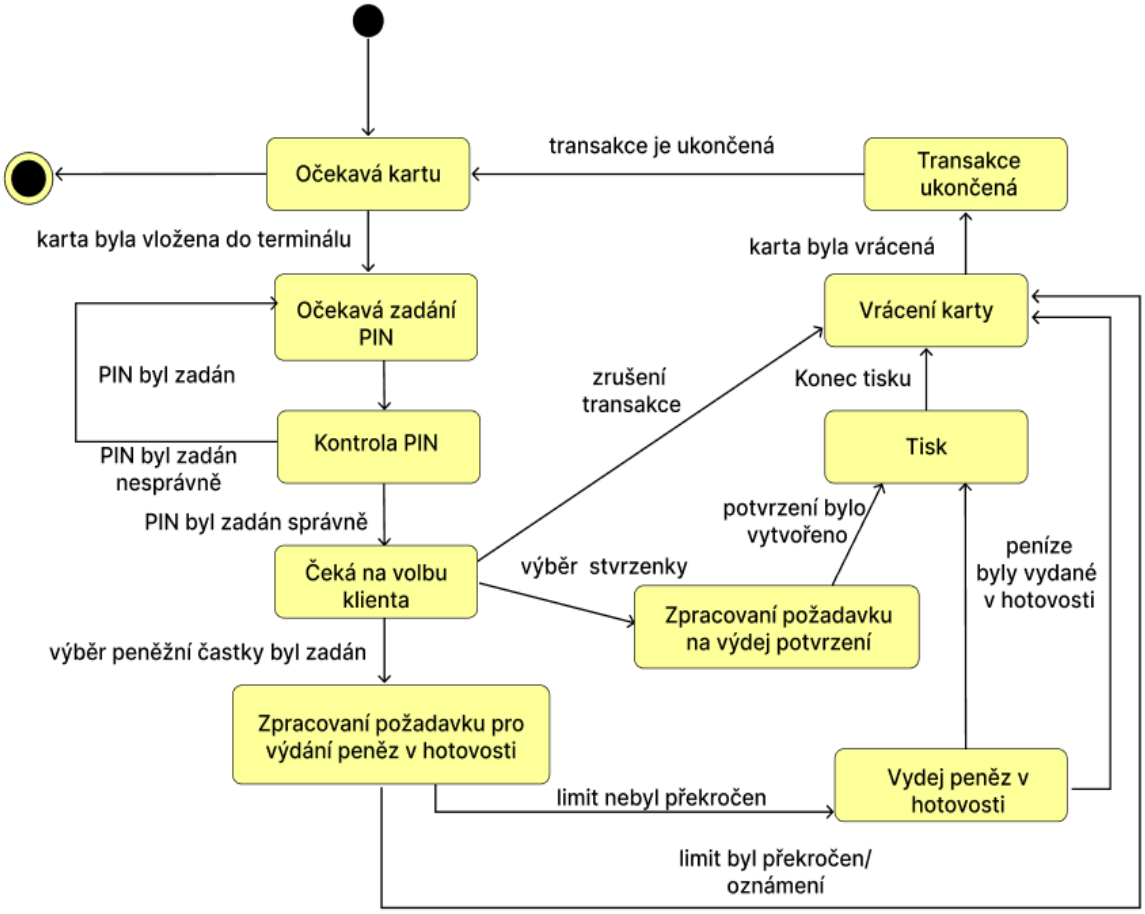
3.4.2 Model stavů

Hlavním účelem stavového modelu je popsat možné posloupnosti stavů a přechodů, které dohromady charakterizují chování prvku modelu během jeho životního cyklu. Nejčastěji se k popisu chování používají stavové diagramy samostatné instance tříd neboli objektů. Stavový diagram je v podstatě speciální druh grafu, který představuje automat. Vrcholy tohoto grafu jsou stavy a některé další typy prvků automatu neboli pseudostavy, které jsou znázorněny odpovídajícími grafickými symboly. Dále digram znázorňuje, že není jenom samotný stav, ale přechod a událost.

V UML jazyce je stav popsán abstraktní metatřídou a využíván k modelování určitých situací pro splnění, při kterých vzniká potřeba splnění stanovených podmínek. Hodnoty atributů třídy nebo objektu, při změně hodnot odráží změnu stavu simulované třídy [13].

Stav na diagramu je znázorněn obdélníkem se zaoblenými vrcholy, tento obdélník často obsahuje dvě části, ve kterých je uváděn název stavu a seznam interních akcí neboli seznam přechodu v daném stavu. Stavy jsou počáteční a koncový. Počáteční stav je speciální případ stavu, který neobsahuje žádné vnitřní akce. V tomto stavu se objekt nachází v počátečním čase. Grafické zobrazení počátečního stavu v jazyce UML je označeno jako vyplněný kruh, ze kterého může vystoupit pouze šipka odpovídající přechod. Konečný stav je zvláštní případem stavu, který také neobsahuje žádné vnitřní úkony. Grafická reprezentace tohoto stavu je vyplněný kruh, v kruhu s šipkou odpovídající přechod [11].

Je třeba poznamenat, že hlavní výhodou tohoto stavového diagramu je možnost modelovat podmíněný charakter implementace všech případů užití.



Obrázek 11: Diagram stavu chování bankomatu (Zdroj: Vlastní tvorba)

3.4.3 Model interakce

V rámci těchto modelů se jedná o zobrazení vzájemného ovlivňování a působení tříd modelu. Interakce na různých úrovních abstrakce jsou představené následujícími diagramy uváděnými níže.

- Diagram případu užití.
- Diagram aktivit.
- Sekvenční diagram.

Tyto diagramy znázorňují iterace systému s vnějším okolí, dále se jedná o podrobný či přesný rozbor, ve kterém se zachycuje chování jednotlivých metod v čase, či popis aktivit[50].

Diagram případů užití

Diagramy případů užití zobrazují interakci mezi případy užití reprezentujícími funkce systému a aktéry, kteří zastupují uživatele nebole fyzické osoby, či hardware, který přijímají informace, či předávají ji do systému. Diagram vypomáhá ve vymezení hranic vnější a vnitřní interakce. V rámci vnitřní části systému se jedná o případy užití a ve vnější části vystupují aktéři, kteří v rámci iterace se systémem vstupují do něho a využívají případy užití [10]. Diagramy případu užití umožňuje usnadnit pochopení toho, k čemu systém je určen a co by měl dělat.

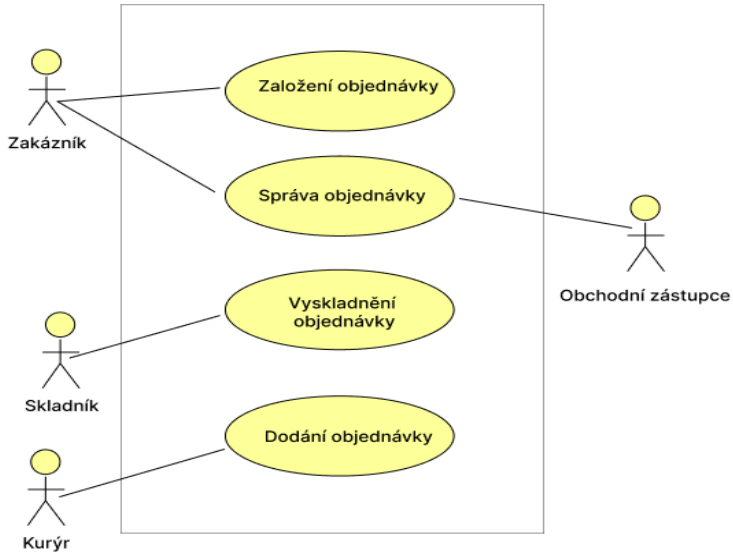
Případ užití se používá k určení společných rysů chování systému, například zadávání objednávky na nákup zboží.

Aktér je entita mimo modelovaný systém, který spolupracuje se systémem a využívá jeho funkčnost. Zároveň herci slouží k označení uceleného souboru rolí, které hrají uživatele v procesu interakce s navrženým systémem.

K vyjádření vztahů mezi aktéry a případy užití platí standardní typy vztahů popsaných výše. Asociační vztahy aplikované v diagramech jsou využívané k vyznačení určité role herce v určitém případu užití, jinak asociace definuje sémantické rysy interakce aktérů a případů užití v grafickém modelu systému. Asociační vazba obsahuje název a násobnost.

Dále v rámci *„Use Case“* existuje další typ vazby *„extend“,* který je využíván k rozšíření mezi případy užití za pomoci tečkované čáry se šipkou, popisuje zahrnutí jednoho případu užití v druhém. Vazby *„include“* jsou často využívány v tomto druhu diagramu. Vztah

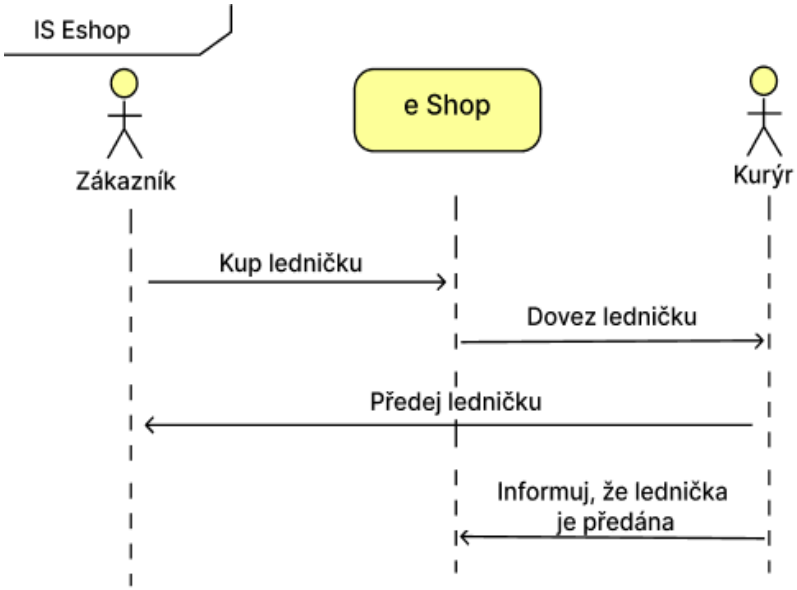
zobrazuje vazbu mezi dvěma případy užití a vyjadřuje, že chování jednoho případu užití je zahrnuto jako složený komponent do sekvence chování jiného případu užití [10].



Obrázek 12: Diagram případů užití (Zdroj:[49] - upraveno)

Sekvenční diagram

Sekvenční diagramy odrážejí tok událostí, ke kterým dochází v rámci případu užití. Tyto diagramy ukazují pouze ty objekty, které se přímo účastní interakce, kde klíčovým momentem je dynamika interakce objektů, na rozdíl od případů užití se tady nejedná o statickou asociaci s jinými objekty [13]. Příklad sekvenčního diagramu je znázorněn na obrázku.



Obrázek 13: Sekvenční diagram (Zdroj: [48] - upraveno)

Diagram aktivit

Při modelování chování navrženého nebo analyzovaného systému vzniká potřeba v reprezentaci procesu změny stavů a zároveň v popisu vlastností prováděných operací. K modelování procesu v jazyce UML jsou určené diagramy aktivit, v rámci kterých je aplikován grafický zápis podobný zápisu stavového diagramu. Důvodem tomu jsou symboly stavu a přechodu, ale rozdíl spočívá v sémantice stavů, které se používají k reprezentaci akcí, či událostí. Každý stav v diagramu aktivit odpovídá provedení nějaké základní operace a přechod na další stav se spustí až po dokončení operace v rámci předchozího stavu.

V kontextu UML aktivita je nějaký soubor jednotlivých výpočtů prováděných automatem. V tomto případě jednotlivé výpočty přivádí k určitým výsledkům nebo akci. Jedná se o zobrazení logiky nebo posloupností přechodu z jedné činnosti do druhé a o konečný výsledek prováděné činnosti, který za následek může mít změnu stavu systému nebo návrat některých hodnoty[13].

Mezi základními prvky diagramu aktivit patří stav, který znázorňuje akce a je zvláštním případem samotného stavu s nějakou vstupní akcí a alespoň jednou akcí s obsahem přechodového stavu. Grafická reprezentace stavu je obdélník se zaoblenými úhly, uvnitř kterých je napsán výraz neboli akce, která je jedinečná. Akce nadále může být napsána v programovacím jazyce bez žádných implicitních omezení.

Základními prvky k návrhu diagramu aktivit jsou využívané další tvary:

- **Šipky** znázorňují toky, které diagram protínají od začátku do konce procesu.
- **Obdélník** reprezentuje akce neboli operace.
- **Řídící uzel** je abstraktní akční uzel, který koordinuje toky akcí a je koncový.
- **Diamant** je rozhodovací prvek v rámci akce, který definuje pravidla větvení a možnosti vývoje scénáře.
- **Černý kruh** je počáteční uzel aktivity.
- **Černý kruh s tahem** určuje konec procesu a zastavuje všechna vlákna v daném diagramu[50].

4 Vlastní práce

Tato část diplomové práce obsahuje praktické řešení vlastního návrhu informačního systému určeného k prodeji kávy a dalšího občerstvení od různorodých podniků. Hlavním záměrem projektu je návrh IS v souladu s aktuální problematikou, jež je popsaná v úvodní časti práce. Informační systém umožnuje malým podnikům a start-upům začít podnikání s menšími náklady na provoz vlastního softwaru s využitím systému v rámci zpoplatněných služeb. Základní myšlenkou je návrh IS pro prodej kávy a zákusků s možností osobního vyzvednutí pro zákazníky a pro podnikatele je zároveň výhodou jeho využití k získání většího množství zákazníků, a tím zvýšení konkurenceschopnosti v prováděné činnosti.

Návrh systému poskytuje strukturovaný náhled mobilní aplikace a je vytvořen pomocí softwarového nástroje „draw.io“. Eventualitou tohoto řešení jsou diagramy, které jsou vykresleny v jazyce UML, jenž byl popsán v teoretické části.

Vytvoření návrhu si zakládá na předem stanoveném cíli, který je východiskem pro vymezení hranic systému, poté jsou navrženy požadavky kladené na systém, ty jsou základem pro tvorbu diagramů. Jednotlivé funkce a celkový přehled struktury systému jsou obsaženy ve statickém, dynamickém a interakčním modelu. Dílčí části návrhu jsou znázorněny dalšími diagramy jako:

- Případy užití.
- Diagramy tříd.
- Sekvenční diagramy.
- Stavové diagramy.
- Diagramy aktivit.

Pro znázornění uživatelské podoby systému na základě již navrženého modelu byl vytvořen grafický návrh mobilní aplikace za pomoci nástroje Figma a s využitím vlastních nabytých poznatků a dovedností v oblastí UX/UI. K testování grafického prototypu byly vytvořeny scénáře pro různé typy uživatelů navrhovaného systému. Dále je část tohoto podkladu realizována v rámci prototypu za pomoci dalších technologií jako HTML, CSS, JavaScript, Spring Boot, Maven, MySql.

4.1 Analýza současné situace

Analýza je zaměřena na průzkum informačních systémů určených pro objednávku a rozvoz jídla či nápojů. V současnosti existuje velké množství restaurací a kaváren, které mají vlastní webové stránky či mobilní aplikaci určené pro pohodlné objednávání občerstvení. Podobných příkladů je velice mnoho a jsou ve většině stavěné na „CMS“ systémech neboli se jedná například o „Wordpress“. Existují i další druhy aplikací, které seskupují mnoho podniků v rámci jednotného uživatelského rozhraní a nabízí možnost objednání jídla a pití. Mezi největšími společnostmi, které působí v současnosti na českém trhu s podobnými nabídkami, jsou společnosti „Dáme Jídlo“, „Wolt“ nebo „Bolt Food“.
Tyto společnosti se zaměřují na prodej různorodých občerstvení a jejich rozvoz.
„Dáme jídlo“ je jedna z nejlepších webových aplikací, která poskytuje českým uživatelům v rámci jednotného rozhraní velice široké odvětví rozmanitých služeb. Společnost rozvíjí svou platformu podle současných trendů a výsledkem je moderní, jednoduché a pohodlné rozhraní s velkým množstvím prvků a funkcí. Nejedná se pouze o desktopovou verzi webové aplikace, ale je k dispozici i mobilní aplikace. Podobné aplikace jsou přínosné nejen pro koncové uživatele systému kvůli rozmanitému výběru zboží, ale také pro dodavatele jídla, ať už se jedná o velké, či malé podniky. V rámci této aplikace mají restaurace možnost nabídnout svoje jídlo většímu množství zákazníků. Služba, kterou poskytuje tato společnost, je velice výhodná pro podniky, které začínají působit na trhu a nemají stálé zákazníky, zároveň je však poskytovaná služba zpoplatněná. Konkurenty této společnosti jsou *„Wolt“* a *„Bolt Food“*, ale na rozdíl od *„Dáme jídlo“* a *„Wolt“ „Bolt Food“* zpřístupnili tento informační systém koncovým zákazníkům pouze prostřednictvím mobilní aplikace pro iOS a Android.
Služby, které poskytují společnosti, jsou:

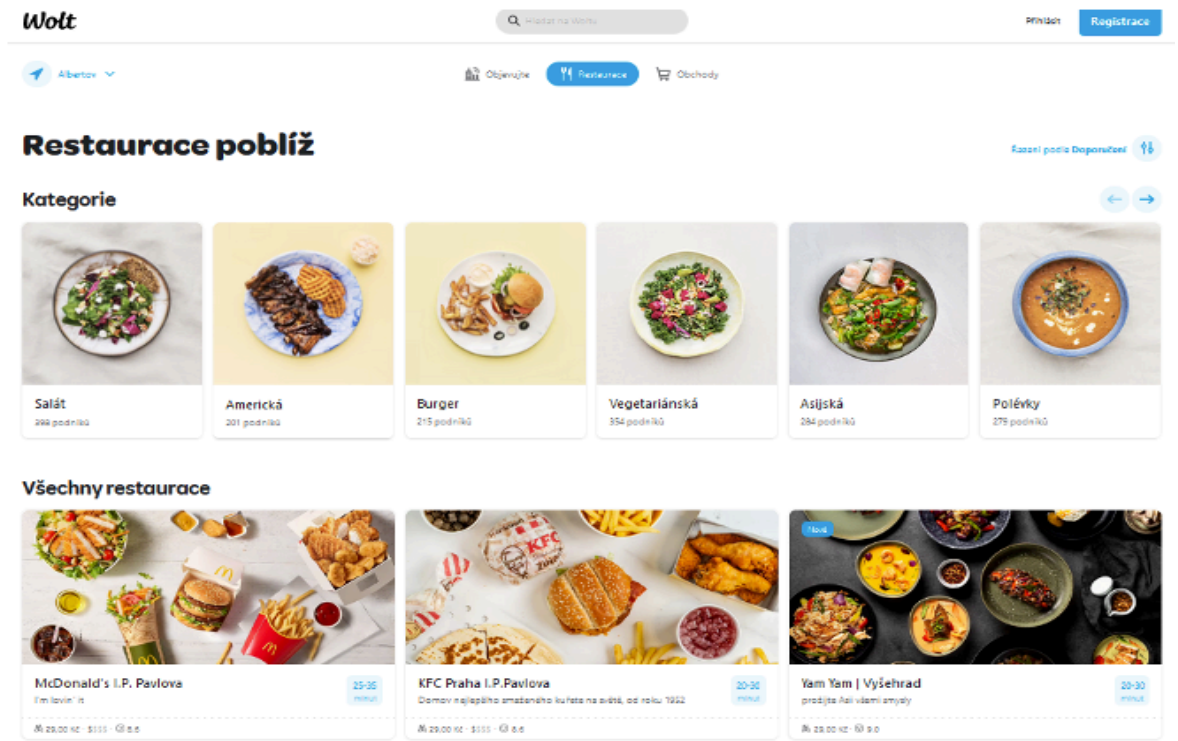
- Objednávka a dovoz jídla.
- Objednávka a vyzvednutí jídla na místě.
- Nákup potravin online umožnil *„Wolt“* a *„Dáme jídlo“*.

Rozhraní poskytované společnostmi se liší způsobem registrace a použitelností.
Informační systém společnosti „Dáme jídlo“ si zakládá na znalosti konkrétní adresy zákazníka, podle toho je schopen mu nabídnout nejbližší restaurace. Systém nabízí možnost registrace, a to jak pro zákazníka, tak i pro podnik. Jednou z dalších funkcí je seznam historie objednávek. Uživatelské rozhraní je navržené tak, aby přilákalo pozornost uživatele,

převážně v červené barvě, což z psychologického hlediska podněcuje chuť a ochotu kupovat.

Obrázek 14: Hlavní stránka „Dáme jídlo“ (Zdroj:[52])

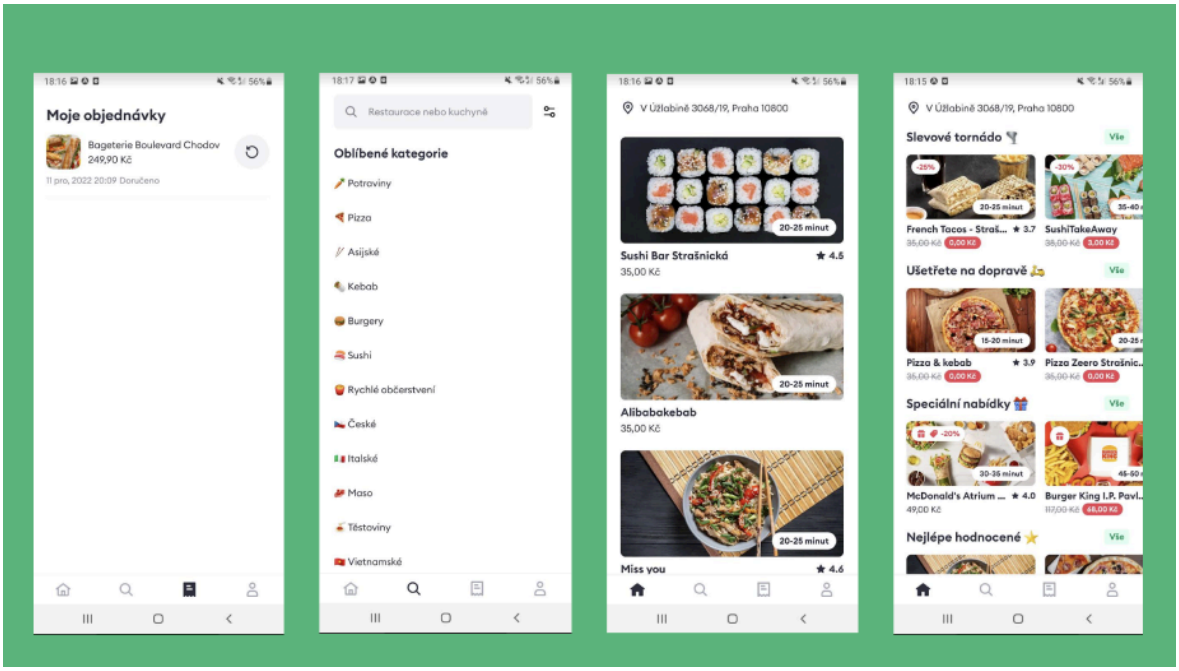
„Wolt“ je mezinárodní služba a má menší zastoupení na trhu než „Dáme jídlo“, ale poskytuje podobné rozhraní s dominantní modrou barvou. Funkčně je stejné jako „Dáme jídlo“, ale umožňuje výběr restaurací podle doby trvaní donášky.



Obrázek 15: Hlavní stránka „Wolt“ (Zdroj:[53])

„Bolt Food“, jak bylo zmíněno výše, poskytuje pouze mobilní verzi uživatelského rozhraní. Jedná se o mezinárodní platformu. V rámci informačního systému se dá prohlédnout čas

doručení objednávky, ale na rozdíl od „Dáme jídlo“ je možná platba pouze kartou. Systém poskytuje uživateli přehled o minulých objednávkách a možnost, jak objednat donášku přes kurýra, tak vyzvednout na místě.



Obrázek 16: Mobilní aplikace „Bolt food“ (Zdroj:[54] - upraveno)

4.2 Funkce systému

Na základě pozorování současné situace na trhu donáškových platforem, bylo zjištěno, že nejsou na trhu systémy, pomocí kterých se dají objednávat pouze nápoje. Na rozdíl od zmiňovaných aplikací je informační systém *„Your takeaway“* zaměřen na malé a střední kavárenské podniky bez možnosti donášky pouze s možností vyzvednutí kávy na místě ve stanoveném čase. Funkce systému lze rozdělit podle rolí uživatele, pokud se jedná o roli běžného uživatele, systém mu umožnuje založit si účet, vytvářet objednávky, upravovat si údaje nebo rušit objednávky. Uživatelé typu podnikatel mají k dispozici jiné rozhraní systému než běžný uživatel. Prostřednictvím tohoto rozhraní je možné přidávat nebo upravovat jednotlivé položky menu, dále je možné přidávat účty zaměstnanců. Administrátor v rámci systému upravuje účty podnikatele.

Výhodou tohoto systému by mělo být zjednodušení platby. Způsob platby v rámci systému by byl zprostředkován pomocí platební brány. Jednou z dalších výhod systému pro běžného uživatele je stanovení přesného času, ve kterém chce mít svůj nápoj připravený k vyzvednutí.

4.3 Vymezení požadavků

Vymezení požadavků kladených na systém provedené na základě metody „MoSCoW“, která stanoví hranice požadavků od nejdůležitějších k méně potřebným. Metoda je popsaná v teoretické časti v podkapitole 3.2.2 „Návrh požadavků na IS“ a obsahuje čtyři fáze ke stanovení požadavků. Na základě počáteční analýzy pomocí metody „MoSCoW“ vývoj informačního systému bude zaměřen především na ty části, které jsou nezbytné pro zprovoznění a alespoň k jeho částečnému fungování. Tyto části systému obsahují různorodé požadavky, které jsou uvedeny v tabulce níže. Požadavky určují vymezení hranic systému a jsou základem pro následné modelování diagramů.

Metoda „MoSCoW“			
„Must have“ *(Musí mít)*	„Should have“ *(Mělo by mít)*	„Could have“ *(Může mít)*	„Won’t have“ *(Není nutné)*
Přihlašování	Úprava účtu	Zobrazení seznamu objednávek	Posílání notifikací
Registraci	Změna hesla	Stanovení času objednávky	Měření návštěvnosti
Vybírání zboží	Úprava objednávky	Rušení účtu	Analýza objednávek
Rušení objednávky	Úprava položek	Vyhledávání podniků	
Zakládání objednávek			
Zakládání podniku			

Tabulka 1: Metoda „MoSCoW“, požadavky systému (Zdroj: Vlastní tvorba)

Funkční versus nefunkční požadavky

V následujících tabulkách se zobrazují informace s ohledem na funkční a nefunkční požadavky v závislosti na prioritách nutných k užívání a fungování systému. Metodické požadavky jsou uvedeny v závislosti na předpokládaných uživatelích systému. Mezi nefunkčními požadavky jsou uvedeny požadavky, které jsou méně potřebné k fungování systému, ale jsou také žádoucí pro jeho modifikace a zlepšení, které lze realizovat v dalších verzích systému. Vlastnosti systému jsou zpracované na základě metody „*FURPS*“, podle které jsou požadavkům přiřazeny jednotlivé identifikátory.

Funkční požadavky		
Aktér	Identifikátor	Požadavek
Zákazník	1) Z01 2) Z02 3) Z03 4) Z04	1) Systém umožní registraci novým uživatelům či přihlášení stávajícím. 2) Systém umožní nákup vybraných produktů online v rámci aplikace. 3) Systém umožní správu účtu neboli změnu osobních údajů či vymazaní účtu. 4) Sytém umožní zákazníkovi prohlédnutí historie objednávek či jejich úpravu nebo mazání.
Administrátor	1) A01 2) A02 3) A03	1) Systém umožní administrátorovi schvalování uživatelů neboli podniku a vlastníka 2) Systém umožní administrátorovi správu účtů neboli jejich schvalování a mazání. 3) Systém umožní správu objednávky.
Zaměstnanec	1) ZM01 2) ZM02	1) Systém umožní zaměstnanci zakládaní menu 2) Systém umožní správu a úpravu položek
Vlastník	1) V01 2) V02	1) Systém umožní prodej zboží podnikatele 2) Systém umožní úpravu a správu menu a jeho položek

Tabulka 2: Funkční požadavky (Zdroj: Vlastní tvorba)

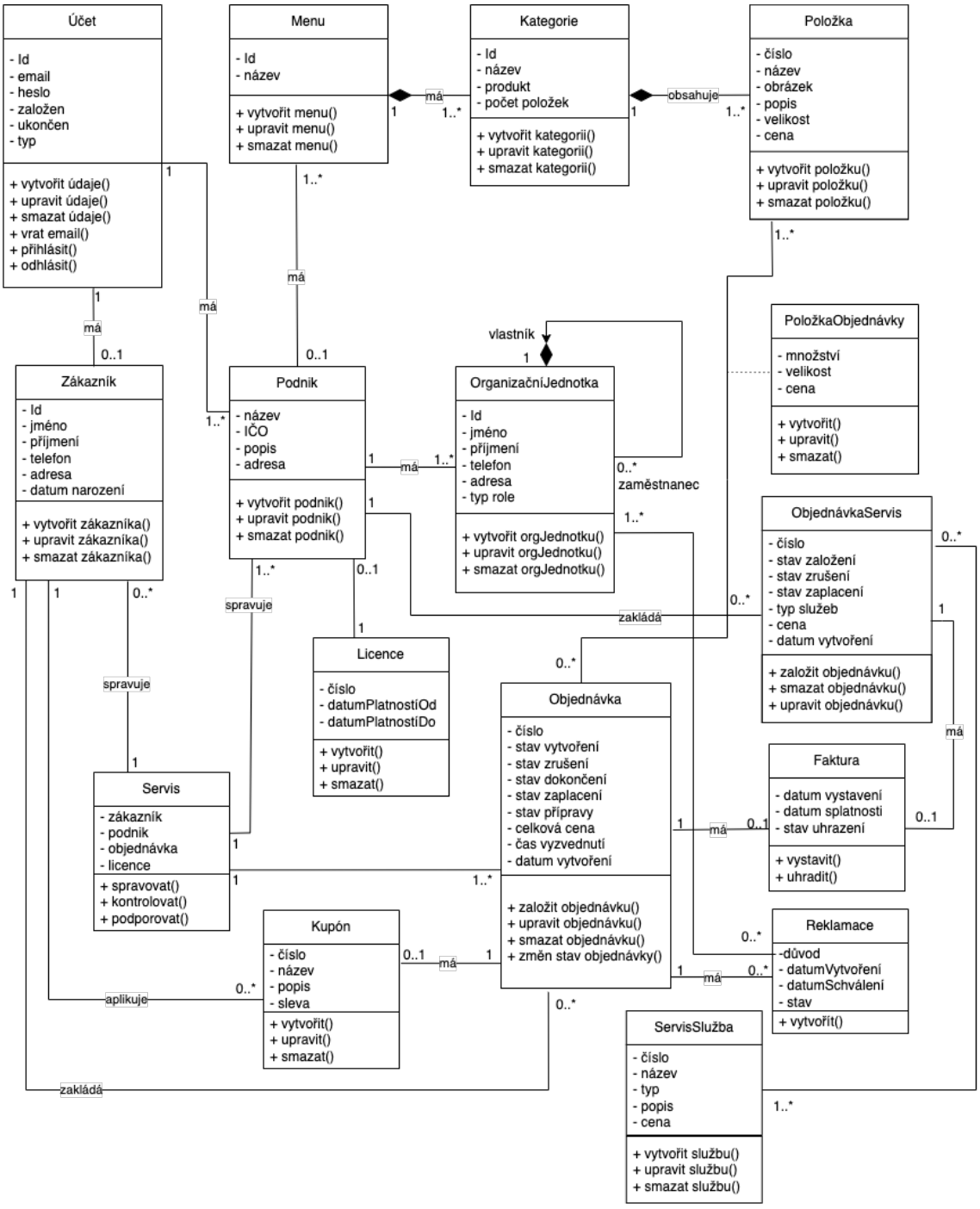
Nefunkční požadavky		
Kritéria	Identifikátor	Požadavek
Použitelnost	N01	Dodržena jednoduchost a srozumitelnost systému, snadná ovladatelnost jednotlivých prvků.
Spolehlivost	N02	Dostupnost systému pro uživatele 24 hodiny 7 dní
Účinnost	1) N03 2) N04 3) N05	1) Generování kávy na základě vlastních preferencí 2) Možnost výběrů velikostí produktů. 3) Zobrazení nejbližších kavárenských podniku v závislostí na lokaci uživatele.
Udržovatelnost	1) N06 2) N07	1) Kompatibilita systému s mobilním zařízením a jejich OS. 2) Snadná údržba.
Bezpečnost	N08	Systém vyžaduje aktualizace přihlašovacích údajů.

Tabulka 3: Nefunkční požadavky (Zdroj: Vlastní tvorba)

4.4 Modelování tříd

Modelování tříd je základem pro sestavení systému. Tento model zobrazuje statický pohled na systém, v rámci, kterého jednotlivé třídy představují objekty. Třídy obsahují metody, pomocí kterých je zajištěna komunikace mezi objekty.

Diagram tříd



Obrázek 17: Diagram tříd IS „Your takeaway“ (Zdroj: Vlastní tvorba)

Datový slovník

Účet – tato třída obsahuje informace o přihlašovacích údajích osoby neboli identifikačních údajích jako: email, heslo, datum založení, datum ukončení a typ účtu. Stav této třídy vyjadřuje, zda je *Zákazník či Podnik jsou* přihlášený či nikoliv.

Zákazník – jedná se o třídu představující uživatele systému ze strany zákazníka, uchovává si atributy jako id, jméno, příjmení, telefonní číslo, adresu uživatele, typ role a datum narození.

Podnik – třída obsahuje název podniku, popis činností podniku, adresu a identifikační číslo neboli IČO.

Faktura – obsahuje datum vystavení faktury, datum splatnosti a stav uhrazení.

Menu – třída obsahuje atribut název, identifikátor. Jinak menu obsahuje jednotlivé kategorie, které mají možnost zakládat zaměstnanci a vlastníci jednotlivých podniků.

Položka – třída popisuje název položky, obrázek položky a její popis, číslo, velikost a cenu. Položku lze vytvořit, upravit nebo smazat.

Objednávka – obsahem je stav, ve kterém se nachází objednávka a datum jejího vytvoření, číslo objednávky, celková cena a čas vyzvednutí. Objednávka mění svůj stav při založení, zaplacení či zrušení.

Kupón – obsahuje číslo, název kupónu jeho popis a celkovou slevu.

Servis – tato třída má atributy jako podnik, zákazník, objednávka, licence. Třída obsahuje metody, které jsou určeny ke správě, kontrole a podpoře.

Objednávka servisu – třída obsahuje stavy objednávky a je určena pro výběr různých servisů poskytovaných podnikatelské osobě a zároveň napojena na třídu *Faktura* za účelem platby poskytovaných služeb.

Servis služba – třída umožnuje vybrat typ poskytovaných služeb a obsahuje atributy jako číslo, název, typ, popis a cena.

Kategorie – obsahuje identifikátor, název a počet položek.

Položka objednávky – tato třída obsahuje velikost, množství a cenu objednávky a je využívaná k propojení *Položky* a *Objednávky*.

Organizační jednotka – na základě typu role popsané v rámci atributu této třídy je možné osobu zařadit do nadřazené jednotky neboli vlastníka či podřízené jednotky, což vyjadřuje, že zařazená osoba je zaměstnancem podniku.

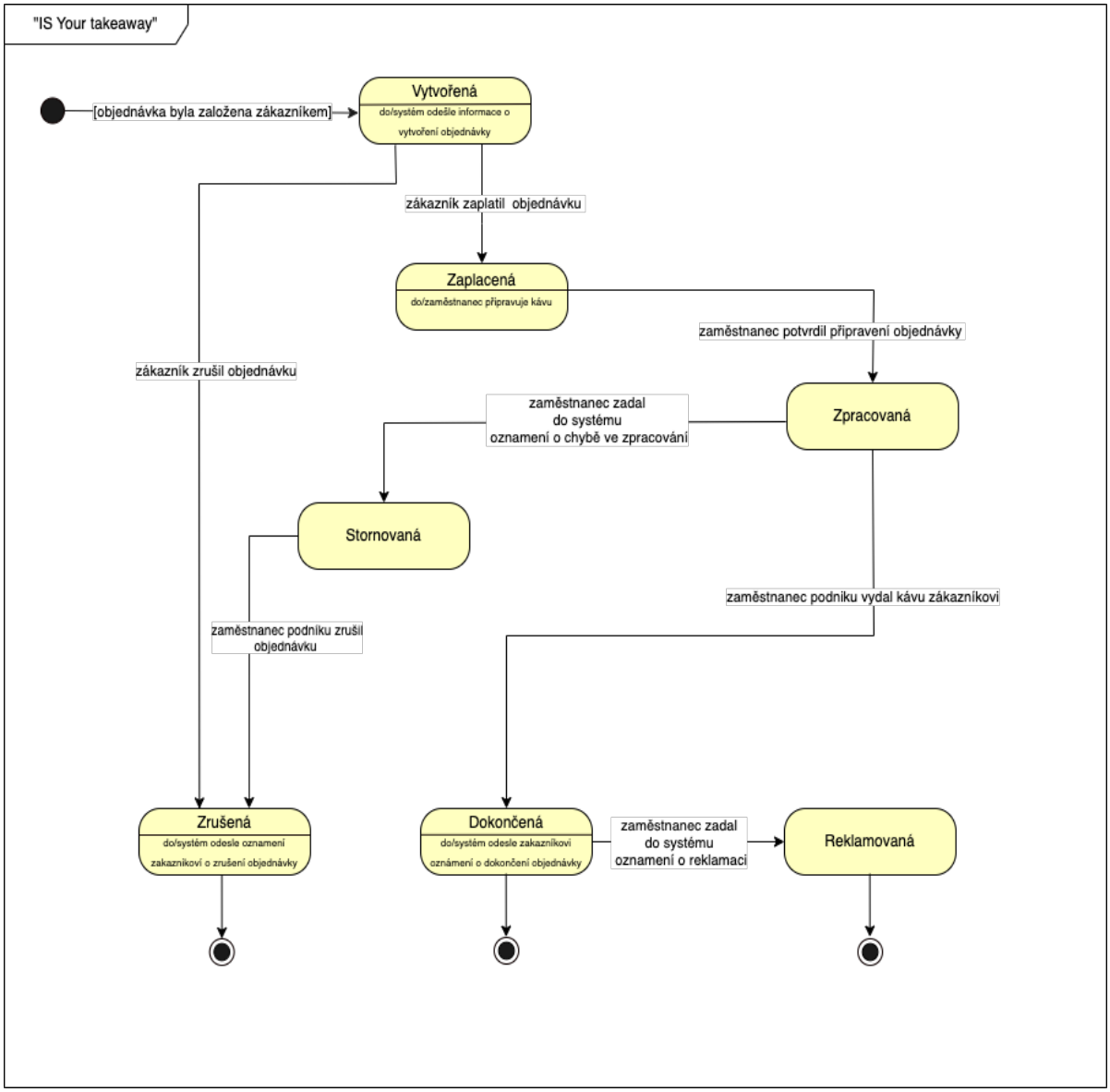
Reklamace – tato třída je určena pro reklamaci již provedené objednávky, obsahuje atributy, jako je datum vytvoření, datum schválení a důvod reklamace.

Licence – třída má atributy číslo licence, datum splatnosti od a do, umožnuje podniku využívat služby poskytnuté v rámci aplikací na základě licenční smlouvy.

4.5 Modelování stavů

V kapitole „Systémové modelovaní“ a v podkapitole 3.4.2 Model stavů byl již popsán význam stavových diagramů, ty jsou vytvářeny za účelem zobrazení změn tříd daného systému v čase. Je důležité zvážit před samotným návrhem, zda tento typ diagramu je vhodný pro návrh či nikoliv. V tomto případě byla vybrána třída „*Objednávka“*, která nabývá dalších stavů:

- *Vytvořená* – objednávka byla založena zákazníkem.
- *Zrušená* – zákazník nebo zaměstnanec zrušil objednávku.
- *Zaplacená* – zákazník zaplatil objednávku, platba byla potvrzená na základě provedené transakce platební bránou.
- *Zpracována* – objednávka byla zpracována zaměstnancem podniku.
- *Dokončená* – objednávka neboli káva byla vydána zaměstnancem podniku zákazníkovi.
- *Stornována* – chyba ve zpracování objednávky byla nahlášena zaměstnanci do systému.
- *Reklamovaná* – zaměstnanec zadal do systému oznámení o reklamaci.



Obrázek 18: Stavový diagram: Objednávka (Zdroj: Vlastní tvorba)

4.6 Modelování interakcí

Další diagram, již byl popsán v podkapitole 3.4.3 „Model interakce“, patří k diagramům jako:

- Případu užití – popisuje interakce systému s okolím.
- Sekvenční diagramy – zobrazují interakce objektů v rámci systému.
- Diagramy aktivit – zobrazují podrobnější popis činností neboli událostí v rámci systému.

4.6.1 Diagram případu užití

„Administrátor“ využívá systém pro registrace podnikatelů, jejich správu, schvalování žádostí o přidání zaměstnance v rámci jednotlivých podniků a správu uživatelů. Pro svou činnost musí být bezpodmínečně do systému přihlášen, po dokončení aktivit se zase odhlásí.

„Vlastník“ - jedná se o roli majitele určitého podniku, má podobná práva jako *„administrátor“*, zároveň má možnost spravovat menu a jeho položky. Má právo žádat *„administrátora“* o přidání role zaměstnance pro další osoby v rámci systému.

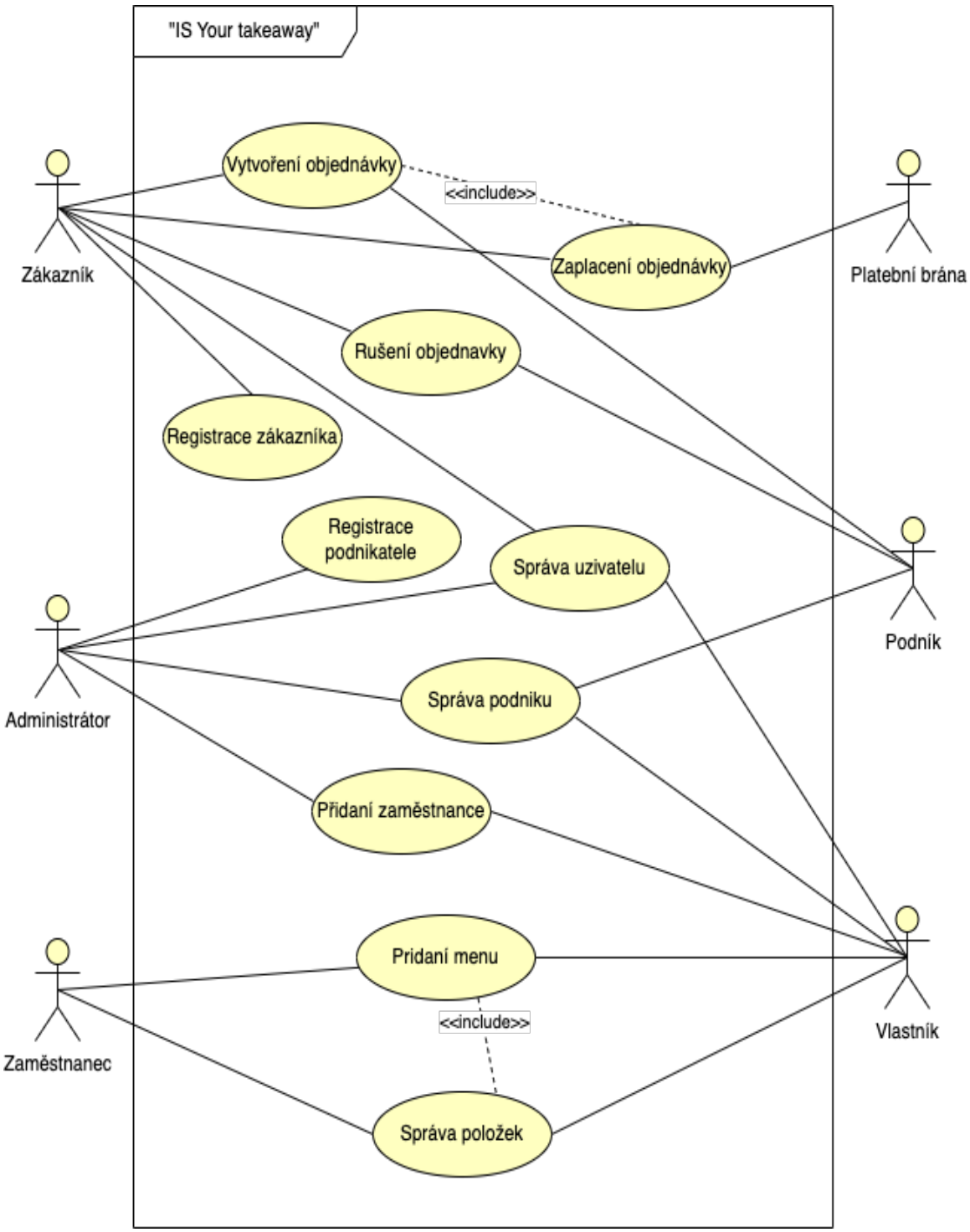
„Zaměstnanec“ - tato role připadá na zaměstnance podniku na základě schválení žádosti administrátorem systému, která byla podaná vlastníkem. Má možnost spravovat jednotlivá menu a jednotlivé položky.

„Zákazník“ - uživatel systému, který má právo vytvoření objednávky, dále má možnost placení a rušení. Zákazníkovi je přiděleno právo pro vytváření vlastního učtu v rámci registrací zákazníka a dalších jeho úprav.

„Podnik IS“ vidí výsledek založené objednávky a je spravován administrátorem či vlastníkem v rámci systému. Podnik řídí celkové zpracovaní objednávky.

„Platební brána“ přijímá výsledek placení proběhlého procesu objednávky.

Obrázek níže znázorňuje „use case“ diagram se základními funkcionalitami systému, jenž mohou aktéři využívat.



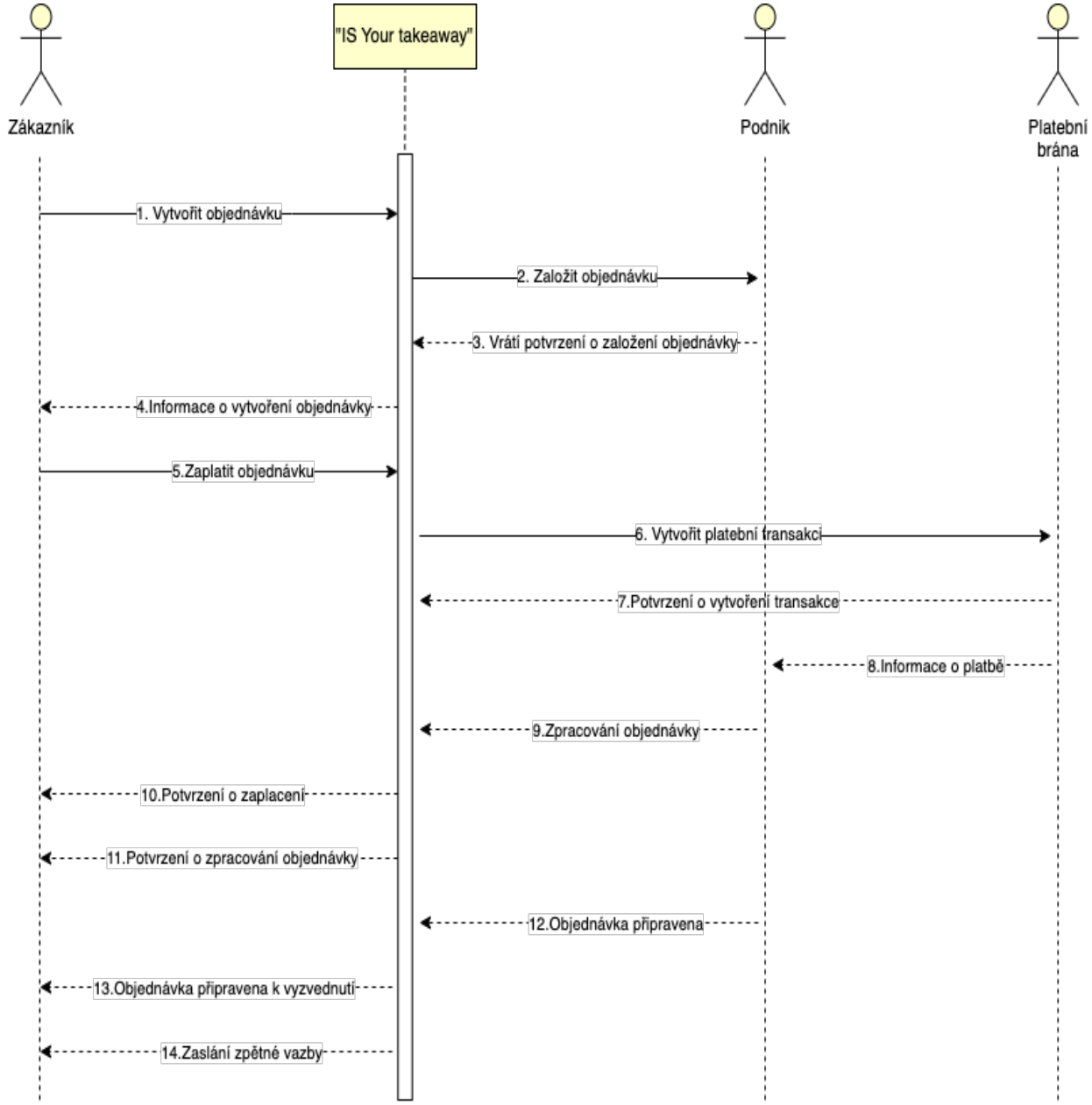
Obrázek 19: Diagram případů užití (Zdroj: Vlastní tvorba)

4.6.2 Sekvenční diagramy

Sekvenční diagramy obsahuje jednotlivé scénáře a jejich grafické znázorněný.

Vytvoření objednávky	
Aktéři	Zákazník Podnik Platební brána
Podmínky pro spuštění	Zákazník je přihlášen do systému.
Základní scénář	1. Zákazník vytvoří objednávku. 2. Systém pošle podniku požadavek pro založení objednávky. 3. Podnik vrací potvrzení o založení objednávky. 4. Systém vrací zákazníkovi informace o vytvoření objednávky. 5. Zákazník platí objednávku. 6. Systém posílá požadavek o vytvoření platební transakce na platební bránu. 7. Platební brána vrací potvrzení o vytvoření transakce. 8. Platební brána zasílá podniku informace o platbě. 9. Podnik zasílá systému potvrzení o zpracování objednávky. 10. Systém zasílá potvrzení zákazníkovi o zaplacení. 11. Systém posílá zákazníkovi informace o zpracování objednávky. 12. Podnik zasílá systému informaci o připravené objednávce. 13. Systém zasílá zákazníkovi informaci, že objednávka je připravena k vyzvednutí. 14. Systém zasílá zákazníkovi požadavek k zaslání zpětné vazby.
Alternativní scénář	Objednávka je zrušena po vytvoření.
Podmínky ukončení	Nová objednávka je vytvořena.

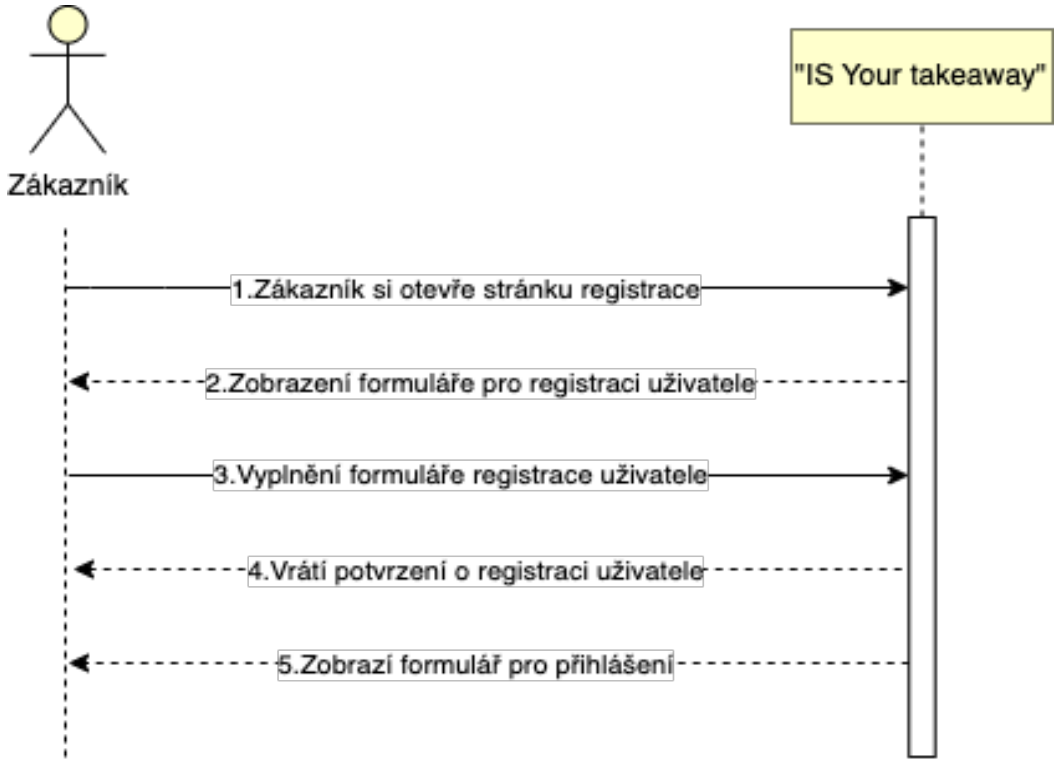
Tabulka 4: Scénář vytvoření objednávky (Zdroj: Vlastní tvorba)



Obrázek 20: Sekvenční diagram: Vytvoření objednávky (Zdroj: Vlastní tvorba)

Registrace zákazníka	
Aktéři	Zákazník
Podmínky pro spuštění	Zákazník neexistuje v systému
Základní scénář	1. Zákazník si otevře stránku registrace. 2. Systém zobrazí formulář pro registraci uživatele. 3. Zákazník vyplní formulář o registraci uživatele. 4. Systém vrací potvrzení o registraci uživatele. 5. Systém zobrazí formulář zákazníkovi pro přihlášení.
Alternativní scénář	Nebyly správně vyplněné údaje pro registraci. Nejsou vyplněny všechny hodnoty položek.
Podmínky ukončení	Nová položka je vytvořena.

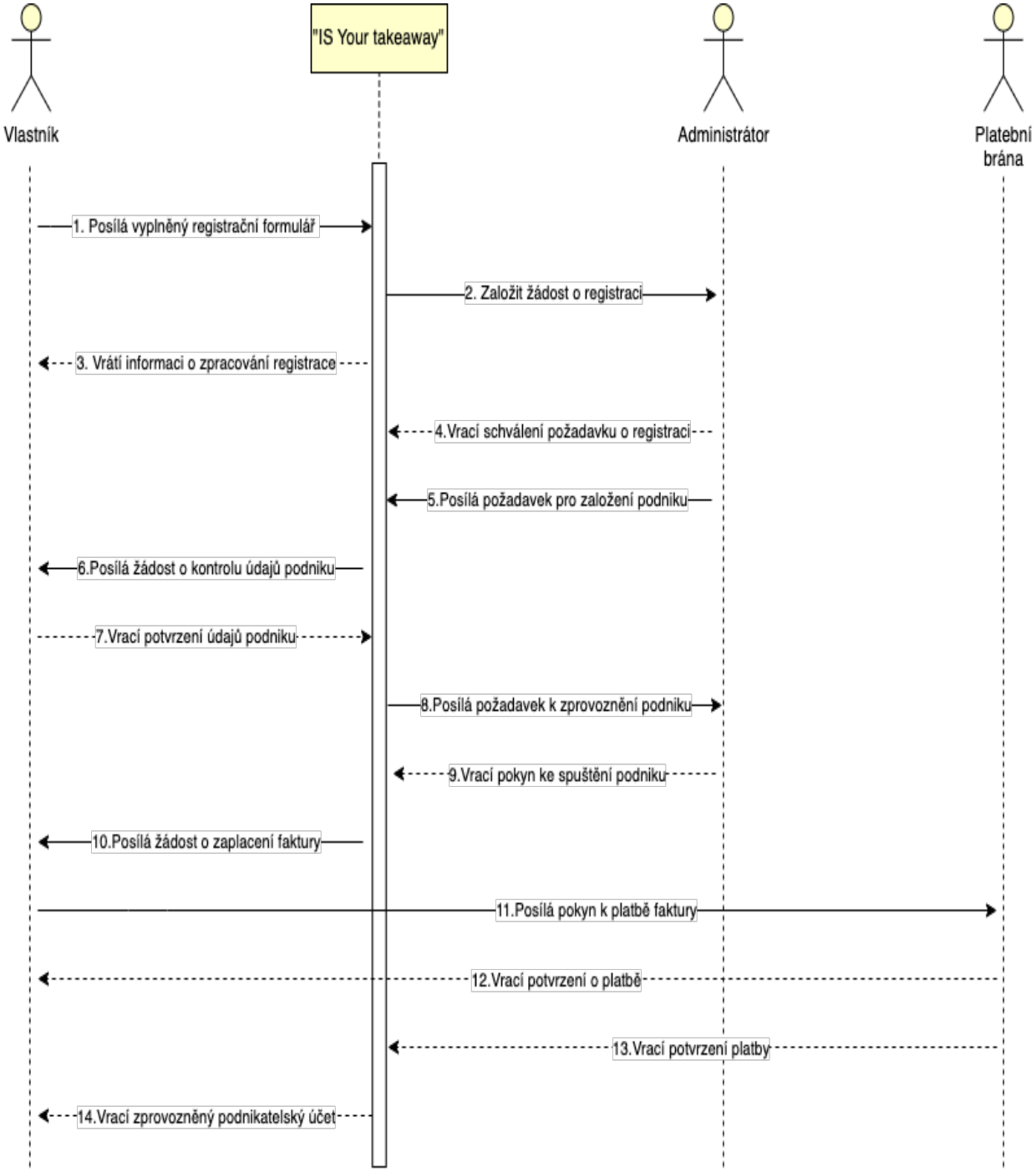
Tabulka 5: Scénář přidání zákazníka (Zdroj: Vlastní tvorba)



Obrázek 21: Sekvenční diagram: Registrace zákazníka (Zdroj: Vlastní tvorba)

Registrace podnikatele	
Aktéři	Vlastník Administrátor Platební brána Systém
Podmínky pro spuštění	Podnikatel neexistuje v systému
Základní scénář	1. Vlastník posílá vyplněný registrační formulář. 2. Systém zakládá žádost o registraci podniku. 3. Systém vrací vlastníkovi informaci o zpracování registrace. 4. Administrátor vrací systému schválení požadavku o registraci. 5. Administrátor posílá systému požadavek o založení podniku. 6. Systém posílá žádost o kontrolu údajů podniku vlastníkovi. 7. Vlastník vrací systému potvrzení údajů podniku. 8. Systém posílá administrátorovi požadavek ke zprovoznění podniku. 9. Administrátor vrací systému pokyn ke spuštění podniku. 10. Systém posílá žádost vlastníkovi o zaplacení faktury. 11. Vlastník posílá platebně bráně pokyn k platbě faktury. 12. Platební brána vrací vlastníkovi potvrzení o platbě. 13. Platební brána vrací systému potvrzení o platbě. 14. Systém vrací vlastníkovi plně zprovozněný podnikatelský účet.
Alternativní scénář	Nebyly správně vyplněny údaje pro registraci. Nejsou vyplněny všechny hodnoty položek.
Podmínky ukončení	Nová položka je vytvořena.

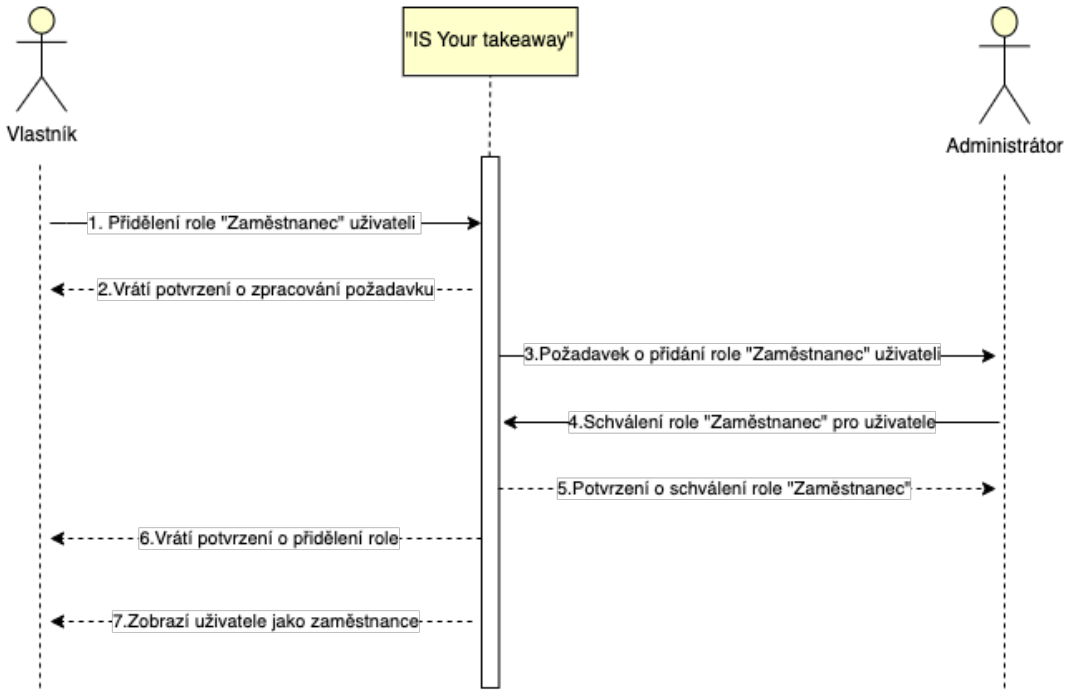
Tabulka 6: Scénář registrace podnikatele (Zdroj: Vlastní tvorba)



Obrázek 22: Sekvenční diagram: Registrace podnikatele (Zdroj: Vlastní tvorba)

Přidání zaměstnance	
Aktéři	Vlastník Administrátor
Podmínky pro spuštění	Vlastník a administrátor jsou přihlášeni do systému.
Základní scénář	1. Vlastník požádá systém o přidělení role „Zaměstnanec“ uživateli. 2. Systém vrací potvrzení o zpracování požadavku. 3. Systém posílá požadavek o přidání role administrátorovi. 4. Administrátor schvaluje role potřebné pro uživatele. 5. Systém zasílá potvrzení o schválení role administrátorovi. 6. Systém vrací potvrzení o přidělení role vlastníkovi. 7. Systém zobrazí uživatele jako zaměstnance vlastníkovi.
Alternativní scénář	Nebyly správně vyplněny údaje uživatele. Administrátor neschválí roli. Nejsou vyplněny všechny hodnoty.
Podmínky ukončení	Nový zaměstnanec je přidán.

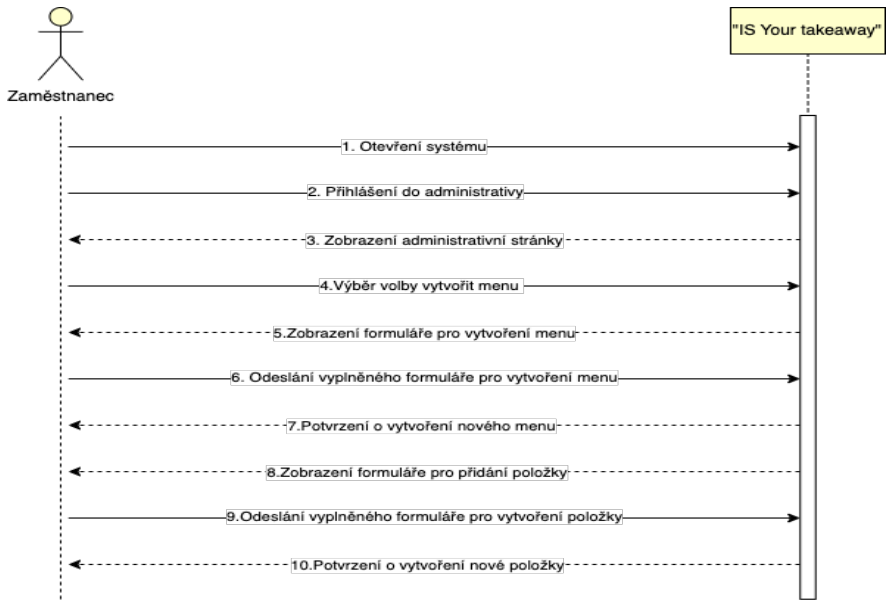
Tabulka 7: Scénář přidání zaměstnance (Zdroj: Vlastní tvorba)



Obrázek 23: Sekvenční diagram: Přidaní zaměstnance (Zdroj: Vlastní tvorba)

Přidání menu	
Aktéři	Zaměstnanec
Podmínky pro spuštění	Zaměstnanec je přihlášen do systému.
Základní scénář	1. Otevření systému zaměstnancem. 2. Zaměstnanec se přihlásí do administrativy. 3. Systém zobrazí administrativní stránku. 4. Zaměstnanec si vybere volbu - vytvořit menu. 5. Systém zobrazí formulář pro vytvoření menu. 6. Zaměstnanec odešle vyplněný formulář pro vytvoření menu. 7. Systém vrací potvrzení o vytvoření nového menu. 8. Systém zobrazí formulář pro přidání položky. 9. Zaměstnanec odešle vyplněný formulář pro vytvoření položky. 10. Systém zobrazí potvrzení o vytvoření nové položky.
Alternativní scénář	Nebyly správně vyplněny přihlašovací údaje. Nejsou vyplněny všechny údaje.
Podmínky ukončení	Nové menu je vytvořeno.

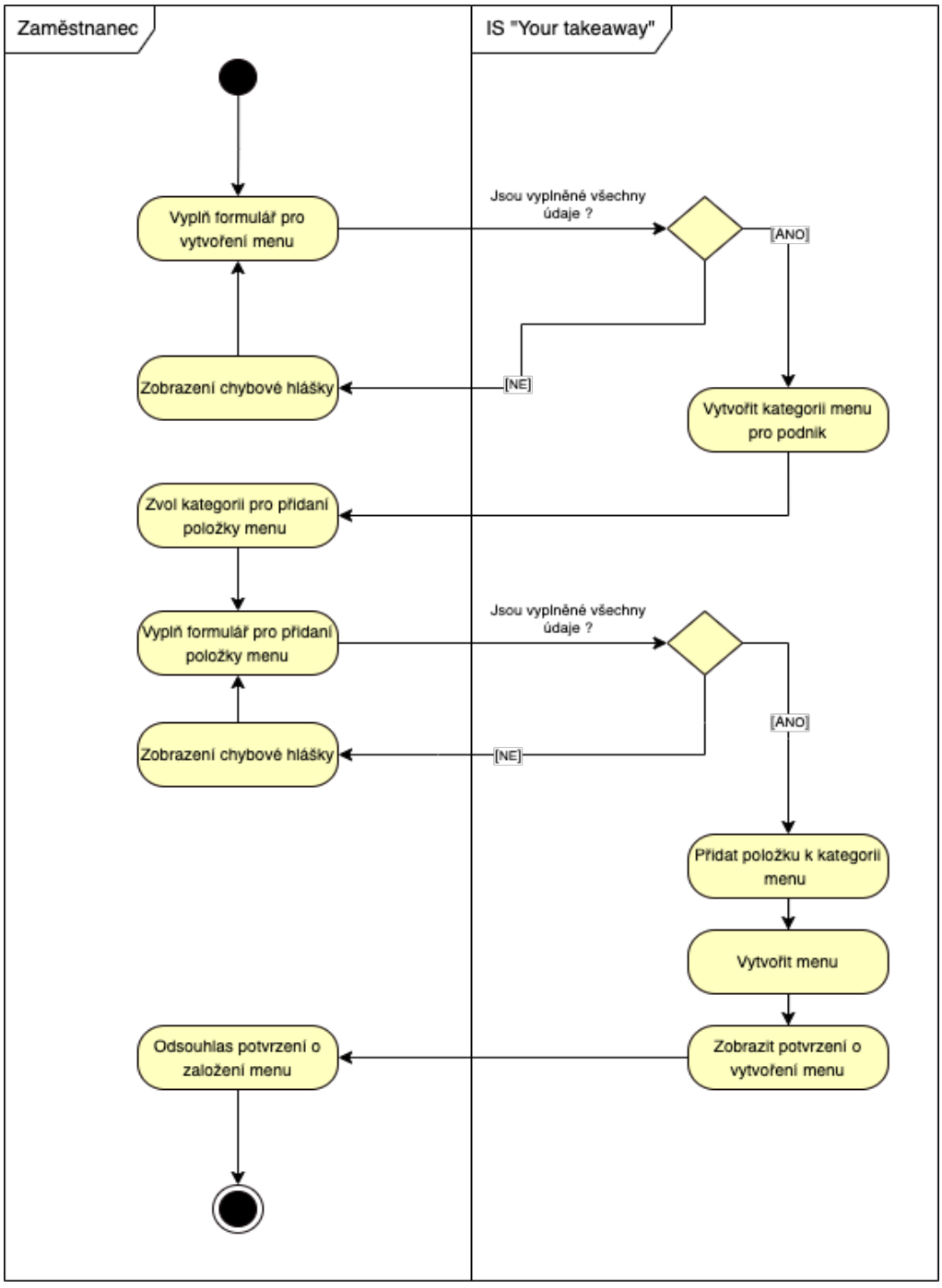
Tabulka 8: Scénář přidaní menu (Zdroj: Vlastní tvorba)



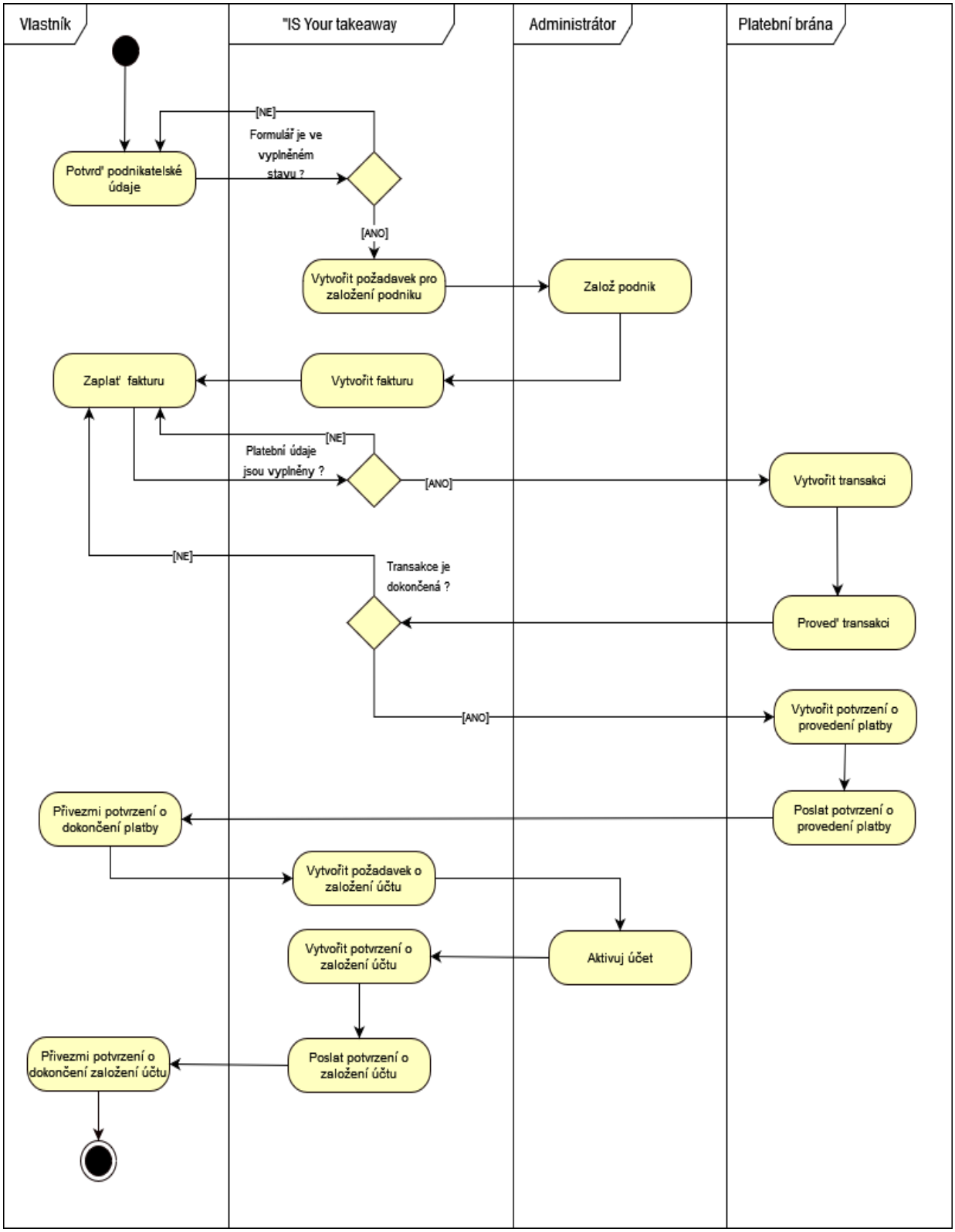
Obrázek 24: Sekvenční diagram: Přidání menu (Zdroj: Vlastní tvorba)

4.6.3 Modelování aktivit

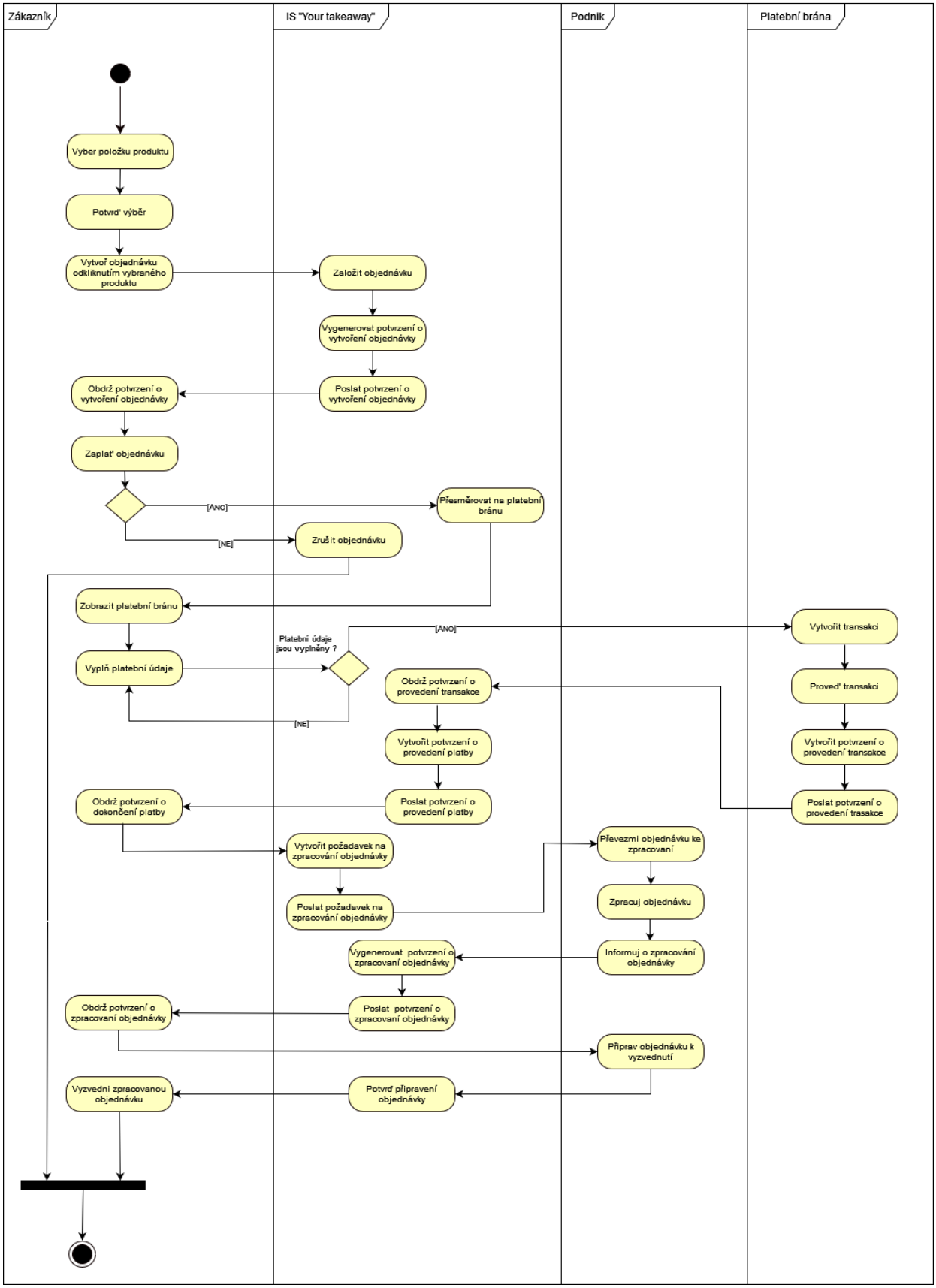
Modelování aktivit je součásti modelování interakce a je popsáno v teoretické části. Tento typ digramů znázorňuje alternativní scénáře a vybrané procesy, a to vytvoření objednávky zákazníkem, přidávání menu zaměstnancem, registrace zaměstnance majitelem, registrace podnikatele, zákazníka.



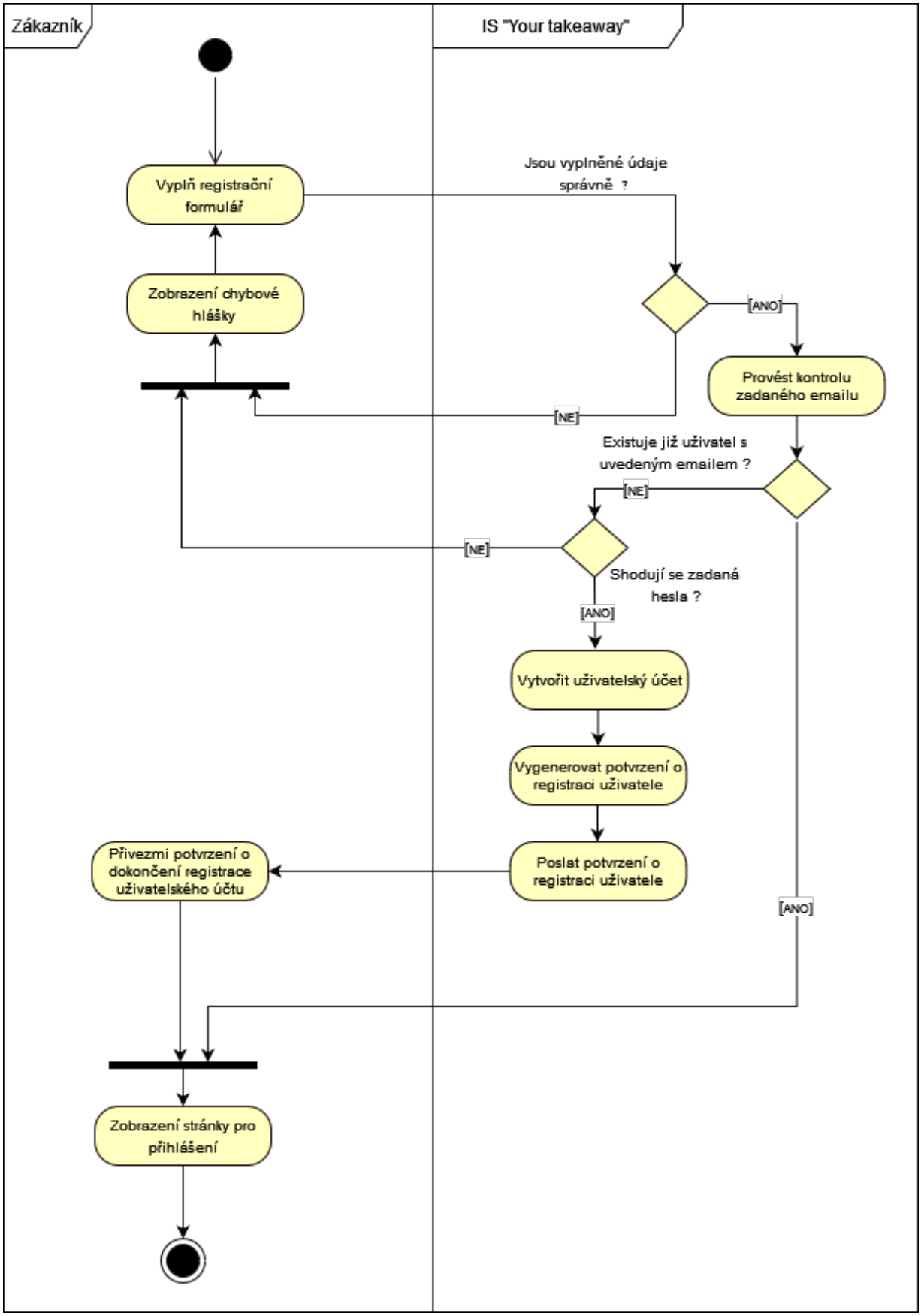
Obrázek 25: Aktivity diagram: Přidání menu (Zdroj: Vlastní tvorba)



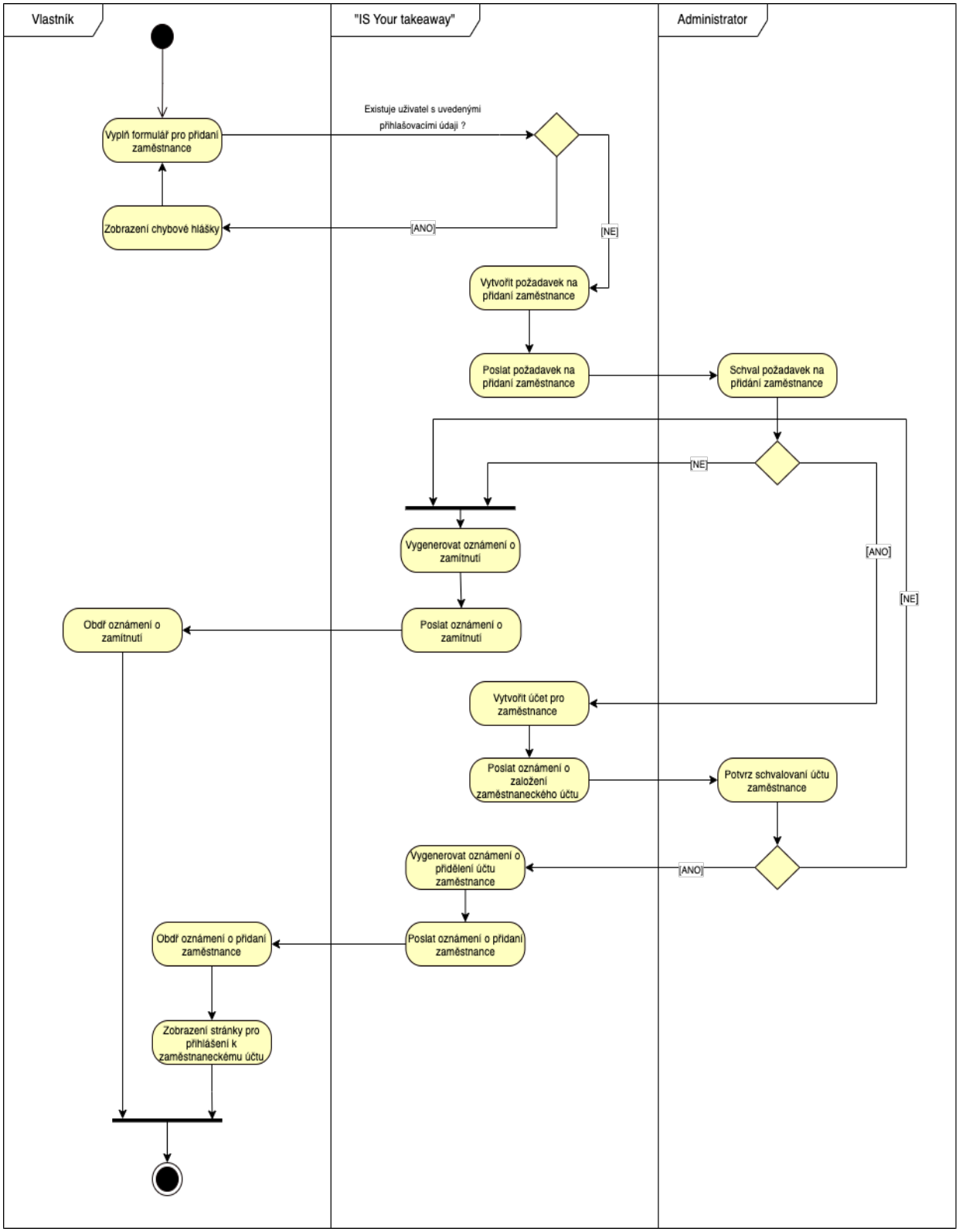
Obrázek 26: Aktivity diagram: Registrace podnikatele (Zdroj: Vlastní tvorba)



Obrázek 27: Aktivity diagram: Vytvoření objednávky (Zdroj: Vlastní tvorba)



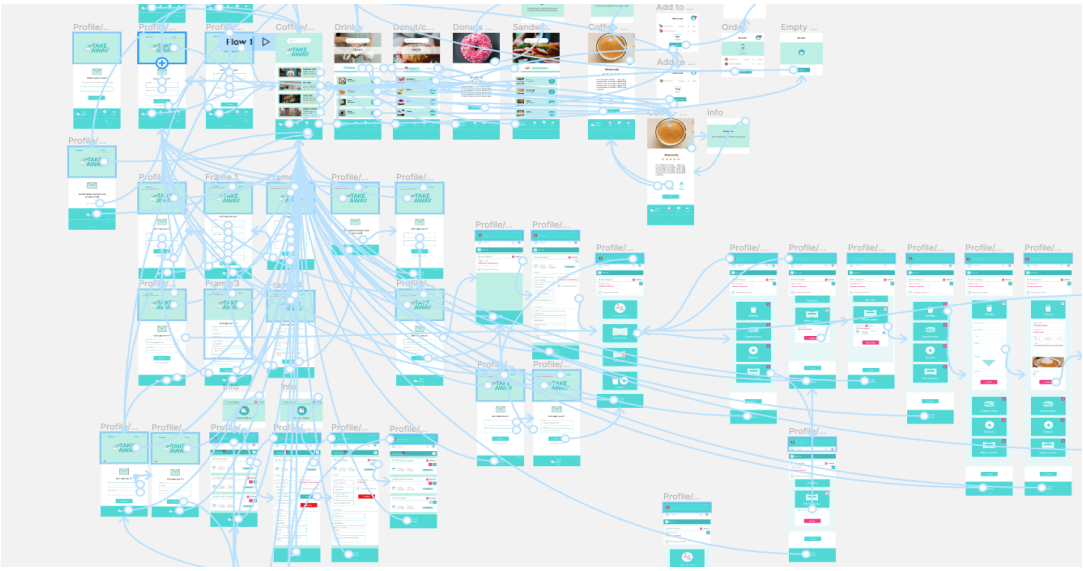
Obrázek 28: Aktivity diagram: Registrace zákazníka (Zdroj: Vlastní tvorba



Obrázek 29: Aktivity diagram: Registrace zaměstnance (Zdroj: Vlastní tvorba)

4.7 Prototyp IS

Návrh informačního systému je doplněn grafickým prototypem, který znázorňuje základní průchody systému, a je realizován za pomoci nástroje Figma, který umožnil zobrazení chování systému koncovým uživatelům. Technika wireframe nebyla využita z důvodu nemožnosti prezentace úplného náhled na systém a jeho základních principů fungování jednotlivých komponent. Uživatelské rozhrání systému je zaměřeno na jeho snadné a jednoduché ovládání jak ze strany uživatelů, tak i ze strany koncových uživatelů neboli zákazníků. Vzhled systému pro zákazníky byl základem k vývoji mobilní verze aplikace, v rámci, které je možné přihlášení do systému a výběr produktů určených k nákupu. Návrh obsahuje řešení pouze pro mobilní zařízení z důvodu a záměru souvisejících se způsobem využití, které již bylo popsáno výše, verze pro desktop může být vypracovaná dále podle potřeby. Neobvyklé řešení grafického návrhu je provedeno na základě UI/UX technologie a je orientováno na přilákání většího množství zákazníků a zvýšení produktivity. Řešení obsahuje vzhled systému určený jak pro zákazníky, tak i pro podnikatele a uživatele administrátorské části systému.



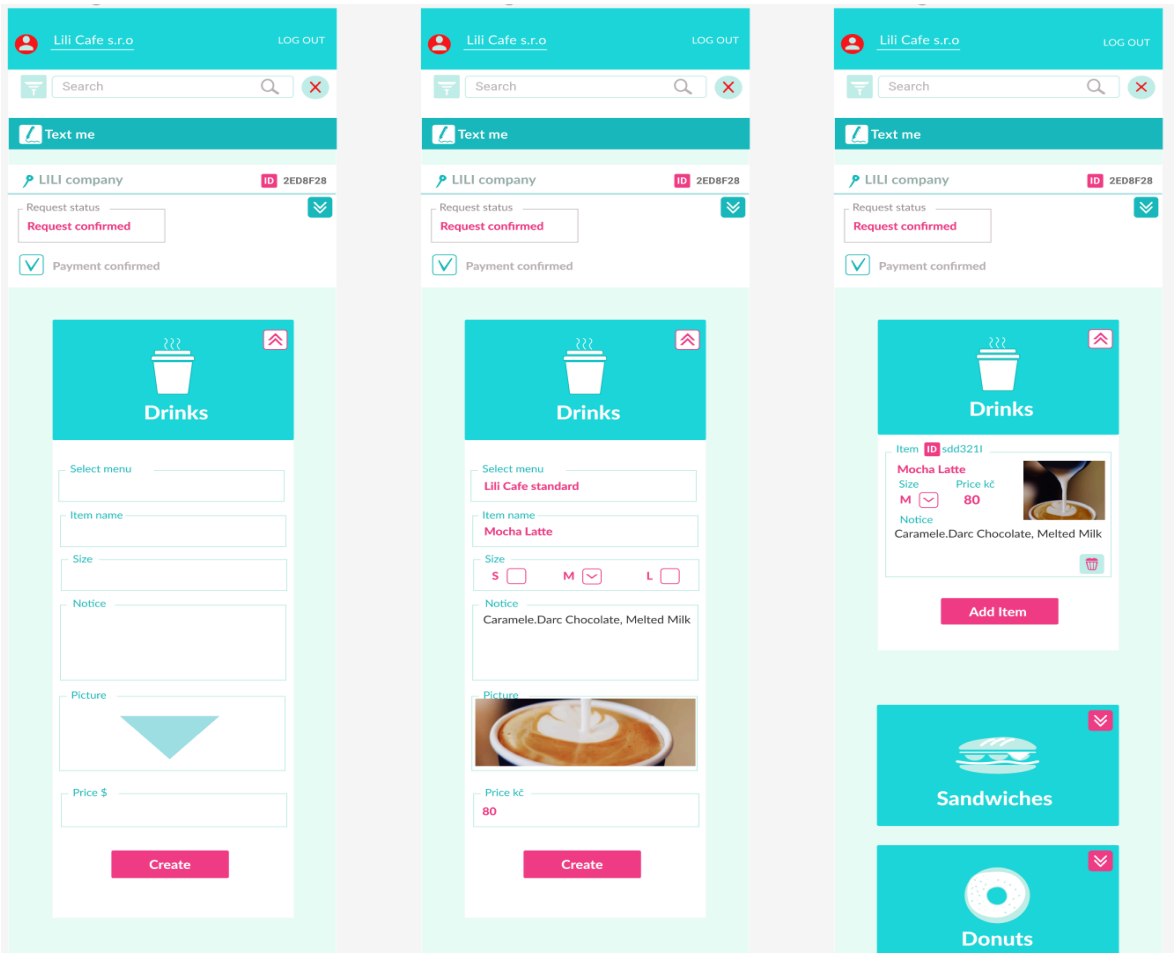
Obrázek 30: Tvorba prototypu, Figma (Zdroj: Vlastní tvorba)

Podkladem k jeho využití jsou scénáře, které jsou dostupné v příloze, a jeho celkový náhled je zpřístupněn z následujícího odkazu: https://www.figma.com/proto/SjHAqcP1Us2k9un6vxo4EF/Prototype-Your-takeaway?node-id=1-1568&scaling=min-zoom&page-id=0%3A1&starting-point-node-id=1%3A1568

5 Výsledky a diskuse

Na základě provedených analýz byly vymezené hranice navrhovaného systému a stanovené požadavky nutné k jeho fungování. Navržený systém posloužil jako východisko pro vytvoření prototypu, který znázorňuje jednoduchost a srozumitelnost jeho fungování. Jednou z nejdůležitějších částí systému, která je určena pro prodej nápojů a občerstvení zákazníkovi, byla realizovaná v rámci mobilní aplikace.

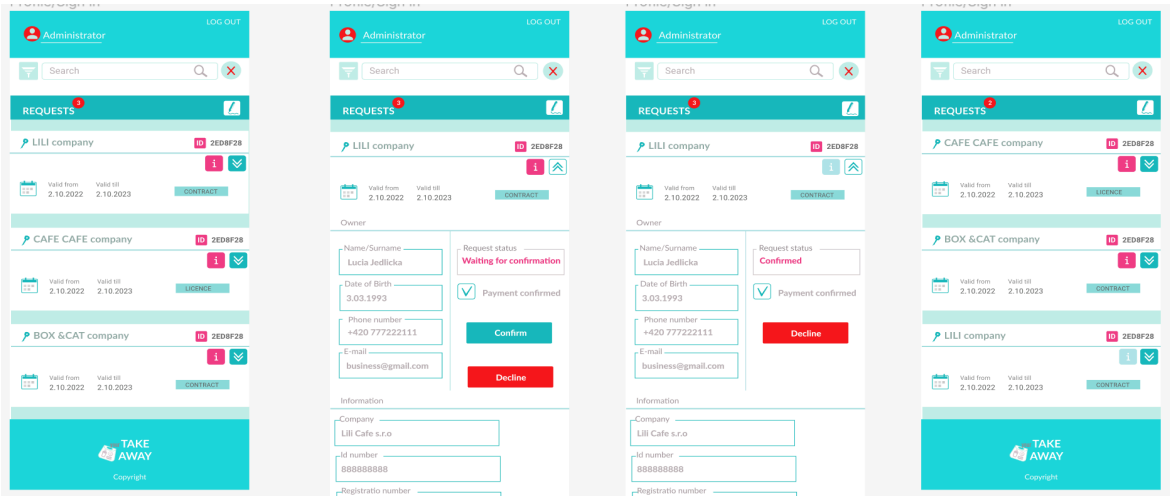
Část informačního systému, která je určena pro podnikatele, obsahuje uživatelské rozhraní a umožnuje registraci podnikatele a dalších pracovníků podniku, dále jejich přihlašování do systému, správu účtu podle role nadřízený a podřízený. Nadřízená osoba má možnost nejen správy objednávek, ale registrace dalších pracovníků podniku. Prototyp systému znázorňuje možnost, zakládání menu a jemu příslušných položek.



Obrázek 31: Prototyp: Zakládání menu, Figma (Zdroj: Vlastní tvorba)

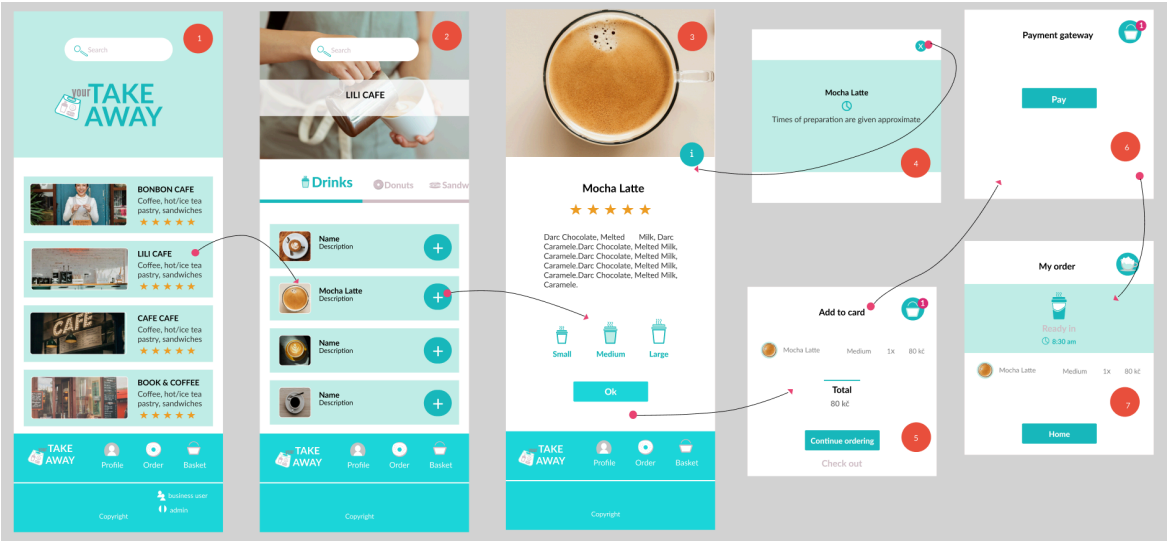
Administrátorská stránka systému je představena přehledem podnikatelských účtů, znázorňuje žádosti prodejce o založení účtu a schvalování jejich pracovníků, zároveň

obsahuje informace o provedených platbách za poskytnutý servis. Přihlášení do systému je možné na základě role administrátora a je realizováno z jiné přihlašovací stránky na rozdíl od dalších uživatelů.



Obrázek 32: Prototyp: Schvalování podnikatelského účtu, Figma (Zdroj: Vlastní tvorba)

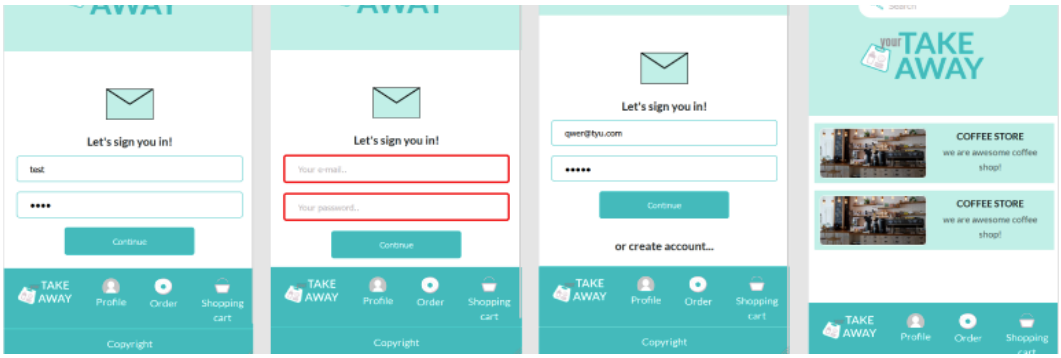
Uživatelské rozhraní určené pro zákazníka je představené hlavní stránkou s širokou paletou výběru z různorodých podniků, které nabízí své občerstvení a kávu. Po výběru podniku se zákazníkovi zobrazí menu a jeho kategorie, po volbě jedné z nich zákazníkovi systém zobrazí jednotlivé položky s obsahem nápojů či občerstvení. Po zvolení produktu se zákazníkovi systému zobrazí další informace o prodávaném produktu, například složení nebo obsah alergenů. Dále po založení objednávky systém uživatele přesměruje na platební bránu a po zaplacení zákazníkovi zobrazí informace o jeho objednávce. Tento průchod systému je zobrazen na následném obrázku.



Obrázek 33: Prototyp: Vytvoření objednávky zákazníkem, Figma (Zdroj: Vlastní tvorba)

Realizace informačního systému a jeho vývoj je samozřejmě komplexnější a vyžaduje však kompletní návrh systému a jistou zkušenost s realizací projektu a jeho nasazování do produkce. Vývoj systému není tématem této diplomové práce, ale jak bylo zmíněno výše, za pomoci nasbíraných a nabytých poznatků během studia byla vytvořena mobilní aplikace, která je představena jednou z častí systému, která je určena k nákupu nápojů a občerstvení zákazníkem. Mobilní aplikace byla vyvíjena s využitím technologií HTML, CSS, Javascript a dalších popsaných v kapitole 4 (Vlastní práce). Řešení umožnuje uživateli se přihlásit do systému a ověřit správnost zadávaných přihlašovacích údajů, po přihlášení si uživatel může vybrat kavárnu a její nabízený produkt k nákupu.

Kód aplikace je zpřístupněn v příloze z odkazu na GitHubu.



Obrázek 34: Mobilní aplikace: Validace, přihlášení uživatele (Zdroj: Vlastní tvorba)

Navržený systém je aplikovatelný a účinný, vynikne svou jednoduchostí, použitelností a moderním vzhledem. Systém je rozsáhlý a v současnosti nepokrývá veškeré potřebné procesy a události, což by mělo být záměrem k rozvinutí stávajícího návrhu.

Rozšíření systému by se mělo zaměřit na vývoj dalších funkčních požadavků jako zobrazení kavárenských podniků podle geolokace uživatele, pak upozornění na určitý čas připravení objednávky, umožnit zákazníkovi zobrazit přehled historie objednávek, přidávání recenzí a vlastních názorů. Dále je relevantní rozšířit systém o funkce „Káva podle mého gusta“, která by umožnila nákup kávy na základě vlastních požadavků, kde kupující by si mohl zvolit potřebné ingredience, jejich množství a druh. Poté vygenerované požadavky poslouží jako podklad pro objednávku originální kávy a systém zobrazí podniky, které tuto kávu po objednání zpracují. Podnikatelskou část systému je potřeba rozšířit o funkce, které sledují množství prodávaného zboží a upozorňují jak podnikatele, tak zákazníka na omezené množství daného produktu atd.

6 Závěr

Cílem této diplomové práce bylo navrhnout informační systém, který je zaměřen na prodej nápojů a občerstvení různými podniky. Návrh systému byl realizován za pomocí modelovacího jazyka UML, zároveň bylo provedeno vymezení hranic systému a stanovení požadavků nutných k jeho fungování. Výstupem provedených analýz je návrh jednotlivých UML modelů, které definují soubor nezbytných vlastností a funkce systému.

Pro lepší znázornění navrženého informačního systému byl vytvořen prototyp s důrazem na jednoduchost a snadnou použitelnost. Na základě vlastních poznatků a zkušeností byl návrh prototypu doplněn vlastním atraktivním designem za pomocí UX/UI technologie a nástroje Figma.

Systém umožnuje registraci a přihlášení podle typů uživatelů, poté v závislosti na předem stanovených požadavcích obsahuje funkce, které jsou zaměřené na zefektivnění pracovních procesů určených k založení a zpracování objednávky. Dále je systémem umožněno zakládání podnikatelského účtu, přihlášení zaměstnanců nadřízenou osobou a zakládání menu jednotlivých kategorií a položek v rámci nabídky cílené na online prodej. Systém zároveň obsahuje administrátorskou stránku, v rámci které probíhají procesy schvalování podnikatelských účtů.

Jeden z nejdůležitějších prvků systému je určený k prodeji poptávaného produktu, byl realizován v rámci mobilní aplikace za pomoci využití technologií HTML, CSS, JavaScript, Spring Boot, Maven a MySql.

Výsledkem práce je návrh informačního systému, který má předpoklady k uspokojení potřeb uživatelů, jak zákazníků, tak podnikatelů. Systém nabízí zákazníkovi velký výběr kavárenských podniků a podnikatelům dává možnost přilákat větší množství zákazníků.

7 Seznam použitých zdrojů

Tištěné zdroje:

[1] *Aristotelés. Metafyzika. 4. Rezek, 2021. ISBN 978-80-86027-45-6.*

[2] *BLACKSTONE JR., John H., ed. APICS Dictionary. 14th Edition. Chicago: APICS - The Association for Operations Management, 2008. ISBN 1558221999.*

[3] *PROCHÁZKA, JAROSLAV a CYRIL KLIMEŠ. SOFTWAROVÉ INŽENÝRSTVÍ. 2., aktualiz. a dopl. vyd. Ostrava: Ostravská univerzita v Ostravě, 2009.*

[4] *VYMĚTAL, Dominik. Informační systémy v podnicích: teorie a praxe projektování. 1. vyd. Praha: Grada, 2009. Průvodce (Grada). ISBN 9788024730462*

[5] *JOSEF, Basl a Blažíček ROMAN. Podnikové informační systémy: Podnik v informační společnosti. 3., aktualizované a doplněné vydání. Praha: Management v informační společnosti. Grada, 2012. ISBN 978 -80-247-4307-3.*

[6] *Избачков, Телина а Петров. Информационные системы: Учебник для вузов. 3-е издание. Питер, 2011. ISBN 978-5-49807-158-9.*

[7] *Moderní metody řízení informačních systémů; Zdeněk Molnár - kapitola 4.4 (str. 58) Řízení projektů informačních systémů; Petr Doucek – kapitola 6 (str. 67)*

[8] *BOEHM, Barry W. A Spiral Model of Software Development and Enhancement. TRW Defense Syst. Group, Redondo Beach, CA, roč. 21, č. 5, August 1988. ISSN: 0018-9162*

[9] *BOEHM, Barry W. Software engineering: Barry W. Boehm's lifetime contributions to software. Wiley-IEEE Computer Society Press. 2007. 832 s. ISBN: 978-0-470-14873-0*

[10] *ARLOW, Jim a Ila NEUSTADT. UML 2 a unifikovaný proces vývoje aplikací: objektově orientovaná analýza a návrh prakticky. 2., aktualiz. a dopl. vyd. Brno: Computer Press, 2007. ISBN 9788025115039.*

[11] *ЛАРМАН, Крэг. Применение UML 2.0 и шаблонов проектирования. 3-е изд. ВИЛЬЯМС, 2013. ISBN 978-5-907144-36-1.*

[12] *BRUCKNER, Tomáš. Tvorba informačních systémů: principy, metodiky, architektury. Praha: Grada, 2012. Management v informační společnosti. ISBN 9788024741536.*

[13] *KANISOVÁ, Hana a Miroslav MÜLLER. UML srozumitelně. 2., aktualiz. vyd. Brno: Computer Press, 2006. ISBN 8025110834*

[14] *ROUDENSKÝ, Petr a Anna HAVLÍČKOVÁ. Řízení kvality softwaru: průvodce testováním. Brno: Computer Press, 2013. ISBN ISBN 978-80-251-3816-8*

[15] *Václav. Procesně řízená organizace. Praha: Grada, 2012. Management v informační společnosti. ISBN 9788024741284.*

[16] *VRANA, Ivan a Karel RICHTA. Zásady a postupy zavádění podnikových informačních systémů: praktická příručka pro podnikové manažery. Praha: Grada, 2005. Management v informační společnosti. ISBN 8024711036.*

Elektronické zdroje:

[17] *Pojem informační systém (IS): základní pojmy a definice. Etapy vývoje IS. [online].[cit.2022-0924].Dostupnéz:https://cde.osu.ru/demoversion/course157/text/1.5.html*

[18] *Knowledge Management Tools: Defining Data, Information, and Knowledge [online]. [cit. 2018-07-23]. Dostupné z: http://www.knowledge-management-tools.net/knowledge-information-data.html*

[19] *INFORMACE [online]. [cit. 20230]. Dostupné z: https://moodle.sspbrno.cz/pluginfile.php/9834/mod_resource/content/4/01_INS_Informace.pdf*

[20] *The Data-Information-Knowledge Cycle: By Michael Brackett [online]. November 14, 2013 [cit. 2022-09-28]. Dostupné z: https://www.dataversity.net/the-data-information-knowledge-cycle/#*

[21] *Úvod do informačních systémů [online]. 2018 [cit. 2022-09-28]. Dostupné z: https://samara.mgpu.ru/~dzhadzha/dis/15/120.html*

[22] *What is an Electronic Document Management System (EDMS)?: by Joe Byrne [online]. 2022 [cit. 2022-09-28]. Dostupné z: https://www.cognidox.com/blog/what-is-electronic-document-management-system*

[23] *Waterfall Model: tutorials point simply easy learning [online]. 2022 [cit. 2022-10-01]. Dostupné z: https://www.tutorialspoint.com/sdlc/sdlc_waterfall_model.htm*

[24] *System Development Life Cycle: tutorials point simply easy learning [online]. INDIA: Kavuri Hills, Madhapur, Hyderabad, Telangana - 500081, 2022 [cit. 2022-10-01]. Dostupné z: https://www.tutorialspoint.com/system_analysis_and_design/system_analysis_and_design_development_life_cycle.htm*

[25] *15 Advantages and Disadvantages of a Waterfall Model: VITTANA [online]. March 19, 2020 by Louise Gaille [cit. 2022-10-01]. Dostupné z: https://vittana.org/15-advantages-and-disadvantages-of-a-waterfall-model*

[26] *Iterative Model – Software Engineering [online]. June 16, 2022 [cit. 2022-10-01]. Dostupné z: https://www.interviewbit.com/blog/iterative-model/*

[27] *Spiral Model (Software Engineering) [online]. August 22, 2022 [cit. 2022-10-01]. Dostupné z: https://www.interviewbit.com/blog/spiral-model/*

[28] *Testování softwaru: Spirálový model [online]. August 22, 2022 [cit. 2022-10-01]. Dostupné z: http://testovanisoftwaru.cz/manualni-testovani/modely-zivotniho-cyklu-softwaru/spiralovy-model/*

[29] *INS - Strukturovaný přístup životního cyklu: Prototypový model [online]. [cit. 2022-10-01]. Dostupné z: https://moodle.sspbrno.cz/pluginfile.php/11146/mod_resource/content/2/10_INS_Strukt%20%C5%BEiv%20cyklus.pdf*

[30] *Software Engineering: Prototype Model [web]. artoftesting.com, April 29, 2021 [cit. 2022-10-01]. Dostupné z: https://artoftesting.com/prototype-model*

[31] *History: The Agile Manifesto [web]. https://agilemanifesto.org/: Jim Highsmith, for the Agile Alliance, 2001 [cit. 2022-10-01]. Dostupné z: https://agilemanifesto.org/history.html*

[32] *Independent Signatories of The Manifesto for Agile Software Development [web]. https://agilemanifesto.org/: Jim Highsmith, for the Agile Alliance, 2001 [cit. 2022-10-01]. Dostupné z: https://agilemanifesto.org/display/index.html*

[33] *Principles behind the Agile Manifesto: We follow these principles [web]. https://agilemanifesto.org/: Jim Highsmith, for the Agile Alliance, 2001 [cit. 2022-10-01]. Dostupné z: https://agilemanifesto.org/principles.html*

[34] *Agilní metodiky: Agilní metodiky vývoje softwaru [web]. cw.fel.cvut.cz: Antonin Komenda [cit. 2022-10-01]. Dostupné z: https://cw.fel.cvut.cz/old/_media/courses/a4b33si/prednaskove_materialy/si4-agile.pdf*

[35] *Lucidchart: What is Unified Modeling Language [online]. [cit. 2023-03-21]. Dostupné z: https://www.lucidchart.com/pages/what-is-UML-unified-modeling-language*

[36] *Der einfache Leitfaden zur Erstellung von UML-Diagrammen und Datenbankmodellen: UML [online]. Microsoft 365 Team, September 24, 2019 [cit. 20230]. Dostupné z: https://www.microsoft.com/de-de/microsoft-365/business-insights-ideas/resources/guide-to-uml-diagramming-and-database-modeling*

[37] *Visual Paradigm Online: MoSCoW Prioritization Template* [online]. 1802, Laford Center, 838 Lai Chi Kok Road, Kln, Hong Kong. [cit. 20230]. Dostupné z: https://online.visual-paradigm.com/contact/

[38] *TechTarget: MoSCoW method by Kate Brush* [online]. [cit. 20230]. Dostupné z: https://www.techtarget.com/searchsoftwarequality/definition/MoSCoW-method

[39] *Общие механизмы UML: УПРАВЛЕНИЕ ЖИЗНЕННЫМ ЦИКЛОМ ИНФОРМАЦИОННЫХ СИСТЕМ* [online]. [cit. 20230]. Dostupné z: https://studme.org/184160/informatika/obschie_mehanizmy

[40] *Lekce 5 - Extrémní programování: Extrémní programování* [online]. [cit. 20230]. Dostupné z: https://www.itnetwork.cz/navrh/metodiky/extremni-programovani

[41] *СОВРЕМЕННЫЕ ТЕХНОЛОГИИ ПРГРАММИРОВАНИЯ: А.П. Пашкевич, О.А. Чумаков* [online]. Минск: Министерство образования Республики

Беларусь Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники» Кафедра систем управления, 2007 [cit. 20230]. Dostupné z: https://www.bsuir.by/m/12_100229_1_65759.pdf

[42] *Scrum.org The home of scrum: What is scrum* [online]. [cit. 20230]. Dostupné z: https://www.scrum.org/learning-series/what-is-scrum

[43] *CONCISO: Scrum-Ursprünge und ein kurzer Überblick* [online]. [cit. 20230]. Dostupné z: https://conciso.de/scrum-urspruenge-und-ein-kurzer-ueberblick/

[44] *Dědičnost – odvození typů za účelem vytvoření specializovanějšího chování: Microsoft* [online]. [cit. 20230]. Dostupné z: https://learn.microsoft.com/cs-cz/dotnet/csharp/fundamentals/object-oriented/inheritance

[45] *DATABÁZE A INFORMAČNÍ SYSTÉMY* [online]. [cit. 20230]. Dostupné z: https://homel.vsb.cz/~dan11/is_skripta/IS%202011%20-%20Databaze%20v%20IS.pdf

[46] *Why and how to use prototypes: KAUST Innovation* [online]. January 17, 2018 [cit. 20230]. Dostupné z: https://innovation.kaust.edu.sa/why-and-how-to-use-prototypes/

[47] *UX vs UI: Their Differences & The Skills Needed: Parity consulting* [online]. [cit. 20230]. Dostupné z: https://www.parityconsulting.com.au/blog/2022/12/ux-vs-ui-their-differences-and-the-skills-needed

[48] *OCUP.CZ: Podrobná příprava k certifikaci znalostí UML.* [online]. 2011 [cit. 2023-03-25]. Dostupné z: http://ocup.ocup.cz/2010/11/sekvencni-diagramy.html

[49] *Interval.CZ: Návrh aplikací v jazyce UML – začínáme s případy užití* [online]. 2011 [cit. 2023-03-25]. Dostupné z: https://www.interval.cz/clanky/navrh-aplikaci-v-jazyce-uml-zaciname-s-pripady-uziti/

[50] *Úvod do UML: Diagramy* [online]. [cit. 2023-03-25]. Dostupné z: https://www.itnetwork.cz/navrh/uml/uml-uvod-historie-vyznam-a-diagramy

[51] *INS - Pojem informace: INFORMACE* [online]. [cit. 2023-03-25]. Dostupné z: https://www.itnetwork.cz/navrh/uml/uml-uvod-historie-vyznam-a-diagramy

[52] *Dáme jídlo: Restaurants & Shops you love brought to you* [online]. [cit. 2023-03-25]. Dostupné z: https://www.damejidlo.cz/en/

[53] *Wolt: Objevte Wolt města [online]. [cit. 2023-03-25]. Dostupné z: https://wolt.com/cs/cze*

[54] *Bolt Food: Delivery & Takeaway: Bolt Technology* [online]. [cit. 2023-03-27]. Dostupné z: https://play.google.com/store/apps/details?id=com.bolt.deliveryclient&hl=de&gl=US&pli=1

8 Přílohy

8.1 Scénář zakládání objednávky zákazníkem určený k testování grafického prototypu

Vytvořeni objednávky	
Aktéři	Zákazník
Podmínky pro spuštění	Zákazník si otevře hlavní stránku systému
Základní scénář	1. Zákazník si zvolí podnik z nabídky 2. Zákazník si zvolí typ nápojů 3. Zákazník přečte informace o nápoji 4. Zákazník si zvolí velikost zvoleného nápoje 5. Zákazník se přihlásí do systému 6. Zákazník pokračuje v nákupu 7. Zákazník si zvolí z nabízených kategorií „*Donuts*“ 8. Zákazník si zvolí občerstvení z nabídky podniku 9. Zákazník si prohlédne informace o produktu 10. Zákazník pokračuje k platbě zvoleného jídla a nápojů 11. Zákazník přejde k platbě *12. Na obrazovce se zobrazí „Payment geteway“, v rámci které je zákazník přesměrován na platební bránu, což není uvedené v prototypu* 13. Zákazník se vrací k zaplacené objednávce, po kliknutí na košík objednávky, kde se mu zobrazí čas vyzvednutí objednávky a možnost se vrátit na hlavní stranu 14. Zákazník se vrátí na hlavní stránku systému
Alternativní scénář	1. Zákazník si zvolí podnik z nabídky 2. Zákazník si zvolí typ nápojů 3. Zákazník si zvolí velikost zvoleného nápoje 4. Zákazník není registrován v systému 5. Zákazník se zaregistruje do systému

	6. Zákazník obdrží potvrzení na e-mail 7. Zákazník využije nadpis „*Let`s sign you up*“ pro přihlášení k systému 8. Zákazník se přihlásí do systému 9. Zákazník pokračuje v nákupu
Podmínky ukončení	Nová objednávka je vytvořená.

Tabulka 9: Scénář zakládání objednávky zákazníkem určený k testování grafického prototypu (Zdroj: Vlastní tvorba)

8.2 Scénář zakládání menu podnikatelem

Zakládání menu	
Aktéři	Podnikatel
Podmínky pro spuštění	Podnikatel si otevře hlavní stránku systému
Základní scénář	1. Podnikatel si zmáčkne tlačítko *„Business user“* 2. Podnikatel se přihlásí do systému 3. Podnikatel si zvolí záložku *„Create menu“* 4. Podnikatel si otevře záložku *„Menu name“* 5. Podnikatel napíše název menu a odklikne v systému možnost *„Create“* 6. Podnikatel uzavře záložku a zvolí záložku „*Drinks*“ 7. Podnikatel vyplní formulář a vloží obrázek nápoje 8. Po vyplnění formuláře podnikatel potvrdí založení položky menu 9. Podnikatel zmačknul tlačítko „*Create*“
Alternativní scénář	1. Podnikatel zmáčkne tlačítko *„Business user“* 2. Podnikatel si zmáčkne tlačítko „*Let`s sign you up*“ určené pro registrace v systému 3. Vyplní formulář 4. Potvrdí vyplněné údaje a pokračuje v registraci 5. Podnikatel vyplní další formulář 6. Potvrdí vyplněné údaje a obdrží e-mail 7. Zmačkne tlačítko „*Let`s sign you up“* 8. Přihlásí se jako registrovaný uživatel systému 9. Po přihlášení podnikatele se zobrazí stránka systému, v rámci které

	podnikatel čeká na schválení podnikatelského účtu administrátorem 10. Podnikatel si otevře položku obsahující údaje o registrace 11. Podnikatel uzavře položku s registračními údaje 12. Žádost je schválena, podnikatel může pokračovat v založení menu 13. Podnikatel si zvolí záložku *„Create menu“* 14. Podnikatel si otevře záložku *„Menu name“* 15. Podnikatel zapíše název menu a odklikne v systému možnost *„Create“* 16. Podnikatel uzavře záložku a zvolí záložku „*Drinks*“ 17. Podnikatel vyplní formulář a vloží obrázek nápoje 18. Po vyplnění formuláře podnikatel potvrdí založení položky menu 19. Podnikatel zmáčknul tlačítko „*Create*“
Podmínky ukončení	Nové menu je vytvořené.

Tabulka 10: Scénář zakládání menu podnikatelem (Zdroj: Vlastní tvorba)

8.3 Scénář schvalování podnikatele administrátorem

Schvalování podnikatele	
Aktéři	Administrátor
Podmínky pro spuštění	Administrátor si otevře hlavní stránku systému
Základní scénář	1. Administrátor zmáčkne tlačítko *„Admin“* 2. Administrátor se přihlásí do systému 3. Administrátor si zvolí položku „*Requests*“ 4. Administrátor otevře položku „*LILI company“* 5. Administrátor zkontroluje informace o zaplacení, zmáčkne tlačítko *„i“* 6. Administrátor schválí žádost podnikatele o založení podnikatelského účtu, zmáčkne tlačítko „*Confirm*“

	7. Administrátor uzavře položku „*LILI company*“ 8. Systém zobrazí administrátorovi stránku, schválená položka se zobrazí na konce seznamu 9. Administrátor se odhlásí ze systému, zmáčkne tlačítko s obsahem logotypu
Alternativní scénář	1. Administrátor zmáčkne tlačítko „*Admin*“ 2. Administrátor se přihlásí do systému 3. Administrátor si zvolí položku „*Requests*“ 4. Administrátor otevře položku „*LILI company*“ 5. Administrátor zkontroluje informace o zaplacení, zmáčkne tlačítko „*i*“ 6. Administrátor schválí žádost podnikatele o založení podnikatelského účtu, zmáčkne tlačítko „*Confirm*“ 7. Administrátor uzavře položku „*LILI company*“ 8. Systém zobrazí administrátorovi stránku, schválená položka se zobrazí na konci seznamu 9. Administrátor zmáčkne tlačítko administrátor 10. Systém zobrazí administrátorovi hlavní administrátorskou stránku
Podmínky ukončení	Podnikatelský učet je schválen.

Tabulka 11: Scénář schvalování podnikatele administrátorem (Zdroj: Vlastní tvorba)

8.4 Ukázka kódu, statická „*Add to card*“

https://github.com/EvgeniyaGorina/diplomka-kavu-dej

HTML

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" type="text/css" href="Glider.js/glider.css">
    <link href="https://fonts.googleapis.com/css?family=Lato:100,300,400,700,900"
rel="stylesheet">
    <link rel="stylesheet" href="style.css">
    <link rel="stylesheet" href="style_add_to_card.css">
    <!-- <link rel="stylesheet" href="style1.css"> -->
    <title>Your take away</title>
</head>

<body>
    <header>

        <div class="header">
            <div class="logo-box">
                <img src="img/logo/logo.svg" alt="logo" class="logo">
            </div>
        </div>
    </header>
    <div class="box_add_to_card">

        <div class="box-container_add_to_card">
            <div class="flex-container_add_to_card">
                <div class="flex-item-left_add_to_card">
                    <h3>Add to card</h3>
                </div>
                <div class="flex-item-right_add_to_card">
                    <img src="img/icons/my_order.svg" alt="my order"
class="my_order">
                </div>
            </div>
        </div>
        <div class="box-container_add_to_card">
            <div class="flex-container_add_to_card">
                <div class="flex-item_1_add_to_card">
                    <button class="close">
                        <img src="img/icons/close_button.svg">
                    </button>
                </div>
                <div class="flex-item_1_add_to_card">
                    <img src="img/icons/ordered_latte.png">
                </div>
                <div class="flex-item_2_add_to_card">
                    <p>Mocha Latte</p>
                </div>
                <div class="flex-item_3_add_to_card">
                    <p>Medium</p>
                </div>
                <div class="flex-item_4_add_to_card">
                    <p>1x</p>
```

```
            </div>
            <div class="flex-item_5_add_to_card">
                <p>80 kč</p>
            </div>
        </div>
    </div>
    <div class="box-container_add_to_card">
        <div class="flex-container_add_to_card">
            <div class="flex-item_1_add_to_card">
                <button class="close">
                    <img src="img/icons/close_button.svg">
                </button>
            </div>
            <div class="flex-item_1_add_to_card">
                <img src="img/icons/Ellipse 1.png">
            </div>
            <div class="flex-item_2_add_to_card">
                <p>Donut Strawberry</p>
            </div>
            <div class="flex-item_3_add_to_card">
                <p></p>
            </div>
            <div class="flex-item_4_add_to_card">
                <p>1x</p>
            </div>
            <div class="flex-item_5_add_to_card">
                <p>50 kč</p>
            </div>
        </div>
    </div>
    <div class="box-container_add_to_card">
        <div>
            <img src="img/icons/Total.svg">
        </div>
        <div class="text_total">
            <h3>Total</h3>
        </div>
        <div class="text_price">
            <h3>130 kč</h3>
        </div>
        <button class="btn-pay">Pay</button>
        <h3>Check out</h3>
    </div>
</div>
<footer>
<div class="box-footer">
    <div class="footer-container">
        <div class="flex-footer">
            <a class="home-logo" href="#">
```

```
                    <img src="img/icons/logo_footer.svg" alt="logo" class="logo-
footer">
                </a>
            </div>
            <div class="flex-footer profile">
                <a class="order-icon" href="#">
                    <img src="img/icons/profile_footer.svg" alt="profile"
class="profile-footer">
                    <p>Profile</p>
                </a>
            </div>
            <div class="flex-footer">
                <a class="order-icon" href="#">
                    <img src="img/icons/order_footer.svg" alt="order"
class="order-footer">
                    <p>Order</p>
                </a>
            </div>
            <div class="flex-footer shopping-cart">
                <a class="shopping-cart-icon" href="#">
                    <img src="img/icons/my_order_footer.svg" alt="my order"
class="my-order-footer">
                    <p>Shopping<br>cart</p>
                </a>
            </div>
        </div>
        <div class="container footer-copyright">
           <p> Copyright</p>
        </div>
    </div>

</footer>
<script src="script1.js"></script>
</body>

</html>
```

CSS

```
.box-container_add_to_card {
    margin: 15px 20px;
    border: 15px;
    border-right: 1px;
    font-size: 3.7vw;
    text-align: center;
    border-radius: 1px;
  }
  .flex-container_add_to_card {
    display: flex;
```

```
    flex-wrap: nowrap;
    color: #828282;
  }

  .flex-item-left_add_to_card {
    background-repeat: no-repeat;
    background-size: cover;
    background-position: bottom;
    flex: 80%;
    padding: 10px;
    border-radius: 6px;
    border: 1px solid white;
    margin-left: 40px;
  }
  .flex-item-left_add_to_card > h3{
     color: black;
}
  .flex-item-right_add_to_card {
    flex: 20%;
    padding: 1px 10px;
    padding-bottom: 2px;
  }
.close{
    border-color: transparent;
    background-color: transparent;
    position: relative;
}
.close img{
    width: 12px;
    margin-right: -40px;
    margin-top: -42px;
    z-index: 1000;
}
.btn-pay {
    padding: 7px;
    width: 7rem;
    display: flex;
    justify-content: center;
    background-color: #1cbaba;
    color: white;
    border: transparent;
    border-radius: 3px;
    margin-bottom: 17px;
    margin-top: 10px;
    font-weight: 600;
}

.text_total{
    color: black;
}
```

```
.text_price{
    color: #828282;
}
.box_add_to_card{
    margin-bottom: 10rem;
}
```