Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačního inženýrství

Diplomová práce

Návrh a implementace agregátoru e-shopů s potravinami působících v ČR

Bc. Jan Gottlieb

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ DIPLOMOVÉ PRÁCE

Bc. Jan Gottlieb

Informatika

Název práce

Návrh a implementace agregátoru e-shopů s potravinami působících v ČR

Název anglicky

Design and implementation of an aggregator of grocery e-shops operating in the CZ

Cíle práce

Cílem diplomové práce je návrh a vytvoření aplikace umožňující srovnání vybraných online e-shopů s potravinami (takzvaného agregátoru), které působí v ČR. Cílem vytvoření této aplikace je usnadnění porovnání cen potravin a na to navazujícího nákupního procesu na různých e-shopech.

Metodika

1) Analýza API vybraných e-shopů s potravinami

2) Analýza současných řešení nákupního procesu různých e-shopů

3) Návrh nového společného nákupního procesu

4) Návrh softwarové architektury agregátoru

5) Návrh databáze

6) Vytvoření testovacích dat

7) Implementace BE

8) Návrh UI

9) Implementace FE

Doporučený rozsah práce

60-80 stran

Klíčová slova

spring angular aplikace agregátor srovnání online e-shop

Doporučené zdroje informací

Spring.io [online], c2022. Palo Alto: VMware [cit. 2022-09-02]. Dostupné z: https://spring.io
WALLS, Craig, 2016. Spring Boot in Action. 1. New York: Manning Publications. ISBN 978-1617292545.
WALLS, Craig, 2018. Spring in Action. 5. New York: Manning Publications. ISBN 978-1617294945.
WILKEN, Jeremy, 2018. Angular in Action. 1. New York: Manning Publications. ISBN 978-1617293313.

Předběžný termín obhajoby

2022/23 LS – PEF

Vedoucí práce

Ing. Jiří Brožek, Ph.D.

Garantující pracoviště

Katedra informačního inženýrství

Elektronicky schváleno dne 26. 11. 2022

Ing. Martin Pelikán, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 28. 11. 2022

doc. Ing. Tomáš Šubrt, Ph.D.

Děkan

V Praze dne 12. 03. 2024

Čestné prohlášení

Prohlašuji, že svou diplomovou práci "Návrh a implementace agregátoru e-shopů s potravinami působících v ČR" jsem vypracoval(a) 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 autor uvedené diplomové práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 28.3.2024 _____

Poděkování

Rád bych poděkoval vedoucímu mé práce panu Ing. Jiřímu Brožkovi, Ph.D. za jeho cenné rady, trpělivost a vstřícný přístup. Dále bych rád poděkoval všem, kteří mi při mé práci byli oporou.

Návrh a implementace agregátoru e-shopů s potravinami působících v ČR

Abstrakt

Hlavním cílem této diplomové práce je vytvoření webové aplikace, která bude umožňovat automatizované srovnání online cen (agregovat) ve vybraných potravinových online obchodech (e-shopech), které se v současné době vyskytují na českém trhu.

V práci jsou nejprve představeny současné možnosti online nákupu potravin, které jsou zákazníkovi na území ČR k dispozici. Poté jsou podrobněji popsány nákupní procesy z pohledu zákazníka u vybraných e-shopů. Na základě předchozích zjištění je navržen nový nákupní proces, který zákazníkovi zjednoduší proces výběru potravin a ve výsledku mu ušetří čas a finance.

Nedílnou a hlavní součástí této práce je však tvorba webové aplikace, která bude řešit hlavní nevýhody nakupování na více online potravinových obchodech zároveň. K tomuto účelu jsou představeny v teoretické části použité technologie a v praktické části je popsán především detailně výsledný softwarový produkt.

Klíčová slova: online, e-shop, agregace, aplikace, potraviny, porovnání, Java, Spring, Spring Boot, Angular, PostgreSQL, Keycloak

Design and implementation of an aggregator of grocery e-shops operating in the CZ

Abstract

The main goal of this diploma thesis is to create a web application that will enable automated comparison of online prices (aggregation) in selected online grocery stores (e-shops) currently present on the Czech market.

The current options for online grocery shopping available to customers in the Czech Republic are introduced first. Then the shopping processes at selected e-shops are described in more detail from the customer's perspective. Based on the previous findings, a new shopping process is designed to simplify the process of selecting groceries for customers and in the end save them time and budget.

However, an integral and main part of this work is the development of a web application that will address the main disadvantages of shopping at multiple online grocery stores simultaneously. For this purpose, the technologies used are introduced in the theoretical part and the resulting software product is described in detail mainly in the practical part.

Keywords: online, e-shop, aggregation, application, grocery, comparison, Java, Spring, Spring Boot, Angular, PostgreSQL, Keycloak

Obsah

1 Úvod 11
1.1 Motivace 11
1.2 Struktura práce 12

2 Cíl práce a metodika 14
2.1 Cíl práce 14
2.2 Metodika 14

3 Teoretická část práce 15
3.1 Rešerše potravinového online trhu v ČR 15
3.1.1 Online supermarkety 17
3.1.2 Srovnávací portály 18
3.2 BPMN 19
3.2.1 Historie a vývoj BPMN 20
3.2.2 Základní elementy BPMN 21
3.3 Softwarová architektura 24
3.3.1 UML 25
3.3.2 Volba architektury 26
3.4 Backendové technologie 28
3.4.1 Java 29
3.4.2 Spring framework 30
3.4.3 REST a Richardson Maturity Model 32
3.5 Keycloak 34
3.6 Frontendové technologie 35
3.6.1 Angular 35
3.7 Databáze 36
3.7.1 Relační databáze a PostgreSQL 36
3.7.2 Objektově relační mapování, JPA a Spring Data JPA 37
3.8 Mocking 37

4 Praktická část práce 39
4.1 Analýza a sběr požadavků 39
4.1.1 Funkční požadavky 40
4.1.2 Nefunkční požadavky 42
4.1.3 Aktéři a případy užití 43
4.1.4 Mapování případů užití na funkční požadavky 48
4.2 Analýza API vybraných e-shopů s potravinami 49
4.2.1 Rohlik API 50
4.2.2 Billa API 50

4.3 Analýza současných řešení nákupního procesu různých e-shopů 52
4.4 Návrh nového společného nákupního procesu 55
4.5 Návrh softwarové architektury agregátoru 57
4.6 Návrh databáze 59
4.7 Vytvoření testovacích dat 63
4.8 Návrh uživatelského rozhraní (UI) 63
4.8.1 Wireframe hlavní nákupní stránky 65
4.8.2 Hlavní nákupní stránka – desktop 65
4.8.3 Responzivní zobrazení 65
4.9 Implementace Backendu 68
4.10 Implementace Frontendu 69

5 Zhodnocení výsledků 71

6 Závěr 73

7 Seznam použitých zdrojů 75

8 Seznam obrázků, tabulek, grafů, diagramů, kódů a zkratek 77
8.1 Seznam obrázků 77
8.2 Seznam tabulek 77
8.3 Seznam grafů 77
8.4 Seznam diagramů 77
8.5 Seznam zdrojových kódů 78
8.6 Seznam použitých zkratek 78

Přílohy 80

1 Úvod

Ačkoliv se v současné době online služby dotýkají téměř všech oblastí dnešního života, tak až do příchodu pandemie COVID-19 do ČR v roce 2020 se stále jedna oblast online služeb a prodeje vymykala oproti ostatním segmentům e-commerce[1], a sice oblast prodeje potravin a domácích potřeb.

V době pandemie byla postupně vládou zaváděna a později zase rozvolňována různá omezení, v závislosti na epidemiologické situaci (počtu nakažených apod.). Tato omezení měla samozřejmě za cíl omezit setkávání většího množství lidí, a tím i zamezit šíření viru v populaci. Tyto restrikce zasáhly do mnoha oblastí našich životů — jednalo se například o omezení pohybu osob, zákaz kulturních akcí, dokonce i omezení nákupu v „kamenných" prodejnách a různých dalších „kamenných" službách jako jsou kadeřnictví apod. Zmíněné restrikce se ale prakticky zcela nedotkly elektronického obchodování, které díky tomu zaznamenalo podstatný úspěch v podobě nárůstu tržeb.

Ve výše zmíněné době byly k dispozici na českém trhu následující společnosti zabývající se online prodejem potravin a jiného zboží. Jednalo se především o následující online obchody: *Rohlik.cz*, *iTesco.cz*, *Košík.cz*, a až poměrně nedávno přibyly na trhu také společnosti *Billa.cz a Albert.cz*. S ohledem na tento vývoj a nárůst poptávky po online nakupování potravin vyvstala také potřeba online srovnávání cen těchto e-shopů.

1.1 Motivace

Autor práce konstatuje, že na počátku pandemie byly v jeho domácnosti také často využívány služby online nákupu, a to výhradně u nejrozšířenějšího českého online supermarketu s potravinami *Rohlik.cz* a tento způsob nakupování se stával časem oblíbenějším. Typický scénář, který se však odehrával stále častěji v domácnosti autora práce byl ten, že postupně s ohledem na vyšší ceny nákupů, u již zmíněného e-shopu, narůstal zájem o ceny i u jiných konkurenčních potravinových e-shopů. Následně byly

[1] E-commerce (též. e-komerce, elektronické obchodování) je forma obchodování, která k realizaci obchodních transakcí podstatným způsobem používá moderní elektronické komunikační prostředky.

zaregistrovány pravidelné významné rozdíly v cenách například masa (a dalších masných výrobků), ovoce, zeleniny a dalšího zboží. V jistou chvíli bylo proto při nákupu přistoupeno k pravidelnému porovnávání cen jednotlivých online supermarketů mezi sebou.

Nevýhoda tohoto procesu spočívala v tom, že při porovnávání jednotlivých cen potravin bylo nutné neustále pracovat s každou webovou aplikací zvlášť. A vzhledem k tomu, že dnes je běžné nakupování pouze přes chytrý telefon, bylo toto řešení velmi nepraktické a neefektivní z důvodu neustálého přepínání aplikací (případně webových stránek), a to obzvlášť na starších telefonech.

Neznalému člověku se nejprve může zdát, že řešením této problematiky v online prostoru by mohlo být použití některého známého českého internetového portálu jako je *Kupi.cz, Heureka.cz, Zboží.cz* či například *AkcniCeny.cz*. Realita je však taková, že každý z těchto portálů má svá určitá omezení, která neumožňují užití k danému účelu.

1.2 Struktura práce

I přesto, že problematika současných možností online nákupů a srovnání cen není hlavním cílem této práce, bylo nakonec rozhodnuto tuto část nevynechat a čtenáři v příslušné kapitole alespoň trochu osvětlit zaměření a nevýhody zmíněných portálů. Protože právě díky zjištění nedostatků (z pohledu srovnávání cen online potravin) byla autorem této práce identifikována tzv. „mezeru na trhu“ a tento faktor byl považován za rozhodující vůbec pro vznik myšlenky na vytvoření této webové aplikace a následně i této diplomové práce.

Následně jsou v krátkosti představeny dnes dostupné online supermarkety a u vybraných (v praktické části) jsou analyzovány jejich API[2] a je představen jejich uživatelský nákupní proces. Analýza API třetích stran je v této práci samozřejmě zcela nezbytná, protože bez její dostatečné znalosti by vytvoření této aplikace nebylo zkrátka vůbec možné.

---

[2] API (Application Programming Interface) – označuje v informatice způsob, jakým spolu mohou komunikovat dva nebo více počítačových programů nebo komponent.

V teoretické části, kromě výše zmíněného, jsou představeny pro tvorbu této aplikace důležité technologie a nástroje, jako je například grafická notace pro procesní modelování (BPMN), programovací jazyk Java, framework Spring Boot a Angular, databáze PostgreSQL, nástroj Keycloak a Mockoon ad. V krátkosti je popsána i teorie zabývající se softwarovou architekturou a aplikačním rozhraním (API).

V praktické části práce kromě výše zmíněné analýzy API a procesů je také popsána analýza a sběr požadavků a dále pak samotný návrh nového nákupního procesu, softwarová architektura, databázový model a v neposlední řadě i uživatelské rozhraní. Pro potřeby testování bylo zapotřebí vytvořit mockovaná (testovací) data a okolnosti jejich tvorby jsou také uvedeny v příslušné kapitole této práce. Praktická část je uzavřena popisem implementace, a to jak backendové, tak i frontendové části aplikace.

V kapitole Zhodnocení výsledků je ve stručnosti shrnuta celá vytvořená aplikace od použitých technologií, přes stručný popis hlavní funkcionality a uživatelského rozhraní, až po možný prostor pro zlepšení do budoucna.

V závěru diplomové práce jsou uvedeny do souvislosti stanovené cíle této práce a dosažené výsledky a hodnocení splnění těchto cílů.

2 Cíl práce a metodika

2.1 Cíl práce

Cílem diplomové práce je návrh a vytvoření aplikace umožňující srovnání vybraných online e-shopů s potravinami (takzvaného agregátoru), které působí v ČR. Cílem vytvoření této aplikace je usnadnění porovnání cen potravin a na to navazujícího nákupního procesu na různých e-shopech.

2.2 Metodika

Tato diplomová práce byla zpracována s využitím níže popsaných metod.

V první fázi byla provedena detailní analýza API vybraných e-shopů s potravinami, které v současné době působí na trhu v ČR, a to s cílem zjistit především možnosti integrace v oblasti přihlašování, získávání dat o produktech a nákupních košících uživatele a další. Poté následovala analýza současného řešení nákupního procesu na vybraných online supermarketech a identifikace případných problémových oblastí ve zkoumaném nákupním procesu. Oproti metodice v zadání byla navíc provedena ještě analýza požadavků na software, která potvrdila soulad mezi funkčními požadavky a případy užití.

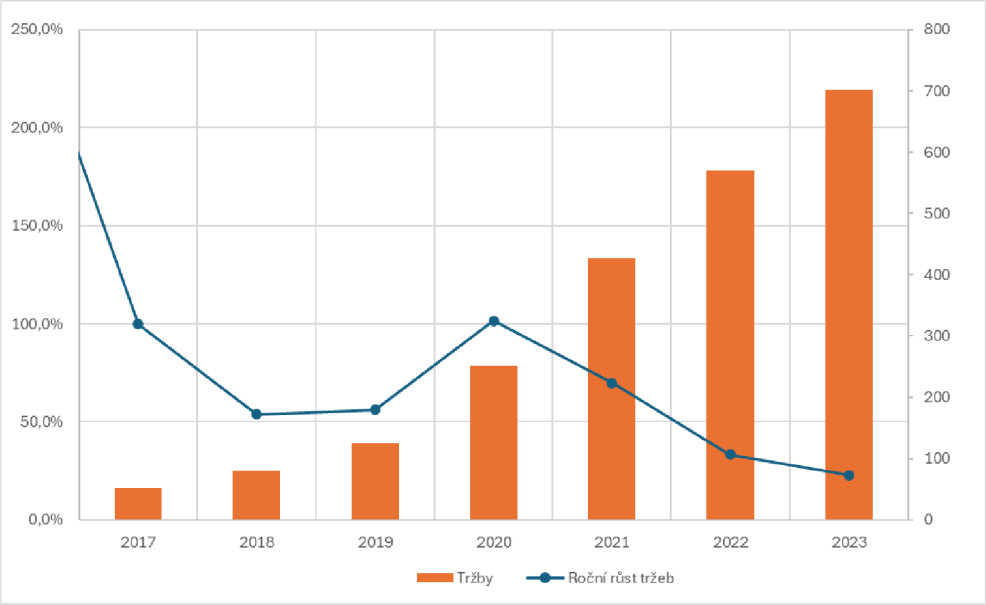
V další fázi byl navržen, na základě získaných poznatků, optimalizovaný sloučený proces pro potravinový agregátor, dále jeho softwarová architektura, datový model databáze a uživatelské rozhraní s využitím drátěného modelu (wireframe). Poté byla vytvořena testovací (mockovaná) data, mj. pro snížení rizik, plynoucích ze závislosti na externích systémech.

V závěru následovala fáze implementace, a to jak backendové části v jazyce Java, s využitím frameworku Spring Boot, tak i frontendové části v jazyce TypeScript, s využitím frameworku Angular a Material Designu. Oproti metodice v zadání byla navíc celá aplikace nakonfigurována s využitím nástroje pro jednotné přihlášení Keycloak, který je navíc připraven společně s databázovým strojem PostgreSQL k nasazení s využitím platformy Docker.

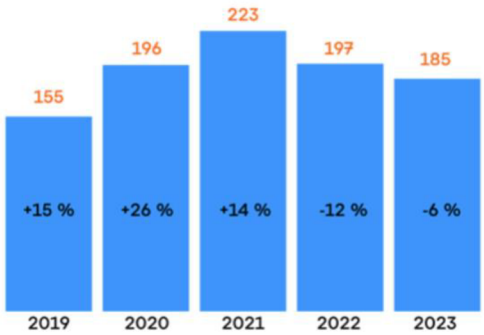
3 Teoretická část práce

3.1 Rešerše potravinového online trhu v ČR

Jak již bylo v úvodní kapitole zmíněno, tak vlivem pandemie COVID-19 rapidně stoupla popularita nakupování potravin online v příslušných letech. Tento trend výrazného růstu tržeb však neustále pokračuje, což dokládá i následující graf (viz Graf 1) u největší společnosti v této kategorii v České republice, a to Rohlik.cz. Výhodou také je, že společnost uveřejňuje každoroční výsledky svých tržeb, takže je možné porovnat roční změny (nárůsty) s ostatními sektory e-commerce (viz Graf 2).

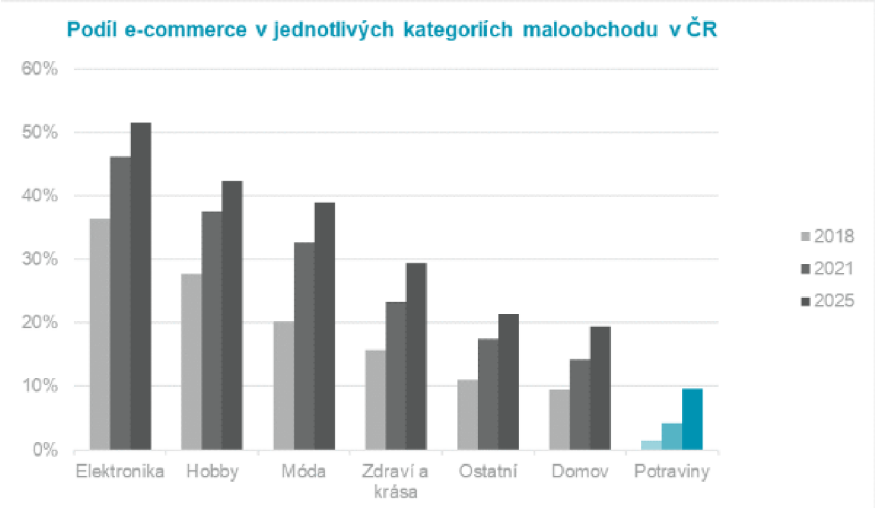


Graf 1: Roční tržby společnosti Rohlik Group (tržby v mil. EUR); Zdroj: vlastní tvorba z různých veřejných zdrojů



Graf 2: Obrat české e-commerce (v mld. Kč); Zdroj: (18)

Ze zmíněných grafů lze konstatovat, že roční růsty tržeb v oblasti online prodeje potravin jsou značné, a i když u největšího hráče na trhu (Rohlik.cz) došlo poměrově ke zpomalení nárůstu tržeb (z vysokých hodnot přesahujících až 100 % na 23 % v roce 2023), tak vzhledem k ostatním sektorům e-commerce (v roce 2023 byl vykázán pokles -6 % a v roce 2022 dokonce -12 %), rozdíl v tržbách dosahuje obrovského rozdílu téměř 30 % v roce 2023.



Graf 3: Podíl e-commerce v jednotlivých kategoriích maloobchodu v ČR; Zdroj: (1)

Samozřejmě je potřeba při porovnávaní ročních tržeb vzít v potaz, že velikost tržeb celého e-commerce sektoru je nesrovnatelná s velikostí jednoho hráče na trhu, a ještě v minoritní oblasti. Z toho důvodu je ještě přiložen graf (viz Graf 3) srovnání sektorů e-commerce v ČR, ze kterého lze konstatovat, že v roce 2021 v sektoru potravin bylo prodáno pouze 4,2 % přes online prodej a lze tedy předpokládat do budoucna velký prostor k rozvoji tržeb, což opět dokládá v tomtéž grafu i predikce na rok 2025. Tuto teorii potvrzují také experti v oblasti investic, například *Michal Soták,* vedoucí investičního týmu společnosti *Cushman & Wakefield*:

> *„Trend online nakupování potravin započal již před pandemií, ta ho pak samozřejmě akcelerovala – možnost nechat si dovézt potraviny domů se stala atraktivní i pro ty, kdo dříve váhali. Mnozí přínosy této formy nákupu potravin oceňují i nadále, a lze tak předpovídat další růst segmentu – do roku 2025 očekáváme více než zdvojnásobení jeho podílu na celkovém prodeji potravin v Česku.“ (1)*

3.1.1 Online supermarkety

V této kapitole jsou více popsáni jen ti nejznámější online prodejci potravin na českém trhu. Záměrně v ní u jednotlivých e-shopů nejsou uvedeny mapy rozvozu, protože největší hráči na trhu (až na několik výjimek) jsou již dnes schopni bez problému obsloužit největší města v ČR. Naopak u těch menších a začínajících e-shopů je problém tuto mapu vůbec získat, případně se aktualizuje tak často, že informace by byly ve velmi krátké době neaktuální.

Rohlik.cz

Rohlik.cz je průkopník a lídr na českém online trhu prodeje potravin, který začal působit v roce 2014. V roce 2022 nabízel přes 21 000 položek včetně čerstvých a lokálních potravin, drogerie i lékárenského zboží. Klade důraz na kvalitu, čerstvost a rychlost a ve vybraných částech Prahy a Brna doručuje už do 60 minut, jinde do 90 minut (2). Má vlastní logistiku a plánuje automatizaci skladů. Působí nejen v ČR (jako Rohlik.cz), ale od roku 2019 i v Maďarsku (Kifli.hu), od roku 2020 v Rakousku (Gurkerl.at) a od roku 2021 v Německu (Knuspr.de). Majitel Tomáš Čupr se netají svými ambiciózními plány na expanzi do mnoha dalších zemí. Rohlik.cz vyniká i v tom, jakým způsobem informuje veřejnost o jeho tržbách a ziscích v českém i evropském kontextu prostřednictvím mateřské společnosti Rohlik Group (více viz Graf 1). Minimální hodnota objednávky je 749 Kč (pro nečleny klubu Premium) a doprava je zdarma nad 1 500 Kč a při doručení do úložné schránky – tzv. Rohlik Pointu již nad 500 Kč. Rohlik.cz má dobrou mobilní aplikaci a nabízí prémiové členství Rohlík Premium s řadou výhod.

Košík.cz

V roce 2015 vznikl online obchod s potravinami Košík.cz, který v roce 2017 odkoupila skupina *Mall Group* a následně ho sloučila se svým obchodem Kolonial.cz. Od roku 2020 spojil Košík.cz své síly s obchodním řetězcem Kaufland a nabízí tak jejich privátní výrobky online. V roce 2021 nabízel přes 16 000 položek potravin, drogerie a dalšího zboží (3). V sortimentu Košík.cz najdete i nabídku oblečení a doplňky od Marks & Spencer, produkty z lékárny Dr.Max nebo potravinové výrobky značky Delmart, která je zaměřena na kvalitní a čerstvé suroviny. Má věrnostní kluby se slevami pro rodiny s dětmi a seniory. V roce 2022 měl tržby 2,3 miliardy Kč (4). Minimální hodnota objednávky je 600 Kč a doprava zdarma je nad 1 200 Kč. Košík.cz má mobilní aplikaci.

iTesco

iTesco je online služba britského obchodního řetězce Tesco fungující už od roku 2012. Má širokou síť prodejen a distribučních center po celé ČR. Nabízí online k dispozici přes 17 000 produktů s vyzvednutím na prodejně nebo doručením domů. Má věrnostní program *Clubcard* s řadou slev a výhod. Umožňuje objednávky i přes mobilní aplikaci a dále také nabízí službu „*Klikni+Vyzvedni*“, díky čemuž je možné zboží objednat online a následně pak vyzvednout na nejbližší pobočce. Minimální hodnota nákupu pro doručení je 700 Kč (pro členy Clubcardu i méně za poplatek), dopravu zdarma iTesco nenabízí za žádných podmínek (nebo nejsou veřejně známy). Společnost Tesco své tržby a ziskovost z online prodeje nezveřejňuje.

Billa

Billa spustila plnohodnotný e-shop v roce 2023 s cílem stát se do roku 2025 dvojkou na trhu, jak uvedl ředitel Liam Casey:

> *„Do konce roku 2025 máme ambici nabídnout naše služby po celé České republice a stát se druhým nejsilnějším hráčem v prodeji potravin online v ČR,“ (5)*

E-shop nabízí 9 000 položek s doručením do 3 hodin, ale protože je na trhu velmi krátce, tak zatím obsluhuje pouze velká města jako Praha a Brno a jejich okolí. Má vlastní distribuční centra a láká na dopravu zdarma, ovšem minimální objednávka je 700 Kč. Billa zatím nemá mobilní aplikaci, přes kterou by bylo možné uskutečnit online nákup. Nabízí členství v BILLA klubu, ale toto členství má vliv jen na ceny produktů podobně jako v kamenném obchodě.

3.1.2 Srovnávací portály

Největší univerzální srovnávače (agregátory) cen jako Heureka.cz a Zbozi.cz obsahují kategorii potravin, ale agregují většinou nabídky spíše menších specializovaných e-shopů. Produkty od největšího online supermarketu Rohlik.cz, který má širokou nabídku potravin, v těchto srovnávačích nenajdeme vůbec. Produkty z Košík.cz nebo iTesco.cz sice nalézt lze, ale je to v minimálním množství (až sporadického charakteru) a když už jsou nějaké produkty k nalezení, tak se většinou jedná jen o trvanlivé potraviny. Pro komplexní

srovnání běžných cen potravin z různých velkých řetězců tedy tyto agregátory nejsou příliš využitelné.

Specializované portály jako je Kupi.cz nebo AkcniCeny.cz se sice zaměřují na agregaci cen potravin (a drogerie) i velkých kamenných supermarketů (jako je Albert, Billa, Globus, Kaufland, Lidl, Tesco apod.), a dokonce i Košík.cz, ale jedná se vždy jen o pár produktů, které jsou zrovna součástí nějaké akční slevové (tzv. letákové) nabídky. S výjimkou Košík.cz se tedy navíc jedná vždy o ceny, které jsou dostupné jen v kamenných obchodech.

Navíc i pokud už si uživatel nějakým způsobem zjistí aktuální výhodnou cenu (ať už z univerzálního cenového agregátoru či specializovaného „letákového" portálu), tak zatím není možné, aby si mohl pomocí jednoho kliknutí přidat daný produkt do košíku u daného potravinového e-shopu.

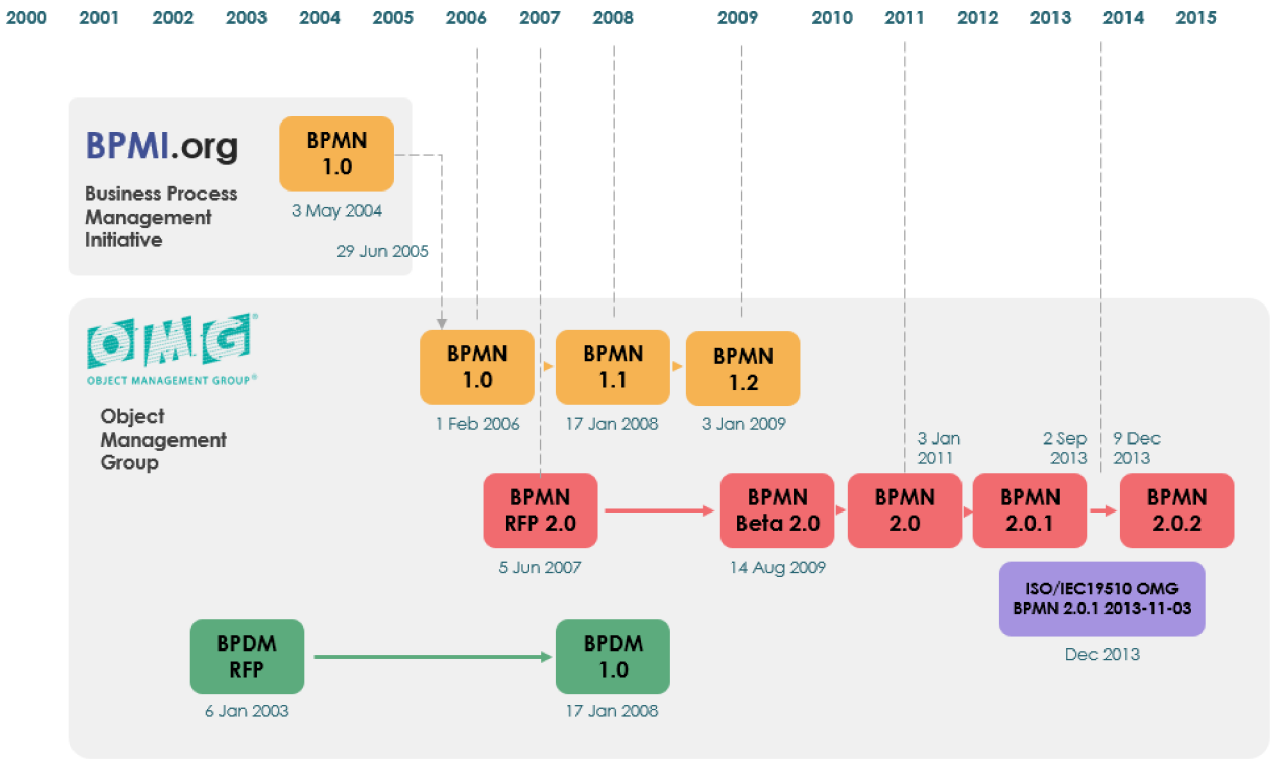
Komplexní agregátor, který by průběžně porovnával úplné nabídky, jak akčních, tak standardních cen online supermarketů (jako je Rohlík, Košík, iTesco, Billa atp.) v ČR zatím dosud nebyl vytvořen. Důvodem je patrně neochota velkých online hráčů poskytovat svá kompletní data srovnávacím portálům. Agregace cen potravin je také náročnější než u jiného zboží kvůli častým změnám cen, akcím, krátkodobým nabídkám a jejich lokalizaci. Zákazník si tak dosud musel ceny porovnávat buď přímo na webech jednotlivých e-shopů, nebo v jejich příslušných mobilních aplikacích.

3.2 BPMN

Business Process Model and Notation (BPMN) je grafická notace, která slouží k modelování a popisu podnikových procesů. Jedná se o soubor grafických objektů a pravidel, podle nichž jsou tyto objekty spojovány, aby vytvořily vizuální reprezentaci procesu. BPMN se stalo v podstatě standardem pro modelování podnikových procesů a je dnes také ratifikovaným standardem Mezinárodní organizace pro normalizaci (ISO) jako norma ISO/IEC 19510:2013, která je ve shodě s BPMN 2.0.1 (6).

3.2.1 Historie a vývoj BPMN

Vývoj BPMN začal v roce 2004 pod záštitou Business Process Management Initiative (BPMI). Po několika verzích se od roku 2014 prosazuje verze BPMN 2.0.2, která přináší řadu vylepšení a rozšíření oproti předchozím verzím.



Obrázek 1: Znázorněný vývoj BPMN; Zdroj: (19)

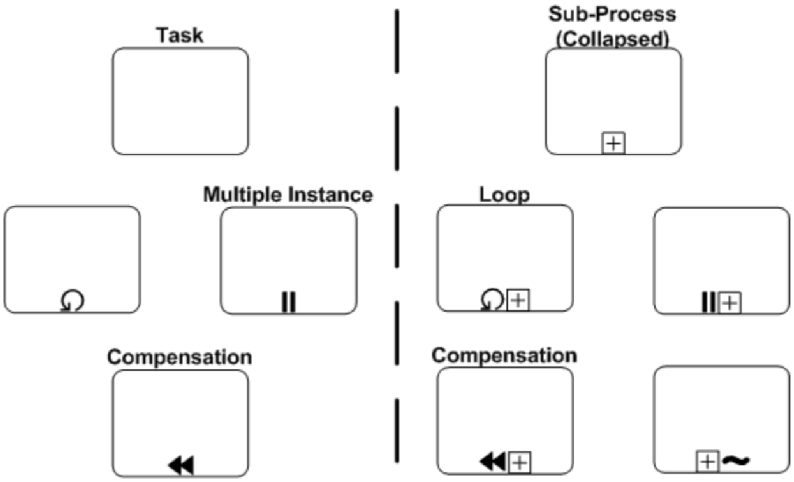
Původním cílem BPMN bylo poskytnout notaci, která bude snadno srozumitelná všem zainteresovaným stranám, od business analytiků, kteří vytváří první návrhy procesů, přes technické vývojáře odpovědné za implementaci technologií, až po manažery dohlížející na procesy. BPMN tak v podstatě pomáhá tvořit jakýsi most mezi návrhem procesu a jeho implementací (7).

Díky neustálému rozvoji (viz Obrázek 1) se tato notace stává stále více známějším a užitečnějším nástrojem nejen pro modelování, ale i pro řízení a optimalizaci procesů.

3.2.2 Základní elementy BPMN

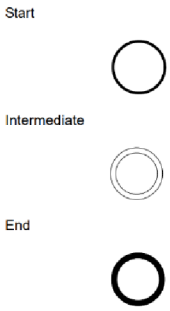
BPMN definuje několik základních kategorií elementů, a u těch zcela nepostradatelných je uvedeno i jejich grafické znázornění podle standardu BPMN 2.0 (8).

- **Plovoucí objekty (Flow Objects)** – Jsou hlavními grafickými elementy definujícími chování procesu. Do této kategorie patří:
    - Aktivity (Activities) – Reprezentují práci nebo úkol vykonávaný v rámci procesu. Mohou být atomické (Task) nebo složené (Sub-Process).



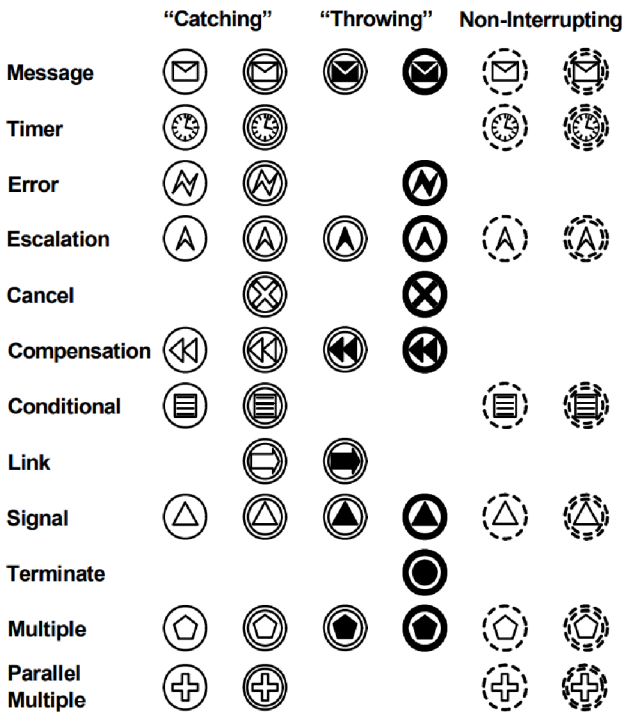
Obrázek 2: Rozdělení aktivit dle BPMN; Zdroj: (8)

    - Události (Events) – Reprezentují něco, co se stane během procesu a ovlivní tak jeho tok.
        - Z procesního pohledu je tedy můžeme dělit na: startovní (Start), průběžné (Intermediate) a koncové (End).



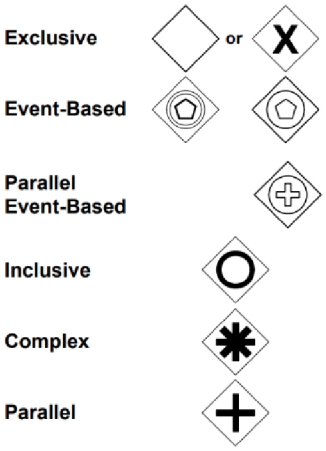
Obrázek 3: Rozdělení událostí dle BPMN z procesního pohledu; Zdroj: (8)

- Podle příčiny, která událost vyvolala, pak zobrazujeme uvnitř tohoto symbolu příslušnou značku, viz níže:



Obrázek 4: Rozdělení událostí dle BPMN z pohledu příčinného; Zdroj: (8)

o Brány (Gateways) – Používají se k řízení větvení a slučování toků v procesu. Mohou být exkluzivní (Exclusive), inkluzivní (Inclusive), paralelní (Parallel) a další viz níže:



Obrázek 5: Rozdělení bran dle BPMN; Zdroj: (8)

- **Data** – Obsah této kategorie je zřejmý z názvu.
    - Datové objekty (Data Objects) – Ukazují, jaká data jsou aktivitami vyžadována nebo produkována.
    - Datové vstupy (Data Inputs) – Jsou externím vstupem pro celý proces. Je to druh vstupního parametru.
    - Datové výstupy (Data Outputs) – Jsou výsledkem celého procesu. Je to druh výstupního parametru.
    - Datová úložiště (Data Stores) – Jsou to místa, kde může proces číst nebo zapisovat data, např. databáze nebo i do kartotéky. Přetrvává i po skončení životnosti instance procesu.
- **Spojovací objekty (Connecting Objects)** – Propojují plovoucí objekty.
    - Sekvenční tok (Sequence Flow) – Ukazuje pořadí, v jakém budou aktivity v procesu vykonávány.
    - Tok zpráv (Message Flow) – Reprezentuje zasílání zpráv mezi dvěma účastníky procesu.
    - Asociace (Association) – Používá se k propojení informací a artefaktů s plovoucími objekty.
    - Datové asociace (Data Associations) – Datové asociace se používají k přesunu dat mezi datovými objekty, vlastnostmi a vstupy a výstupy aktivit, procesů a globálních úloh.

Obrázek 6: Rozdělení spojovacích objektů dle BPMN; Zdroj: (8)

- **Plavecké dráhy (Swimlanes)** – Používají se k organizaci a kategorizaci aktivit.
    - Bazény (Pools) – Reprezentují účastníka v procesu. Bazén může být tzv. "Black Box" pokud jeho interní procesy nejsou v diagramu viditelné.
    - Dráhy (Lanes) – Jsou části bazénu a používají se k organizaci a kategorizaci aktivit podle funkce nebo role.
- **Artefakty (Artifacts)** – Umožňují přidat do diagramu další informace.
    - Skupiny (Groups) – Slouží k vizuálnímu seskupení elementů, které spolu logicky souvisí. Neovlivňují tok procesu.
    - Poznámky (Text Annotations) – Umožňují přidat k diagramu dodatečné informace ve formě textu.

Díky této sadě elementů jsme pomocí BPMN schopni popsat i poměrně složité podnikové procesy, a to způsobem stále srozumitelným pro všechny zúčastněné strany. Další popis této notace by vydal na mnoho stran textu, a přesto by bez grafického znázornění stejně nebyl kompletní. Z tohoto důvodu je považován (vzhledem k zaměření a rozsahu této diplomové práce) popis tohoto tématu jako dostačující.

3.3 Softwarová architektura

Pojem *Softwarová architektura* není dodnes zcela přesně vymezen (9) a je bohužel poměrně nejednoznačný. V tuto chvíli asi nejrozšířenější definice pochází z Institutu softwarového inženýrství při Carnegie Mellon University v Pittsburghu, která v překladu zní takto: „Softwarová architektura je struktura komponent programu/systému, jejich vzájemné vazby, principy a předpisy určující jejich návrh a vývoj v průběhu času“ (9).

Jelikož tato definice není zatím všeobecně přijímána a ukotvena a i podle známé české renomované společnosti v oblasti softwarového vývoje platí, že: "Neexistuje přesná definice toho, co je to softwarová architektura, a naopak existuje spousta výkladů a rozdělení, co se za softwarovou architekturu označuje“ (10), tak v této práci vzhledem k tematickému zaměření není dopodrobna popsáno rozdělení softwarové architektury na styly a vzory (apod.) a jsou v následujících podkapitolách uvedeny pouze nejznámější pojmy, které jsou relevantní vzhledem k tématu této práce.

Architektura vyvíjeného softwaru je v podstatě základním kamenem každého úspěšného softwarového projektu. Definuje celkovou strukturu a organizaci systému, jeho komponent a jejich vzájemné interakce. Kvalitní architektura by měla zajišťovat to, že software bude splňovat funkční i nefunkční požadavky, bude dostatečně škálovatelný, udržovatelný a rozšiřitelný vzhledem k požadavkům na něj kladených.

Bude-li se tedy vycházet z nejčastěji přijímané definice (9), tak důležitým aspektem této architektury je způsob komunikace mezi komponentami. Zde se často používá koncept API, což je sada pravidel a protokolů pro interakci mezi softwarovými komponentami. Nejběžnějším typem API pro webové aplikace je REST (Representational State Transfer), který využívá standardní HTTP[3] metody (GET, POST, PUT, DELETE) pro manipulaci s tzv. zdroji. Protože se jedná dnes o převažující způsob vývoje moderních webových aplikací, tak je toto téma vyčleněno zvlášť do kapitoly 3.4.3.

3.3.1 UML

Pro vizualizaci a dokumentaci softwarové architektury je možné použít více různých diagramů. Nicméně pro mapování softwarové architektury na fyzickou architekturu systému se jeví jako skutečně nejvhodnější jazyk UML (Unified Modeling Language), a to konkrétně diagram nasazení (deployment diagram).

Tento diagram používá především následující prvky, jejichž český překlad není dosud zcela ustálen:

- Node (Uzel) – Je buď hardwarový nebo softwarový prvek.
- Artifact (Artefakty) – Je produktem procesu vývoje softwaru. To může zahrnovat jak modely procesů (např. modely případů použití, modely návrhu atd.), tak zdrojové soubory, spustitelné soubory, dokumenty návrhu, zprávy o zkouškách, prototypy, uživatelské příručky atd.
- Association (Asociace) – Představuje komunikační propojení mezi uzly.

[3] HTTP – Hypertext Transfer Protocol

- Node as Container – Uzel však může obsahovat i další prvky, jako jsou komponenty nebo artefakty, a často se tak díky tomu používá jako jakýsi kontejner pro zobrazení komponent.
- Interface (Rozhraní) – Znázorňuje smluvní vztah. Objekty, které realizují rozhraní, musí splnit nějaký závazek.

Vzhledem k tomu, že diagram nasazení může kromě výše uvedeného zobrazovat i vztahy mezi komponentami (a jejich smluvní vztahy pomocí Interface), tak je v praktické části této práce upřednostněno použití pouze tohoto diagramu nasazení před použitím diagramu komponent. Díky tomu je zcela zřejmé, nejen jak probíhá komunikace mezi komponentami, ale i to, jaký vztah mají dané komponenty ke svým uzlům.

3.3.2 Volba architektury

Volba vhodné softwarové architektury závisí na mnoha faktorech, jako jsou funkční a nefunkční požadavky, očekávaný růst a změny systému, dostupné zdroje a zkušenosti týmu. Dobrá architektura by měla najít rovnováhu mezi komplexitou a flexibilitou a umožnit efektivní vývoj, testování a rozvoj systému v budoucnu.

Existuje několik známých přístupů ke tvorbě softwaru (architektonických stylů či vzorů), každý s vlastními výhodami a nevýhodami, přičemž tyto přístupy se nemusí vzájemně vždy vylučovat. Například je možné mít aplikaci v monolitickém stylu, avšak v podobě tří vrstvé architektury apod.

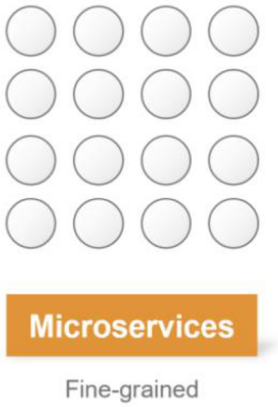
Monolitická architektura

Celá aplikace je vyvinuta jako jeden nerozdělitelný celek. Všechny komponenty jsou těsně provázané a často jsou spuštěny v jednom procesu. Tato architektura je zpočátku jednodušší na vývoj a nasazení, ale hůře se škáluje a při růstu složitosti systému se čím dál hůře udržuje. Nezanedbatelný problém může být také v pozdějších fázích životního cyklu softwaru při zaučování nových vývojářů do týmu, nebo také při nahrazování zastaralých technologií novými.

Obrázek 7: Monolitická architektura; Zdroj: (10)

Mikroservisní architektura

Na rozdíl od monolitické architektury je v tomto případě aplikace rozdělena na množství menších, nezávislých služeb, kde každá služba implementuje specifickou funkcionalitu a komunikuje s ostatními pomocí předem definovaného rozhraní (např. REST API). Hlavní výhodou je zjednodušení složitosti systému, takže jednotlivé služby pak jsou srozumitelnější, škálovatelnější a snadněji modifikovatelné, protože mohou být vyvíjeny, nasazovány a škálovány *nezávisle na sobě*.



Obrázek 8: Mikroservisní architektura; Zdroj: (10)

Klient-server model

Jedná se o starý známý způsob rozdělení systému na dvě hlavní části: klienta a server. Klient je zodpovědný za prezentaci dat a interakci s uživatelem (dnes široce používaný termín „*frontend*"), zatímco server obvykle zajišťuje logiku aplikace, přístup k datům

a komunikaci s dalšími službami (dnes široce používaný termín „*backend*“). Klient a server spolu komunikují přes síťové rozhraní pomocí API (nejčastěji pomocí HTTP protokolu). Jedná se v podstatě o dvouvrstvou architekturu, a ačkoliv dnes většina aplikací je ve skutečnosti vícevrstvá, tak přesto se pojmy jako frontend a backend široce ujaly i jako obecný způsob, jak jednoduše a rychle označit softwarové technologie podle toho, ve které oblasti jsou použity ve vztahu k uživateli.

Vícevrstvá architektura

Systém je organizován do několika vrstev, kde každá vrstva poskytuje služby vrstvě nad sebou a využívá služby vrstvy pod sebou. Nejčastější vrstvy jsou prezentační (zajišťující prezentaci dat uživateli), aplikační (někdy také známá jako servisní), doménová (která má na starosti byznys logiku) a datová (zajišťující přístup k databázi a uložení dat). Tato architektura vzhledem k výše uvedenému je zaměřena na oddělení zodpovědností a v posledních letech, zejména s rozvojem webových aplikací, zaznamenala velký úspěch.

3.4 Backendové technologie

Jak již bylo řečeno dříve, tak za backend je obecně považována ta část (webové) aplikace, která se nachází na serverové straně, a stará se proto především o zpracování dat, business logiku a práci s databází. Jinak řečeno je také backend zodpovědný za správné fungování aplikace a zajišťuje, že data jsou konzistentně a bezpečně uložena.

Při implementaci backendu se používá více technologií a programovacích jazyků. Mezi nejznámější jazyky se podle známého průzkumu „Stack Overflow Developer Survey 2023“ řadí například Java, PHP, Python, C# a JavaScript (Node.js) (11). V dnešní době už jsou ale prakticky nedílnou součástí všech webových aplikací také frameworky využívající daný jazyk (například: Express.js pro JavaScript, Spring pro Javu, Laravel pro PHP, Django pro Python atd.). Ze zmiňovaného průzkumu také vyplývá, že nejpoužívanější databáze jsou v dnešní době např. PostgreSQL, MySQL, MongoDB, Microsoft SQL Server ad. Každá z těchto technologií má samozřejmě své výhody a nevýhody a hodí se pro různé typy aplikací. Zažité je také použití tzv. *softwarových stacků*, což je obecně přijímaná kombinace vzájemně vhodných technologií.

I přesto, že softwarové stacky volbu vhodné kombinace backendových technologií v mnoha případech značně usnadňují, nemusí být volba technologií vždy úplně jednoduchá, a protože závisí na mnoha faktorech (například typu aplikace, její velikosti a složitosti, výkonnostních požadavcích a zkušenostech vývojářského týmu), tak je samozřejmě možné se odchýlit od daných zvyklostí a v odůvodněných případech je možné použít i méně oblíbenou / neobvyklou kombinaci.

3.4.1 Java

Java je všeobecně známý objektově orientovaný programovací jazyk vyvinutý společností *Sun Microsystems* v roce 1995 (od roku 2010 však vlastněný společností *Oracle*). Jedná se o jeden z nejpopulárnějších programovacích jazyků na světě, který byl navržen pro vývoj různých typů aplikací, od mobilních přes desktopové až po rozsáhlé podnikové systémy.

Programy napsané v Javě jsou kompilovány do tzv. bajtkódu, který může být spuštěn na jakémkoliv zařízení s nainstalovaným *Java Virtual Machine* (JVM). Díky tomu lze pak tyto aplikace přenášet mezi různými platformami teoreticky bez nutnosti změny kódu. Další výhodou Javy je její robustnost a bezpečnost. Java je také známá tím, že disponuje automatickou správou paměti díky tzv. „*garbage collectoru*“.

I když dříve byla univerzálnost (přenositelnost) javových aplikací jednou z hlavních výhod, v dnešní době, kdy se technologický svět posunul směrem ke kontejnerizaci, webovým řešením a chytrým telefonům, tak dnes už tato přenositelnost programů není zdaleka tak zásadní. Nicméně Java stále zůstává jedním z nejpopulárnějších programovacích jazyků pro vývoj podnikových aplikací.

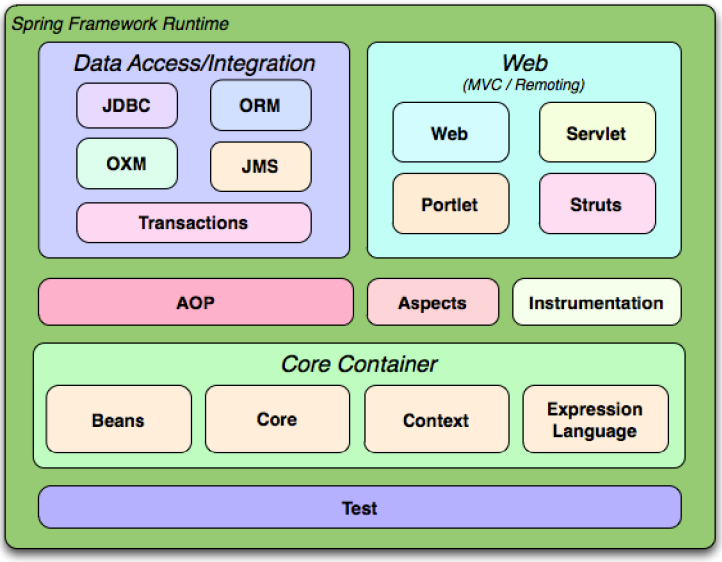
Java prošla od svého vzniku mnoha vylepšeními a změnami. S příchodem Javy 8 v roce 2014 byly představeny významné novinky, jako jsou lambda výrazy, streamy a funkcionální rozhraní, které usnadňují práci s kolekcemi a umožňují psát stručnější a čitelnější kód. Pozdější verze Javy přinesly další užitečné funkce (i když z pohledu autora práce ne tak zásadní) a k dnešnímu dni se používá v součtu již 23. verze v podobě Java 22.

Jednou z největších výhod Javy je její rozsáhlý ekosystém knihoven a frameworků. Ačkoliv Java samotná nabízí standardní knihovnu (Java Class Library), která sice poskytuje docela bohatou škálu funkcí, tak dnes mnoho z těchto funkcionalit přebírají frameworky, které mnohem více zrychlují a usnadňují vývoj tím, že poskytují již hotová řešení pro běžné problémy a úlohy. Jedním z nejpopulárnějších frameworků pro vývoj webových aplikací ve světě Javy je *Spring*.

3.4.2 Spring framework

Spring je open-source aplikační framework pro vývoj aplikací v jazyce Java, který byl vyvinut Rodem Johnsonem v roce 2003 a od té doby se stal jedním z nejpopulárnějších frameworků. Cílem bylo vytvořit framework, který dokáže usnadnit a zefektivnit vývoj robustních a škálovatelných aplikací. Toho bylo dosaženo díky využití návrhových vzorů jako *Inversion of Control* (IoC) a *Dependency Injection* (DI). Tyto vzory umožňují volné provázání komponent a zlepšují modularitu, testovatelnost a znovupoužitelnost kódu.

Jádrem Springu je tedy tzv. IoC kontejner, který se stará o životní cyklus objektů a jejich závislostí (12). Programátor definuje závislosti mezi objekty (tzv. *beans*) pomocí anotací nebo XML konfigurace a Spring už se sám stará o jejich životní cyklus a vzájemné provázání. Tím se minimalizuje množství zbytečného kódu (tzv. *boilerplate*) a zvyšuje se tak přehlednost a udržovatelnost aplikace.

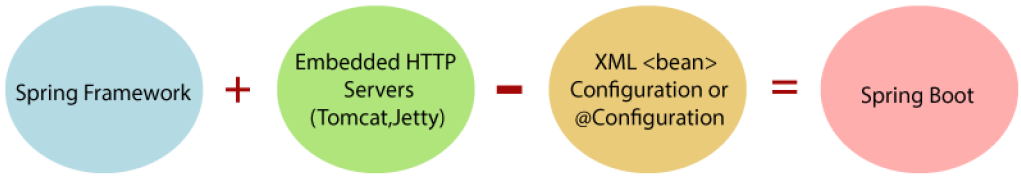


Obrázek 9: Přehled Spring framework; Zdroj: (20)

Spring také poskytuje širokou škálu modulů pro různé oblasti vývoje. Tyto moduly se dělí na Core Container, Data Access/Integration, Web, AOP (Aspect Oriented Programming), Instrumentation a Test, jak ukazuje Obrázek 9. Mezi nejpoužívanější moduly patří Spring MVC pro tvorbu webových aplikací, Spring Data pro práci s databázemi, Spring Security pro zabezpečení aplikací nebo Spring Cloud pro rychlý vývoj distribuovaných aplikací. Tyto moduly lze snadno integrovat a kombinovat podle potřeb konkrétní aplikace.

Značnou výhodou Springu je i jeho rozšíření a dobré povědomí v komunitě vývojářů. Spring také poskytuje kvalitní dokumentaci a na internetu je k dispozici mnoho návodů a ukázkových příkladů, což usnadňuje osvojení si frameworku a řešení běžných i komplikovanějších problémů.

Pro ještě větší urychlení a zjednodušení vývoje vznikl projekt Spring Boot, který je postavený na samotném frameworku Spring. Jedná se vlastně o běžný projekt frameworku Spring, který ale využívá tzv. „Spring Boot starters“ a auto-konfiguraci (13). Díky tomu a také díky přítomnosti vestavěného serveru (tzv. *Embedded serveru*), například v podobě *Apache Tomcat*, odpadá nutnost nasazovat aplikaci na externí server a ve výsledku tak umožnuje tzv. rychlý vývoj aplikací (RAD).



Obrázek 10: Jednoduché grafické znázornění pojmu Spring Boot; Zdroj: (21)

Vývojáři tak v podstatě stačí vytvořit jednoduchý projekt, přidat potřebné závislosti a může začít programovat business logiku. Všechny výše zmíněné vlastnosti významně urychlují vývoj a zkracují čas potřebný k uvedení aplikace do provozu.

3.4.3 REST a Richardson Maturity Model

Principy REST

REST[4] je architektonický styl pro navrhování distribuovaných systémů. Byl představen Royem Fieldingem v roce 2000 a definuje sadu omezení a vlastností, které by mělo API splňovat, aby bylo považováno za RESTful.

Dnes všeobecně známé klíčové principy REST jsou:

- Jednotné rozhraní (Uniform Interface) – server vystavuje prostředky klientovi jednotným známým způsobem a skládá se dle Fieldinga z následujících čtyř principů:
    - Identifikace zdrojů (Identification of resources)
    - Manipulace se zdroji skrze jejich reprezentace (Manipulation of resources through representations)
    - Samopopisující zprávy (Self–descriptive messages)
    - Hypermedia jako aplikační stav (HATEOAS[5])
- Bezstavovost (Stateless) – každý požadavek musí obsahovat všechny informace nutné pro jeho zpracování, server neudržuje stav mezi požadavky
- Klient / Server (Client / Server) – klient a server jsou nezávislí a odděleni pomocí rozhraní
- Vícevrstvý systém (Layered System) – mezi klientem a serverem může být více mezilehlých vrstev (např. proxy, load balancer), klient o nich nemusí vědět
- Cachování (Caching) – server může podporovat využití mezipaměti

Richardson Maturity Model

Richardson Maturity Model (RMM) je způsob, jak ohodnotit vyspělost API z pohledu dodržování REST principů. Byl navržen Leonardem Richardsonem (14) a rozděluje návrh API do 4 úrovní (0 až 3), kde vyšší úroveň znamená lepší soulad s principy REST.

---

[4] REST – Representational State Transfer
[5] HATEOAS – Hypermedia As The Engine of Application State

Jednotlivé úrovně RMM jsou (viz Obrázek 11):

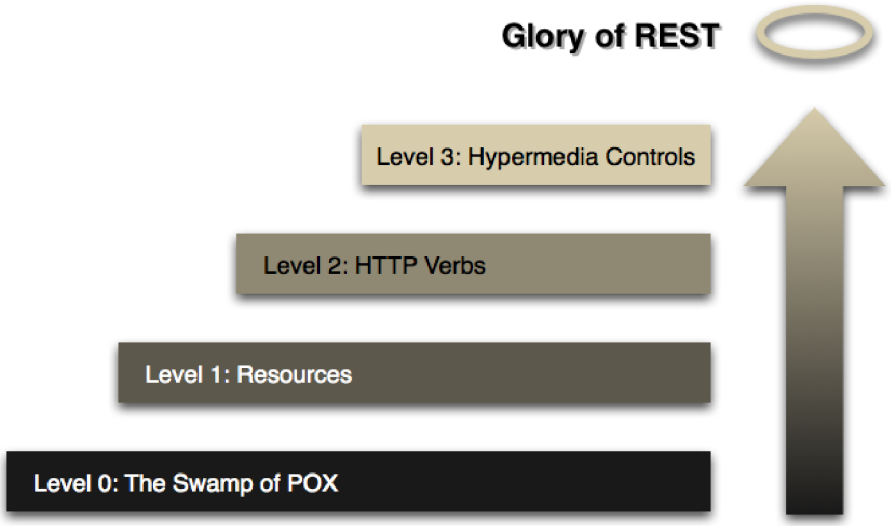
- Nultá úroveň (The Swamp of POX[6]) – Odpovídá nejméně stylu architektury REST. API používá pouze jediný identifikátor URI[7] a typicky HTTP s metodou POST. Zprávy mohou být v mnoha různých formátech (např. XML[8], JSON[9] a jiných textových formátech). Jedná se i o protokol SOAP[10].
- První úroveň (Resources) – API používá více URI pro identifikaci zdrojů, ale stále jen jednu HTTP metodu.
- Druhá úroveň (HTTP verbs) – API používá správné HTTP metody (viz níže) pro manipulaci se zdroji namísto jedné. Samotné názvy zdrojů smí obsahovat jen podstatná jména a žádná slovesa, protože akci, kterou chceme provést, vyjadřujeme právě pomocí tzv. HTTP verbs (splňující CRUD[11]):
    - GET – získání zdroje (Read)
    - POST – vytvoření zdroje (Create)
    - PUT – aktualizace zdroje (Update)
    - DELETE – smazání zdroje (Delete)
    - PATCH – částečná úprava zdroje (Update)

  Používají se také tzv. návratové (stavové) kódy, aby si klient mohl zkontrolovat výsledek operace.
- Třetí úroveň (Hypermedia controls) – Jedná se o nejvyspělejší úroveň API. Napomáhá automatizaci klientské strany díky dynamickému vytváření odkazů. Využívá se tzv. „hypermedia links“ (HATEOAS) pro navigaci mezi zdroji (a případně i popis možných akcí). V praxi se až tolik nevyužívá, protože se spíše dává přednost vývoji podle dokumentace.

Pouze API na třetí úrovni je dle RMM považováno za skutečně RESTful. RMM klasifikuje kvalitu API z pohledu REST principů a tím napomáhá identifikovat nedostatky v návrhu a navrhnout možná vylepšení.

---

[6] POX – Plain Old XML
[7] URI – Uniform Resource Identifier
[8] XML – Extensible Markup Language
[9] JSON – JavaScript Object Notation
[10] SOAP – Simple Object Access Protocol
[11] CRUD – Create, Read, Update, Delete



Obrázek 11: Richardson Maturity Model; Zdroj: (14)

3.5 Keycloak

Keycloak je open-source řešení pro tzv. správu identit a přístupu (IAM), které poskytuje funkce jednotného přihlašování (SSO[12]), autentizace a správy uživatelů pro moderní aplikace a služby (15). Keycloak funguje jako tzv. poskytovatel identity (IdP[13]), což znamená, že spravuje identitu uživatelů a poskytuje informace o jejich autentizaci a atributech ostatním službám a aplikacím, které se označují jako poskytovatelé služeb (SP[14]). Tyto externí aplikace (v roli SP) pak spoléhají na IdP pro ověření identity uživatelů a získávají jeho prostřednictvím potřebné informace pro řízení přístupu, a to díky podpoře standardních protokolů jako je např. *OpenID Connect* (OIDC) a OAuth 2.0.

Keycloak také nabízí celou řadu moderních a zajímavých funkcí (například vícefázové ověření, federaci identit s externími poskytovateli a mnoho dalšího), a díky svým vlastnostem a cenové dostupnosti (open-source), tak nalézá využití od malých aplikací až po podnikové systémy.

---

[12] SSO – Single Sign-On
[13] IdP – Identity Provider
[14] SP – Service Provider

3.6 Frontendové technologie

Jak již bylo uvedeno v kapitole 3.3.2, tak tzv. *frontend* je klientská část webové aplikace, která zajišťuje uživatelské rozhraní a interakci s uživatelem. Využívá se při tom v oblasti softwarového vývoje dnes všeobecně známých technologií jako jsou HTML (pro strukturu obsahu), CSS (pro grafický design a layout) a JavaScript (pro interaktivitu a dynamické chování) a dalších (11). A proto tyto všeobecně známé technologie nejsou v této práci více popisovány.

Pro usnadnění vývoje moderních webových aplikací se dnes velice často používají frontendové frameworky a knihovny. Mezi nejznámější řadíme například knihovnu React (od společnosti Facebook), framework Angular (od společnosti Google) a Vue.js. React je knihovna zaměřená na tvorbu znovupoužitelných komponent s využitím deklarativního přístupu. Angular je komplexní framework vhodný i pro složité webové aplikace, který byl autorem vybrán pro praktickou část této práce, a proto bude čtenáři více popsán v následující kapitole. Vue.js vhodně kombinuje některé koncepty z Reactu a Angularu.

3.6.1 Angular

Angular je moderní platforma pro webové aplikace, která poskytuje vývojářům ucelenou sadu nástrojů a možností pro vytváření rozsáhlých a robustních aplikací (16). Jedná se v podstatě o komplexní open-source framework, který je založený na komponentovém přístupu. Jeho hlavním programovacím jazykem je jazyk TypeScript, který vývojářům oproti JavaScriptu poskytuje statické typování a lepší podporu pro objektově orientované programování.

Jak již bylo zmíněno, tak základním stavebním kamenem Angularu jsou komponenty. Každá komponenta se skládá z HTML šablony, CSS stylů a třídy definující chování komponenty v TypeScriptu. Tyto komponenty jsou znovupoužitelné a lze je skládat dohromady pro vytvoření propracovaného uživatelského rozhraní.

Angular nabízí velké množství dalších funkcionalit, jako například: služby pro sdílení dat mezi komponentami, dependency injection pro správu závislostí, routování a také Angular CLI pro efektivní vývoj a sestavování aplikací.

3.7 Databáze

Databáze je soubor informací, jako jsou znaky, čísla, diagramy, jejichž systematická struktura umožňuje, aby tyto informace mohly být vyhledávány pomocí počítače (17). Databáze se dělí podle způsobu ukládání dat na několik typů, ale nejrozšířenějším typem jsou relační databáze, které jsou založeny na relačním modelu.

Databáze hrají důležitou roli v backendových aplikacích, ve kterých slouží ke spolehlivému ukládání, dotazování a správě dat. V této kapitole se zaměříme na relační databáze (konkrétně PostgreSQL), také na objektově–relační mapování pomocí Java Persistence API (JPA) a vazby na framework Spring.

3.7.1 Relační databáze a PostgreSQL

Relační databáze jsou založeny na relačním modelu dat, kde jsou data organizována do tabulek (relací) a vztahy mezi nimi jsou vyjádřeny pomocí cizích klíčů. Tento model podporuje dotazování pomocí strukturovaného dotazovacího jazyka SQL[15].

PostgreSQL je výkonný open–source objektově–relační databázový systém s více než 35 lety aktivního vývoje. Vyniká svou spolehlivostí, robustností, škálovatelností a díky open–source řešení i cenovou dostupností, což z něj dělá oblíbenou volbu pro vývoj backendových aplikací. PostgreSQL také podporuje pokročilé datové typy a umožňuje psaní procedur a funkcí v různých populárních jazycích. Dále také nabízí pokročilé možnosti indexování a optimalizace dotazů.

Z výše uvedených důvodů byla v praktické části práce zvolena tato databáze pro vývoj aplikace potravinového agregátoru.

[15] SQL – Structured Query Language

3.7.2 Objektově relační mapování, JPA a Spring Data JPA

Objektově relační mapování (ORM) je technika pro převod dat mezi relační databází a objektově orientovaným programovacím jazykem jako je například Java. ORM umožňuje pracovat s daty v databázi pomocí objektů a není tak nutné psaní SQL dotazů. Vývojář je do značné míry odstíněn od datové vrstvy a konkrétní relační databáze a díky tomu se může více soustředit na samotnou business logiku. Další výhodou ORM je v konečném důsledku také snadnější údržba samotného kódu. Nevýhodou může být režie způsobená mapováním objektů a samozřejmě horší optimalizace dotazů.

Java Persistence API (JPA) je jednoduše řečeno rozhraní pro ORM v Javě. JPA poskytuje jednotný způsob, jak definovat mapování za využití anotací nebo XML. Entity v JPA reprezentují databázové tabulky a atributy entit odpovídají sloupcům. JPA také obsahuje dotazovací jazyk JPQL, který je platformově nezávislý. Mezi nejznámější implementace JPA patří například EclipseLink a Hibernate, který je výchozí volbou ve Spring Boot.

V aplikacích postavených na Spring Boot se často používá nadstavba nad JPA s názvem ***Spring Data JPA***. Ta přináší další zjednodušení a abstrakci v podobě tzv. repozitářů, které zapouzdřují databázové operace pro konkrétní entity. Vývojář jen definuje rozhraní repozitáře a Spring Data JPA automaticky generuje jeho implementaci se základními metodami pro CRUD operace a dotazování, a tím se ještě více redukuje množství kódu, které vývojář musí napsat. Tento přístup podporuje i pokročilé funkce jako je stránkování, řazení nebo dynamické dotazy odvozené z názvů metod apod.

3.8 Mocking

Mocking je často využívaná technika v oblasti vývoje softwaru, která spočívá ve vytváření zjednodušených náhrad za skutečné objekty, komponenty nebo služby. Tyto tzv. mocky napodobují chování skutečných entit a lze je snadno přizpůsobit různým situacím. Mocking se využívá v různých fázích vývoje softwaru, od prototypování přes testování a ladění až po prezentaci hotových softwarových produktů.

Jedním z hlavních důvodů pro použití mockování je oddělení určité části systému (komponent, služeb apod.) od původních závislostí a jejich nahrazení vlastními

náhradami (mocky). Díky tomu je možné například vyvíjet a testovat různé části nezávisle na sobě (v různých scénářích), a je tak tedy možné jednoduše ověřit, zda testovaná (vyvíjená) část funguje správně, aniž by bylo nutné při vývoji využívat produkční prostředí okolních systémů (komponent, služeb apod.) a také spoléhat na jejich dostupnost a funkčnost.

V neposlední řadě mockování umožňuje paralelní práci na vývoji vzájemně závislých komponent i přesto, že některé části ještě nejsou zcela implementovány. Uplatnění nalézá také při prototypování a návrhu API, testování bezpečnosti, výkonnosti a škálovatelnosti systému, kde mocky simulují různé zátěžové scénáře a umožňují ověřit chování aplikace při vysoké zátěži.

4 Praktická část práce

Jak již bylo zmíněno v teoretické části v kapitole 3.1.2 (Srovnávací portály), tak v současné době neexistuje na trhu agregátor, který by byl schopen v reálném čase a na jednom místě porovnat online ceny potravin příslušných e-shopů, a to navíc s možností přidat si vybrané zboží rovnou do nákupního košíku konkrétního online supermarketu.

Tato kapitola se věnuje popisu současného stavu nákupního procesu vybraných e-shopů a představuje řešení „mezery na trhu“ v podobě návrhu a implementace *agregátoru e-shopů s potravinami*, který sdružuje online supermarkety Rohlik.cz a Billa.cz. Rozšíření této aplikace v budoucnu o další jejich online konkurenci, je možné bez výrazných nároků na pracnost, a to díky vhodnému návrhu této aplikace — je především nutné dobře znát API nového konkurenčního e-shopu.

Kromě zmíněné hlavní funkčnosti, real-time agregace výsledků hledání napříč připojenými e-shopy, nabízí tato aplikace i další zajímavé a užitečné funkce, jejichž bližší popis je uveden v následující kapitole.

4.1 Analýza a sběr požadavků

Analyzované požadavky jsou rozděleny do dvou kategorií podle všeobecně přijímaných standardů:

- Funkční požadavky (FR) – jedná se o požadavky, které koncový uživatel konkrétně požaduje po systému a odpovídají na otázku *„co“* má systém dělat. Jsou to požadavky uvedené zadavatelem, které lze na rozdíl od nefunkčních požadavků vidět přímo ve výsledném produktu.
- Nefunkční požadavky (NFR) – odpovídají na otázku *„jak“* to systém má dělat (požadavky na architekturu aplikace, uživatelské rozhraní, prostředí pro běh aplikace apod.)

4.1.1 Funkční požadavky

FR_1: Správa uživatelských účtů

Systém musí umožňovat registraci nových uživatelů, přihlašování a odhlašování existujících uživatelů, a také resetování zapomenutého hesla. Registrace bude vyžadovat zadání základních údajů. Po úspěšném přihlášení bude mít uživatel přístup k funkcím vyžadujícím autentizaci. Odhlášením dojde k ukončení relace a zneplatnění přihlašovacích tokenů.

FR_2: Správa přihlašovacích údajů k obchodům

Přihlášený uživatel bude moci v systému spravovat své přihlašovací údaje k jednotlivým napojeným online obchodům s potravinami. V prvotní fázi jen Rohlik.cz a Billa.cz. Tyto údaje budou sloužit k automatizovanému přihlášení do konkrétního obchodu za účelem provedení nákupu. Uživatel bude moci přidávat nové přihlašovací údaje, editovat je nebo odebírat. Pro každý online obchod bude možné přidat maximálně jeden přihlašovací údaj.

FR_3: Agregované vyhledávání produktů

Systém bude umožňovat přihlášenému uživateli zadat vyhledávaný produkt podle názvu a prohledá databáze všech napojených e-shopů. Výsledky z jednotlivých e-shopů budou agregované do jedné stránky a zobrazeny uživateli. U každého produktu se zobrazí základní informace, jako je název produktu, jeho cena, cena za jednotkové množství a množství. Dále se ještě zobrazí odkaz na produkt do příslušného e-shopu, kde bude možné si zobrazit jeho detailní popis.

FR_4: Tvorba a správa nákupních seznamů

Přihlášený uživatel bude moci vytvářet a spravovat své nákupní seznamy. Do seznamu bude moci přidávat produkty z výsledků vyhledávání. U položek na seznamu bude možné měnit jejich množství nebo je přímo odebírat. Seznamy bude možné pojmenovávat a mazat. Systém nebude zobrazovat celkovou cenu položek na seznamu.

FR_5: Hlídání cen produktů

Přihlášený uživatel si bude moci nastavit hlídacího psa na konkrétní produkt pomocí agregovaného vyhledávání. Systém bude disponovat připraveným API, díky kterému bude možné monitorovat v pravidelných intervalech cenu hlídaných produktů u připojených e-shopů a při dosažení cílové ceny (či ještě nižší) zašle uživateli upozornění (notifikaci) na jeho email. Uživatel bude moci hlídací psy přidávat, editovat a mazat.

FR_6: Provedení nákupu

Přihlášený uživatel uvidí v aplikaci obsah svého košíku pro každý připojený e-shop zvlášť. Přihlášený uživatel si vybere pomocí sjednoceného vyhledávání produkt a přidá si ho pomocí aplikace do košíku (v příslušném e-shopu), bude moci měnit u jednotlivých položek jejich množství nebo je také rovnou odebrat. Aplikace mu také umožní rovnou smazat celý obsah košíku v daném e-shopu. Pokud bude uživatel spokojený s výběrem, pak dokončení nákupu proběhne přímo v prostředí příslušného e-shopu.

FR_7: Kopírování košíku do nákupního seznamu

Přihlášený uživatel bude mít možnost zkopírovat obsah košíku všech připojených e-shopů do svého nákupního seznamu v aplikaci. Tato funkcionalita umožní uživateli uložit si rozdělaný nákup pro pozdější dokončení nebo úpravu, či případně jako šablonu pro opakované využití. Systém zkopíruje všechny položky a jejich množství z košíku e-shopu do nově vytvořeného nákupního seznamu uživatele.

FR_8: Kopírování nákupního seznamu do košíku

Systém umožní přihlášenému uživateli zkopírovat obsah svého nákupního seznamu do košíku v připojených e-shopech. Tím se uživateli usnadní proces objednávky, protože nebude muset vyhledávat a vkládat jednotlivé položky do košíku ručně. Systém zajistí správné přenesení produktů a množství z nákupního seznamu do košíku cílového e-shopu.

4.1.2 Nefunkční požadavky

NFR_1: Architektura aplikace

Aplikace bude implementována jako webová aplikace s využitím frameworku Angular pro frontend a Spring Boot pro backend. Tato architektura zajistí oddělení prezentační vrstvy od aplikační a datové logiky, což zlepší udržovatelnost a škálovatelnost aplikace, a především usnadní případný další rozvoj. Backend bude poskytovat REST API pro komunikaci s frontendem.

NFR_2: Databázový systém

Pro ukládání dat bude použita objektově-relační open-source databáze PostgreSQL. Databáze bude provozována v Docker kontejneru, což usnadní nasazení a správu v různých prostředích. Pro přístup k databázi bude využit ORM framework.

NFR_3: Autentizace a autorizace

Autentizace a autorizace uživatelů bude zajištěna pomocí externího IdP systému Keycloak. Keycloak poskytuje robustní řešení pro správu identit a přístupových práv. Bude nasazen v Docker kontejneru a integrován s frontendovou i backendovou částí aplikace pomocí standardního protokolu OIDC. Identita v backendové části bude získána z tokenu.

NFR_4: Responzivní design

Uživatelské rozhraní aplikace bude navrženo s ohledem na responzivní design. Aplikace bude optimalizována pro různá zařízení a velikosti obrazovek, aby poskytovala dostatečný uživatelský zážitek jak na desktopu, tak i na chytrém telefonu.

NFR_5: Výkon

Aplikace bude přinejmenším schopna zvládat běžnou zátěž odpovídající jednotkám přihlášených uživatelů.

NRF_6: Dostupnost

Systém bude provozován v režimu 24/7. Jeho dostupnost bude minimálně 98 %.

NRF_7: Anglická lokalizace

Veškeré uživatelské prostředí aplikace bude lokalizováno do anglického jazyka.

4.1.3 Aktéři a případy užití

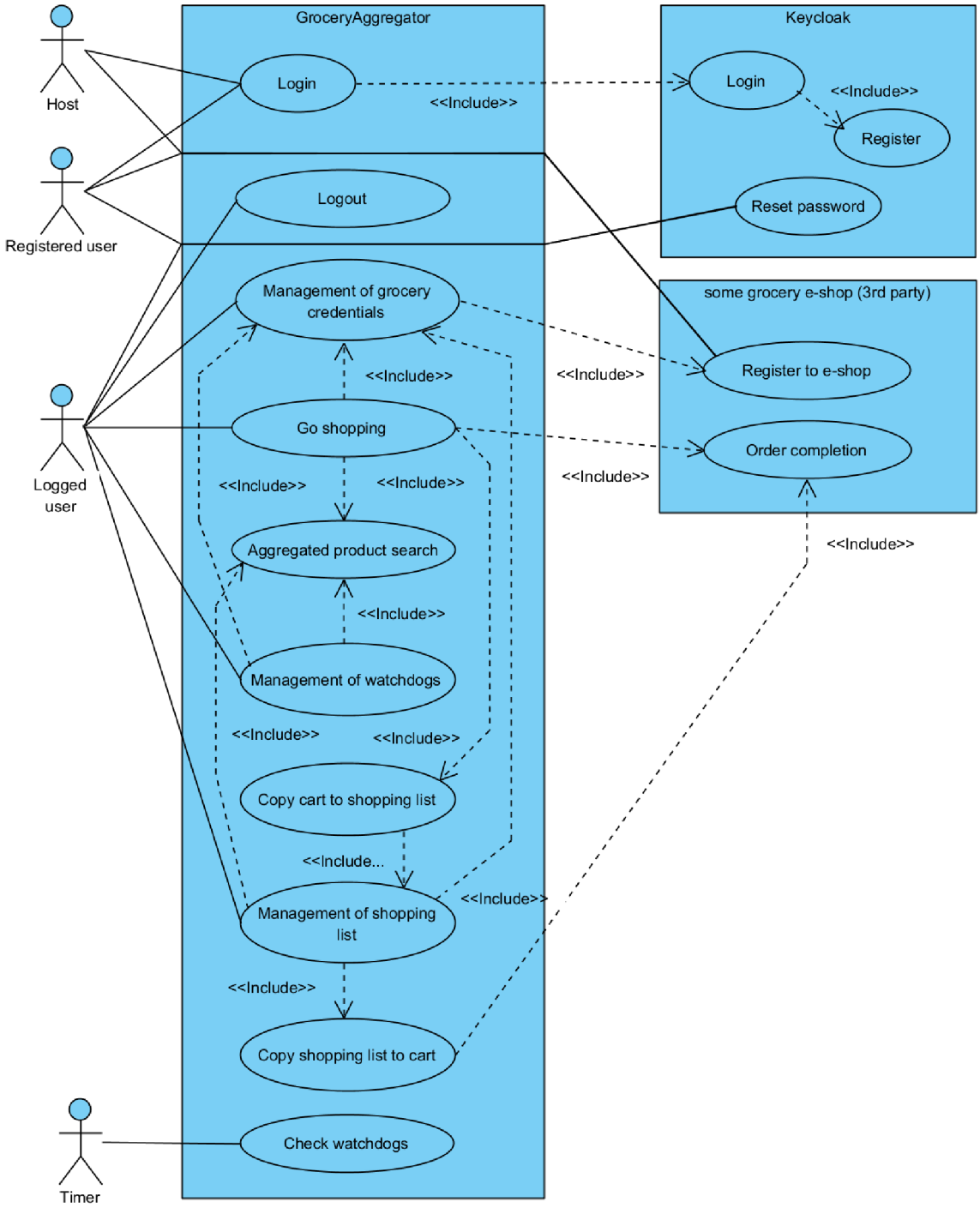


Diagram 1: Use case diagram; Zdroj: vlastní tvorba

Tato kapitola představuje identifikované účastníky (tzv. aktéry), kteří mohou navrhovaný systém využívat (navrhovaná aplikace „GroceryAggregator“ a k tomu přidružený systém IdP Keycloak) a případy užití (use cases), které mohou účastníci provádět. Vztahy mezi nimi a případy užití jsou zobrazeny na předchozím diagramu (viz Diagram 1).

Aktéři

Byli identifkováni následující čtyři různí aktéři, kteří mohou pracovat s navrhovanou aplikací:

1. **Host** – Aktér, který představuje anonymního návštěvníka webové aplikace, který dosud nemá registraci v přidruženém systému Keycloak.
2. **Registered user** – Aktér, který představuje registrovaného uživatele v systému Keycloak (který je přidružen k aplikaci) a přitom není přihlášen do aplikace.
3. **Logged user** – Aktér, který představuje zaregistrovaného uživatele v systému Keycloak (který je přidružen k aplikaci) a je již přihlášen do aplikace.
4. **Timer** – Aktér, který představuje systémového (technického) uživatele, který bude pomocí API provádět v pravidelných intervalech volání daného REST endpointu.

Případy užití

UC_1: Registrace uživatele (Register)

Aktér: Host

Popis: Neregistrovaný uživatel (host) může vytvořit nový uživatelský účet v systému Keycloak. Uživatel zadá požadované údaje jako jméno, příjmení, e-mail, uživatelské jméno a heslo. Po odeslání registračního formuláře systém Keycloak vytvoří nový uživatelský účet (jednoznačně identifikovaný uživatelským jménem a e-mailem) a uživatel se tak stává registrovaným uživatelem systému Keycloak.

UC_2: Přihlášení uživatele (Login)

Aktér: Host, Registered user

Popis: Registrovaný uživatel se může přihlásit do aplikace tím, že klikne na tlačítko „*Login by SSO*“, tím je přesměrován do IdP systému Keycloak a v něm se po vyplnění svých přihlašovacích údajů (uživatelské jméno a heslo) přihlásí kliknutím na

tlačítko „*Sign In*“. Pokud se jedná o hosta, který ještě není registrován v Keycloak, pak provede registraci (UC_1). Po úspěšném ověření údajů v systému Keycloak je uživatel přihlášen a následně proběhne automatické přesměrování zpět do aplikace GroceryAggregator. Ta ověří získaný token, a v případě úspěchu je ověření považováno za dokončené, a v aplikaci se tak uživatel stává přihlášeným uživatelem.

UC_3: Správa přihlašovacích údajů k potravinovým e-shopům (Management of grocery credentials)

Aktér: Logged user

Popis: Přihlášený uživatel může v aplikaci spravovat své přihlašovací údaje k jednotlivým napojeným online obchodům s potravinami. Uživatel může přidávat nové přihlašovací údaje k dostupným e-shopům, editovat stávající nebo je odebírat. Ke každému e-shopu je možné mít přiřazeny maximálně jedny přihlašovací údaje. Pokud přihlášený uživatel není registrován u žádného e-shopu a nemá tedy žádné přihlašovací údaje, tak se musí nejprve zaregistrovat pomocí UC_20.

UC_4: Vyhledávání produktů (Aggregated product search)

Aktér: Logged user

Popis: Přihlášený uživatel může vyhledávat produkty napříč všemi napojenými e-shopy. Zadá hledaný výraz a aplikace mu zobrazí agregované výsledky ze všech napojených e-shopů. U každého produktu se zobrazí základní informace, jako je název produktu, jeho cena, cena za jednotkové množství, samotné množství a odkaz na produkt do příslušného e-shopu.

UC_5: Správa nákupních seznamů (Management of shopping list)

Aktér: Logged user

Popis: Přihlášený uživatel může vytvářet, upravovat a odstraňovat své nákupní seznamy, které pro lepší organizaci lze také pojmenovávat. Pokud je se seznamem spokojen, je možné ho stisknutím tlačítka „*Load to cart*“ načíst do košíku u příslušného e-shopu (UC_10). Uživatel po kliknutí na tlačítko „*Detail*“ může do konkrétního seznamu přidávat vybrané produkty z výsledků agregovaného vyhledávání (UC_4). U položek na konkrétním seznamu lze měnit jejich množství nebo je rovnou odstraňovat. Pokud by

z nějakého důvodu přihlášený uživatel neměl zadán žádný přihlašovací údaj k e-shopům, aplikace ho vyzve k jeho přidání (UC_3).

UC_6: Správa hlídacích psů (Management of watchdogs)

Aktér: Logged user

Popis: Přihlášený uživatel může pomocí agregovaného vyhledávání (UC_4) vytvářet a spravovat hlídací psy na konkrétní produkty z připojených e-shopů. Hlídací pes pak monitoruje cenu produktu v příslušném e-shopu a při poklesu ceny pod stanovenou cenovou hranici informuje uživatele e-mailem. Uživatel může hlídací psy vytvářet, upravovat a mazat. Pokud by z nějakého důvodu přihlášený uživatel neměl zadán žádný přihlašovací údaj k e-shopům, aplikace ho vyzve k jeho přidání (UC_3).

UC_7: Nakupování (Go shopping)

Aktér: Logged user

Popis: Jedná se v podstatě o hlavní případ užití této aplikace. Aplikace přihlášenému uživateli nejprve na pozadí zařídí automatizované přihlášení do jím připojených e-shopů a díky tomu jsou načteny obsahy košíků pro každý připojený e-shop zvlášť. Pokud má košík prázdný, může použít agregované vyhledávání (UC_4) a vybraný produkt si přidat do příslušného košíku. Pokud má košík již něčím naplněný, tak může upravit jeho obsah, tzn. přidat produkt(y), upravit množství položek pro jednotlivé produkty nebo jej může uložit jako nákupní seznam (UC_12). Dále také uživatel může položky odstraňovat či rovnou smazat obsah celého košíku. Pokud je uživatel spokojený s výběrem, pak pokračuje pomocí UC_21. Pokud by z nějakého důvodu přihlášený uživatel neměl zadán žádný přihlašovací údaj k e-shopům, aplikace ho vyzve k jeho přidání (UC_3).

UC_8: Odhlášení uživatele (Logout)

Aktér: Logged user

Popis:

Přihlášený uživatel se může odhlásit z aplikace kliknutím na profilovou ikonu v horním pravém rohu obrazovky a výběrem „*Logout*“ a tím dojde k ukončení jeho relace a odhlášení. Po odhlášení se stává opět jen registrovaným uživatelem.

UC_9: Reset hesla (Reset password)

Aktér: Registered user, Logged user

Popis: Registrovaný nebo přihlášený uživatel si může zažádat o reset svého zapomenutého hesla. V systému Keycloak vyplní svůj e-mail nebo uživatelské jméno a systém Keycloak mu zašle instrukce pro vytvoření nového hesla. Po úspěšném resetování hesla se uživatel může opět přihlásit do GroceryAggregatoru se svými novými přihlašovacími údaji.

UC_10: Kopírování nákupního seznamu do košíku (Copy shopping list to cart)

Aktér: Logged user

Popis: Přihlášený uživatel klikne na tlačítko „*Load to cart*“ a po potvrzení uživatelem se aplikace přes API a pomocí zadaných přihlašovacích údajů přihlásí do všech potřebných e-shopů a zkopíruje obsah uživatelem zvoleného nákupního seznamu do košíku v příslušném e-shopu. Aplikace nejprve vysype košík v tomto e-shopu a poté přidá položky z nákupního seznamu do košíku. Tím se uživateli usnadní proces objednávky a nemusí vyhledávat a vkládat položky do košíku ručně.

UC_11: Kontrola hlídacích psů (Check watchdogs)

Aktér: Timer

Popis: Aplikace při registraci API volání zkontroluje ceny produktů, na kterých si uživatelé nastavili nějakého hlídacího psa. Pokud se cena produktu sníží pod nastavenou hranici, aplikace si tuto informaci zaznamená. Po dokončené kontrole odešle všem dotčeným uživatelům sdruženou notifikaci (formou emailu) o změnách cen produktů. Zpráva obsahuje název produktu, cenu, definovanou cenovou hranici a odkaz na produkt.

UC_12: Kopírování košíku do nákupního seznamu (Copy cart to shopping list)

Aktér: Logged user

Popis: Přihlášený uživatel, pokud má košík již něčím naplněný, může jeho obsah kliknutím na tlačítko „Save all as new shopping list“ pojmenovat a uložit jako nákupní seznam. Aplikace se přes API a pomocí zadaných přihlašovacích údajů přihlásí do všech potřebných e-shopů a zkopíruje aktuálně přihlášenému uživateli obsahy košíků ze všech připojených e-shopů do jeho zvoleného nákupního seznamu v aplikaci. Uživatel tak může snadno uložit rozdělaný nákup pro pozdější objednání.

Následující případy užití se týkají vyvíjené aplikace nepřímo, protože jsou součástí externích aplikací (připojených e-shopů).

UC_20: Registrace na e-shopu (Register to e-shop)

Aktér: Host, Registered user, Logged user

Popis: Uživatel provede registraci na e-shopu (online supermarketu) třetí strany. Tento use case je pro každý e-shop různý a nesouvisí přímo s touto aplikací, a proto ani nemůže být do detailu popsán.

UC_21: Dokončení objednávky na e-shopu (Order completion)

Aktér: Logged user

Popis: Pokud je přihlášený uživatel spokojený s výběrem, pak po kliknutí na odkaz přejde do vybraného e-shopu a tam s již předvyplněným košíkem dokončí nákup. Zbytek tohoto use case je pro každý e-shop různý a nesouvisí přímo s touto aplikací, a proto ani nemůže být do detailu popsán.

4.1.4 Mapování případů užití na funkční požadavky

Následující tabulka (viz Tabulka 1) je určená ke kontrole splnění všech funkčních požadavků (FR) vzhledem k případům užití (UC). Z výsledné tabulky vyplývá, že všechny funkční požadavky jsou naplněny alespoň jedním případem užití i po odečtení UC2 (zajišťující přihlášení).

FR\UC	UC_1	UC_2	UC_3	UC_4	UC_5	UC_6	UC_7	UC_8	UC_9	UC_10	UC_11	UC_12	UC_20	UC_21
FR_1	X	X						X	X					
FR_2		X	X										X	
FR_3		X		X									X	
FR_4		X			X								X	
FR_5		X		X		X					X		X	
FR_6		X		X			X						X	X
FR_7		X										X	X	
FR_8		X								X			X	X

Tabulka 1: Mapování funkčních požadavků na případy užití; Zdroj: vlastní tvorba

4.2 Analýza API vybraných e-shopů s potravinami

Tato kapitola je zaměřena na analýzu API dvou českých online supermarketů, a sice Rohlik.cz a Billa.cz. Cílem je tedy díky tomu získat obecný přehled a dostatek informací (způsob přihlášení, práce s nákupním košíkem, vyhledávání, produktové informace a další) k implementaci potravinového agregátoru.

I když ani Rohlik.cz ani Billa.cz oficiální veřejné API neposkytují, tak jejich webové aplikace jsou vhodně rozděleny na frontend a backend, přičemž backend API mají veřejně přístupné. I když tyto API jsou tedy určeny pro komunikaci s jejich vlastním frontendem, dají se s úspěchem využít i pro potřeby potravinového agregátoru.

Nese to však s sebou jisté komplikace:

- i když je API veřejně dostupné, není oficiálně dokumentované,
- API se také může kdykoli změnit a aplikace třetích stran, které toto API využívají, tak mohou snadno přestat správně fungovat.

4.2.1 Rohlik API

Rohlik.cz poskytuje REST API s JSON formátem pro komunikaci. Základní URL pro přístup k API je „`https://www.rohlik.cz`“. Autentizace je řešena pomocí session cookie[16], která je zasílána v hlavičce každého požadavku. Cookies jsou malé textové soubory vytvořené konkrétní webovou stránkou, které jsou uložené v prohlížeči uživatele.

API rohlíku nabízí celou škálu různých endpointů[17] pro komplexní použití. Pro vývoj této aplikace však postačovalo použít pouze vybrané (viz následující tabulka).

Dostupné endpointy a jejich využití

Metoda	Endpoint	Popis
POST	`/services/frontend-service/login`	Přihlášení uživatele
POST	`/services/frontend-service/logout`	Odhlášení uživatele
GET	`/services/frontend-service/search-metadata`	Vyhledávání produktů
GET	`/services/frontend-service/v2/cart`	Získání obsahu košíku
POST	`/services/frontend-service/v2/cart`	Přidání produktu do košíku
PUT	`/services/frontend-service/v2/cart`	Změna množství produktu v košíku
DELETE	`/services/frontend-service/v2/cart`	Odebrání produktu z košíku
DELETE	`/services/frontend-service/v2/cart?clear=true`	Vysypání košíku
GET	`/api/v1/products`	Získání detailů produktů
GET	`/api/v1/products/prices`	Získání cen produktů
GET	`/api/v1/products/stock`	Získání dostupnosti produktů

Tabulka 2: Dostupné endpointy Rohlik API; Zdroj: vlastní tvorba

Z přehledu je patrné, že API e-shopu Rohlik.cz nabízí dostatečné množství endpointů pro potřeby potravinového agregátoru.

4.2.2 Billa API

Billa.cz také poskytuje REST API s JSON formátem pro komunikaci. Základní URL pro přístup k API je „`https://shop.billa.cz/api`“. Autentizace je také řešena pomocí cookies, které jsou zasílány v hlavičce každého požadavku.

[16] session cookie – je dočasný soubor cookie, který existuje pouze po dobu trvání relace uživatele
[17] endpoint – koncový bod API

Na rozdíl od Rohlik API používá Billa API pro zabezpečení mechanismus CSRF[18] (konkrétně „*Cookie-to-header token*“). Před voláním endpointů, které mění stav na serveru (např. přidání produktu do košíku), je nutné nejprve zavolat endpoint „`/csrf`“, který vygeneruje CSRF token a vrátí ho klientovi v cookies. Jedná se o častou bezpečnostní praktiku v moderních webových aplikacích.

Dostupné endpointy a jejich využití

Metoda	Endpoint	Popis
GET	`/csrf`	Získání CSRF tokenu
POST	`/auth/login`	Přihlášení uživatele
POST	`/auth/logout`	Odhlášení uživatele
GET	`/products/search/{query}`	Produktové vyhledávání
GET	`/carts`	Získání obsahu košíku
POST	`/carts/items`	Přidání produktu do košíku / Změna množství
DELETE	`/carts/items/multiple`	Odebrání produktu z košíku
DELETE	`/carts/items/all`	Vysypání košíku

Tabulka 3: Dostupné endpointy Billa API; Zdroj: vlastní tvorba

Z přehledu je také zřejmé, že i API online supermarketu Billa.cz poskytuje dostatečné množství endpointů pro potřeby potravinového agregátoru.

Shrnutí analýzy

Při implementaci agregátoru je zcela nezbytné reagovat na rozdílnost mezi API, a to jak v oblasti zajištění autentizace (a mechanismu CSRF u Billa.cz), tak především práce s odlišnými datovými strukturami. Oba e-shopy mají některé endpointy, které se odchylují od čistých RESTful principů, zejména v oblasti pojmenování zdrojů a používání HTTP metod.

Zároveň je nutné počítat s tím, že vzhledem k tomu, že ani jeden z e-shopů neposkytuje veřejnou dokumentaci ke svému API, tak informace o přesném fungování endpointů, datových strukturách a případných dalších omezeních, je nutné získávat experimentálně.

---

[18] CSRF (Cross-Site Request Forgery) – jedna z metod útoku do internetových aplikací pomocí podvrhování požadavků z nelegitimních zdrojů

4.3 Analýza současných řešení nákupního procesu různých e-shopů

Tato kapitola se zabývá analýzou současných řešení nákupního procesu dvou českých online supermarketů Rohlik.cz a Billa.cz. Cílem je porozumět, jak u těchto e-shopů v současné době probíhá nákupní proces z pohledu zákazníka. Pro popis nákupních procesů byla zvolena metodika BPMN, která je již představena v teoretické části a která umožňuje přehledně vizualizovat a popsat jednotlivé kroky nákupu. Díky této procesní analýze lze v následující kapitole snáze navrhnout nový nákupní proces, který bude respektovat případné odlišnosti a bude také možné identifikovat jeho případné nedostatky a navrhnout optimalizaci.

Diagram nákupního procesu na e-shopu ***Rohlik.cz*** zachycuje jednotlivé kroky, kterými zákazník prochází od vytvoření objednávky až po její dokončení. Diagram přehledně ukazuje možné scénáře nákupního procesu a podmínky, za kterých se proces ubírá různými větvemi. Specifické pro online supermarket Rohlik.cz je to, že nabízí uživatelům prémiové členství, díky kterému mohou mít dopravu zdarma, ale hlavně nemusí 4x do měsíce řešit minimální výši objednávky (v současné době 749 Kč). Z toho důvodu se e-shop snaží zákazníka opakovaně přihlásit a identifikovat, zda disponuje prémiovým členstvím. Podrobnosti jednotlivých kroků jsou patrné přímo z diagramu (viz Diagram 2).

Pro analýzu nákupního procesu na e-shopu ***Billa.cz*** byl rovněž použit BPMN diagram (viz Diagram 3). Z procesní analýzy vyplývá, že tento nákupní proces probíhá vcelku standardně a jediným v podstatě klíčovým okamžikem je splnění podmínky na minimální výši objednávky (v současné době 700 Kč).

V obou případech z pohledu návrhu potravinového agregátoru je problematickou částí podproces dokončení objednávky, protože obsahuje výběr času doručení, který může být v obecném případě komplikovaný, a zároveň pro nižší hodnoty nákupu může mít i značný vliv na výslednou cenu.

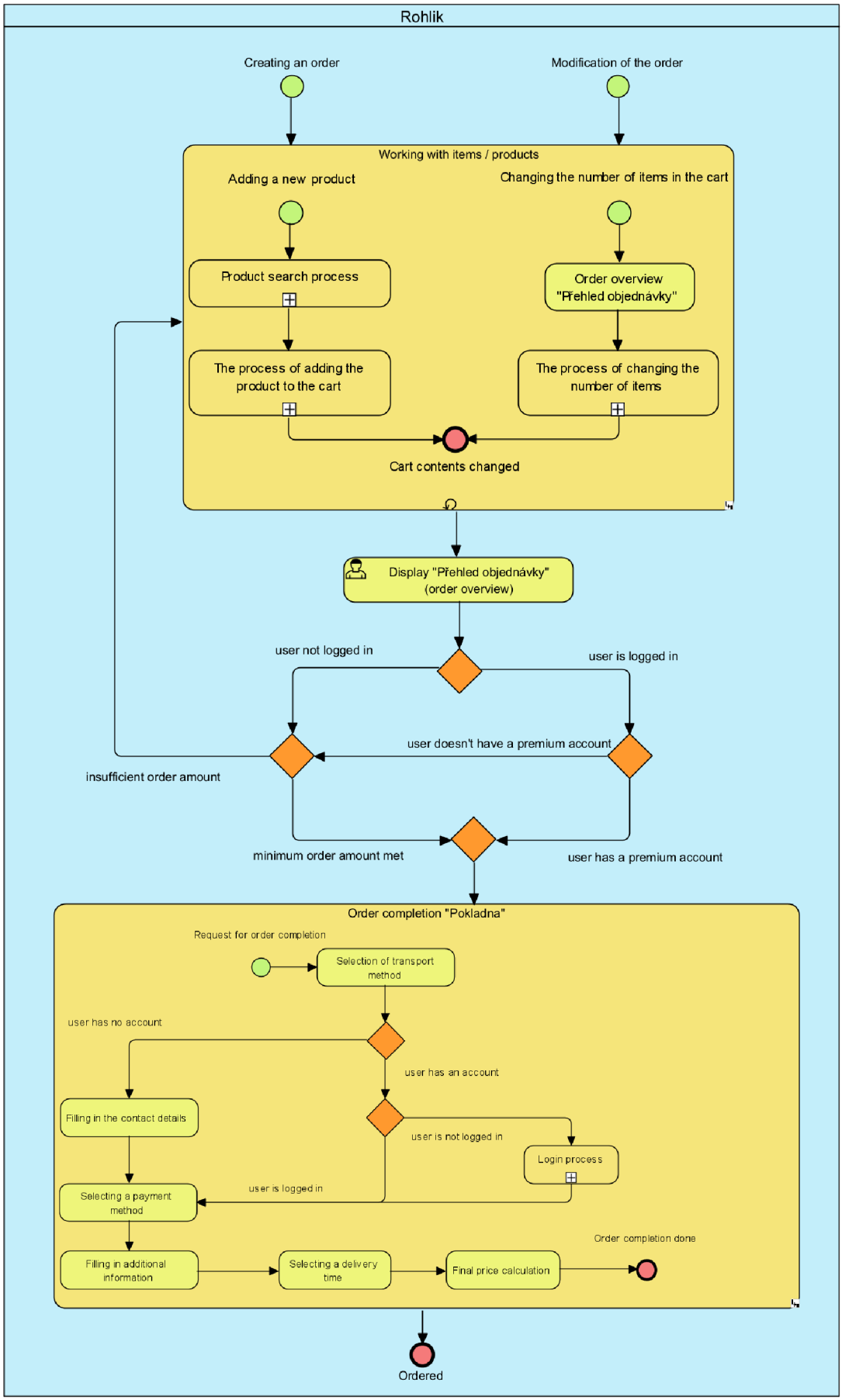


Diagram 2: Nákupní proces Rohlik.cz; Zdroj: vlastní tvorba

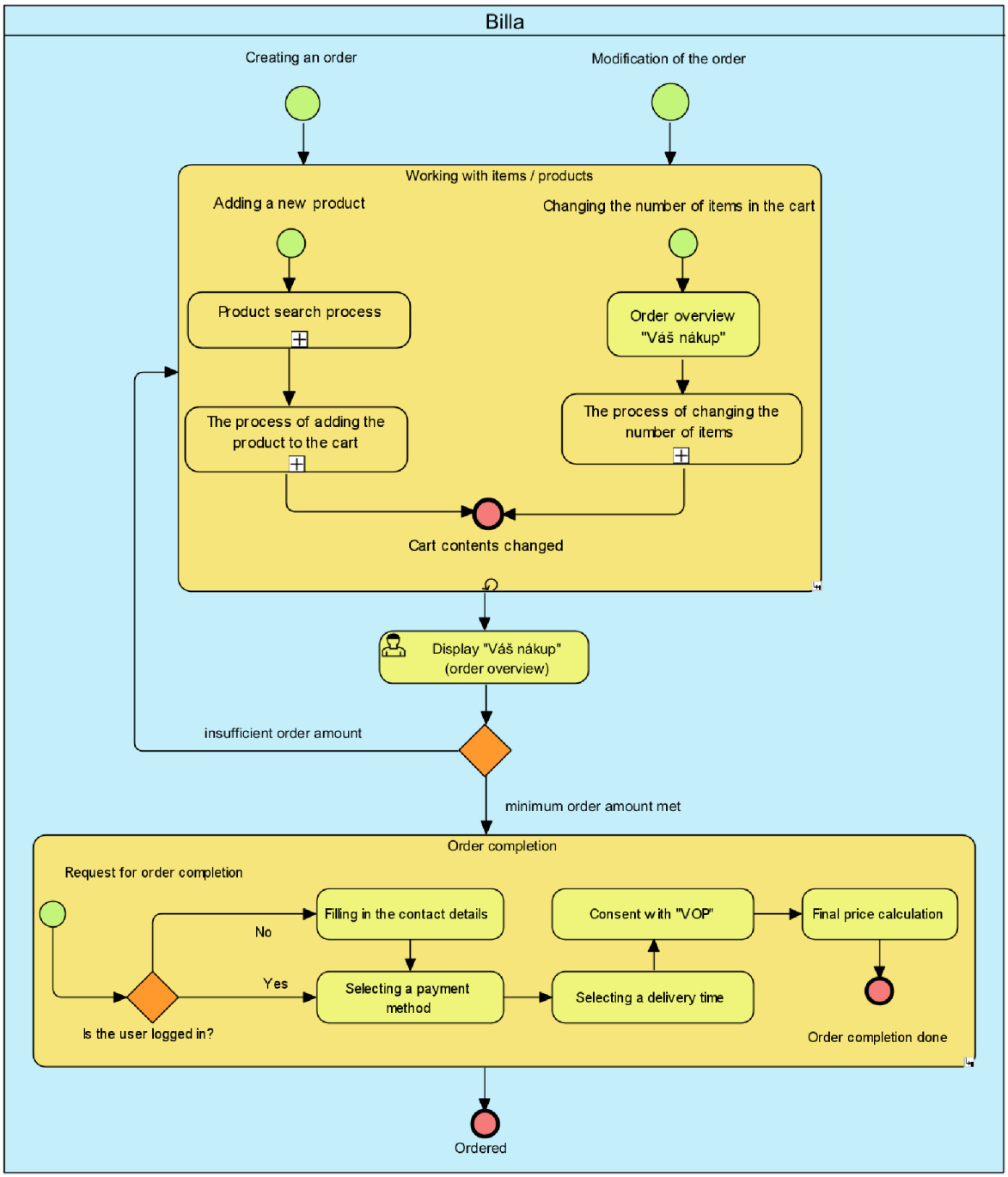


Diagram 3: Nákupní proces Billa.cz; Zdroj: vlastní tvorba

4.4 Návrh nového společného nákupního procesu

Na základě analýzy současných řešení nákupního procesu na e-shopech Rohlik.cz a Billa.cz byl navržen nový proces pro potravinový agregátor (viz Diagram 4 a Diagram 5). Cílem bylo najít způsob, jak nabídnout uživateli možnost agregovaného vyhledávání, hromadného ovládání nákupních košíků a díky hromadným operacím na jedné webové stránce tak značně zjednodušit nákupní proces.

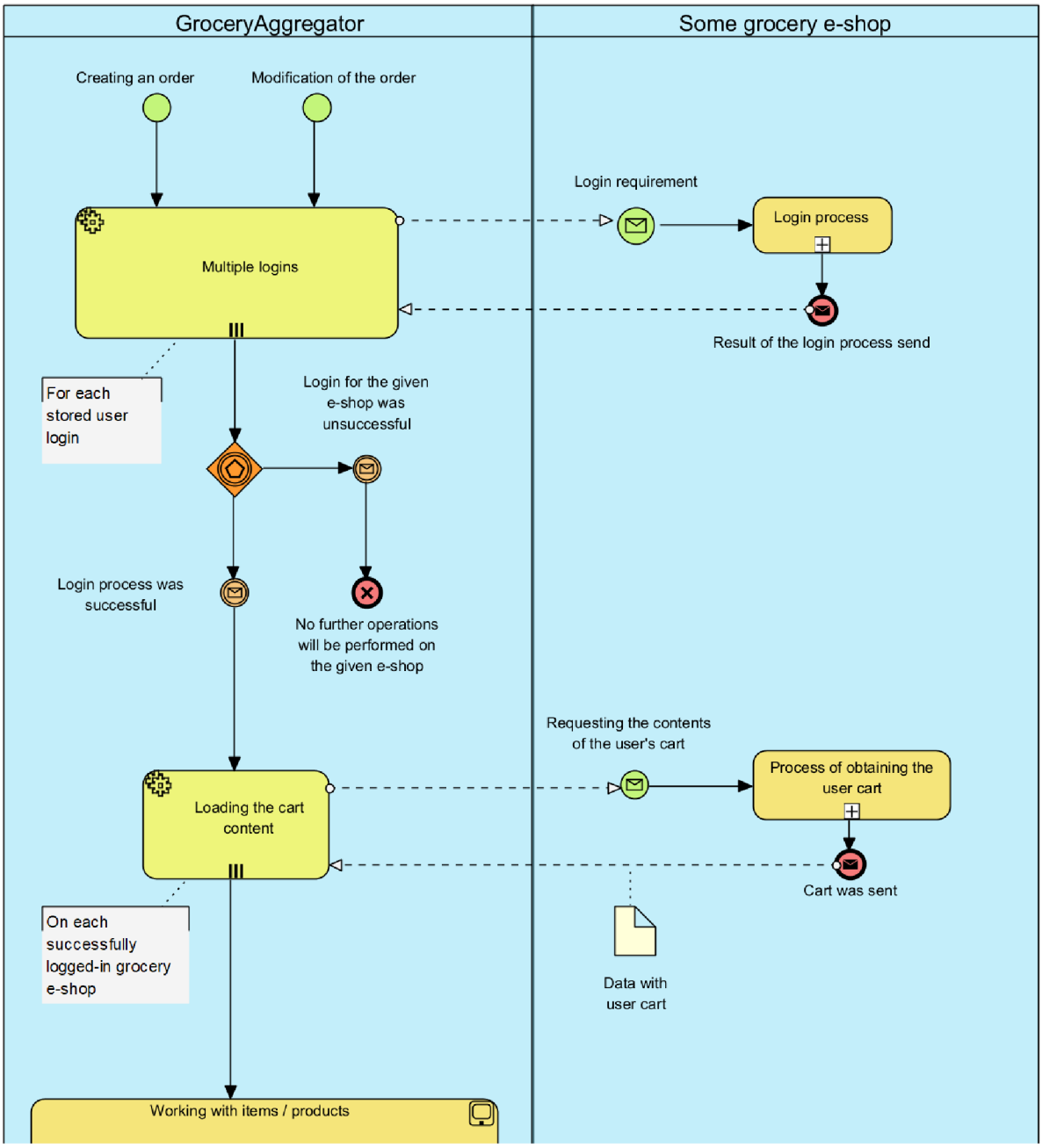


Diagram 4: Nákupní proces potravinového agregátoru (1. část): Zdroj: vlastní tvorba

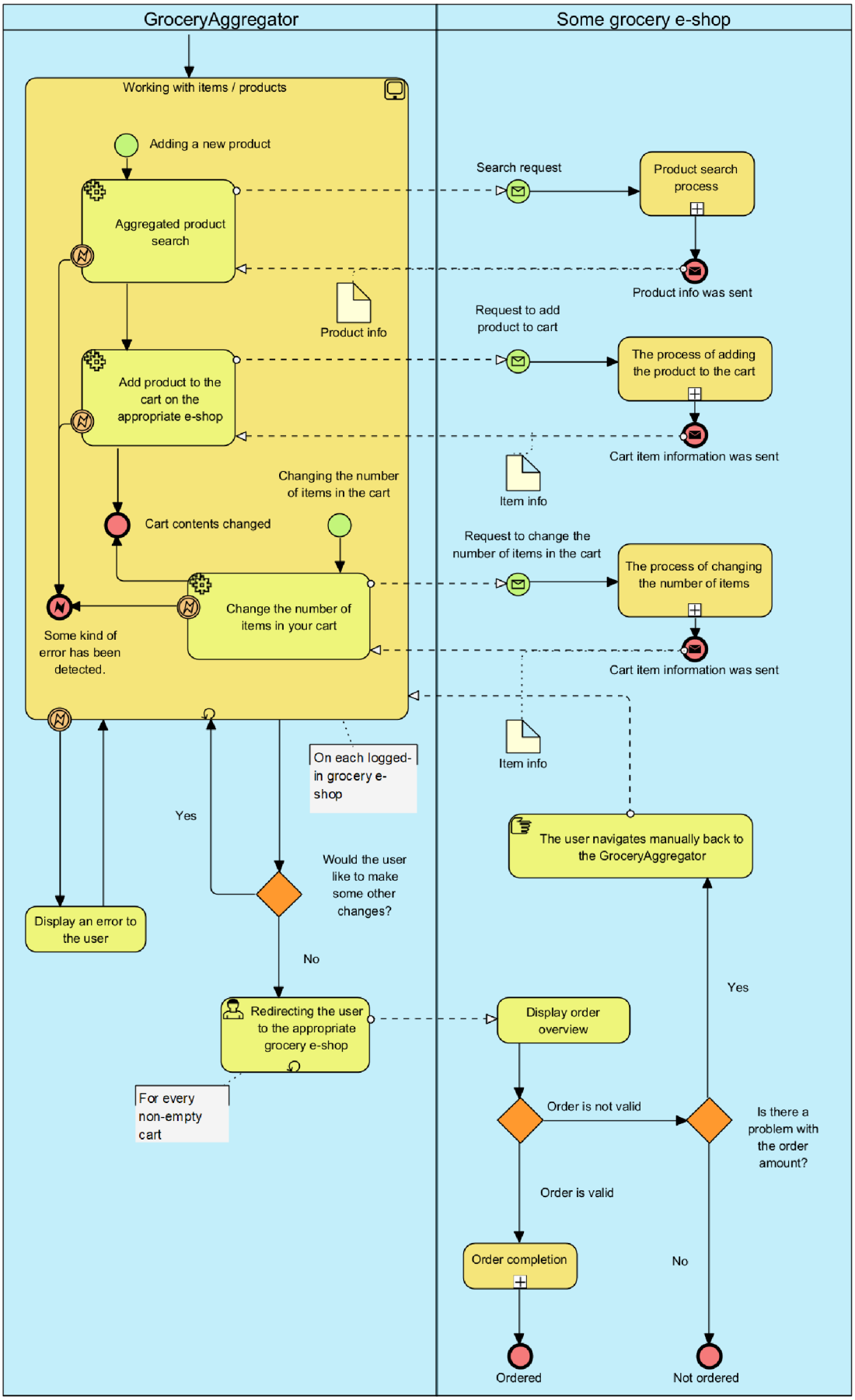


Diagram 5: Nákupní proces potravinového agregátoru (2. část): Zdroj: vlastní tvorba

Důraz byl kladen na uživatelskou přívětivost a zefektivnění nejčastěji opakující se části procesu, tj. podproces s názvem „*Working with items / products*“. I přesto, že tento podproces tvoří v diagramu vizuálně jen menší část, tak je pro uživatele ale nejpodstatnější. Jedná se totiž o často opakovaný podproces, ve kterém většina uživatelů tráví nejvíce času. Nově zákazník bude moci snadno porovnávat ceny a další parametry produktů napříč více obchody.

Vzhledem ke zjištění problematické části procesů jednotlivých e-shopů, na které již byl kladen důraz v předchozí kapitole, bylo rozhodnuto, že danou problematickou část bude vhodnější ponechat dále na uživateli. Z toho důvodu uživatel poté, co využije funkčnost agregátoru (a samozřejmě bude-li spokojen s obsahem nákupních košíků), dokončí objednávku přímo u příslušného e-shopu (podproces „*Order completion*“).

4.5 Návrh softwarové architektury agregátoru

Agregátor byl navržen jako vícevrstvá architektura (viz Diagram 6), která důsledně odděluje prezentační vrstvu (frontend) od aplikační vrstvy (backend). Toto rozdělení umožňuje nezávislý vývoj, škálování jednotlivých částí systému a celkově větší flexibilitu.

Prezentační vrstvu tvoří webová aplikace (ve webovém frameworku Angular), která běží v prohlížeči uživatele a komunikuje s backendem pomocí REST API. Backend je implementován jako aplikace fungující na aplikačním serveru Apache Tomcat (s využitím Java frameworku Spring Boot). Pro persistenci dat je použita objektově–relační databáze PostgreSQL, která „žije“ v Docker[19] kontejneru. Autentizace uživatelů byla řešena pomocí samostatného robustního řešení Keycloak, také hostovaného v Docker kontejneru. Oba tyto kontejnery je možné snadno vytvořit a nastavit s využitím nástroje „*docker compose*“.

Agregátor také potřebuje získávat data z potravinových e-shopů, se kterými spolupracuje. Tato komunikace probíhá pomocí API, které poskytují jednotlivé e-shopy. Agregátor tedy vystupuje vůči e-shopům jako klient a musí se přizpůsobit jejich rozhraním.

[19] Docker – je softwarová platforma, která slouží k vytváření, nasazování a správě kontejnerů

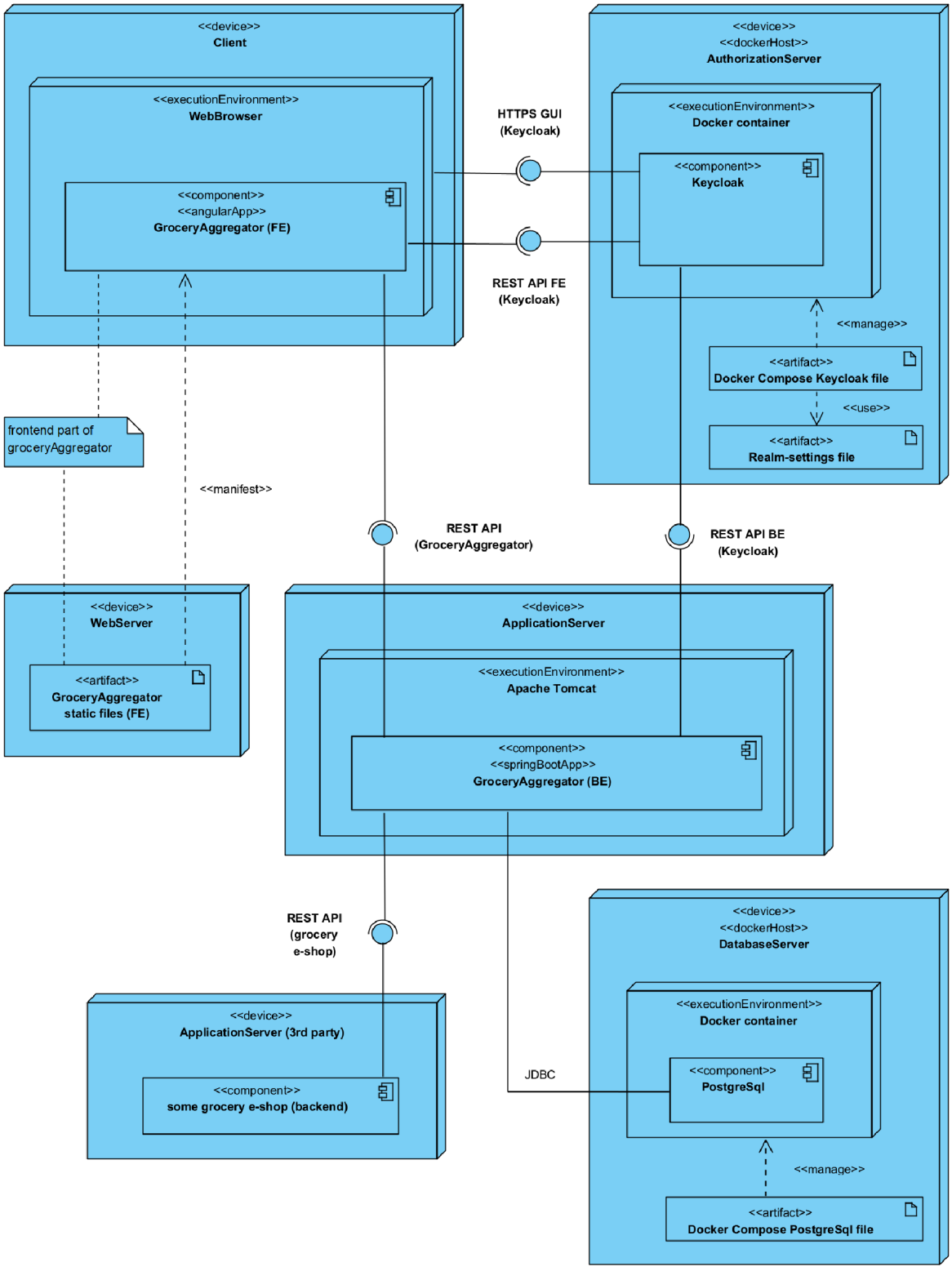


Diagram 6: Architektura potravinového agregátoru; Zdroj: vlastní tvorba

4.6 Návrh databáze

Databáze pro aplikaci byla navržena s využitím přístupu „JPA First“ v kombinaci s frameworkem Spring Boot a ve výchozím nastavení s pomocí ORM frameworku Hibernate. Tento přístup obecně umožňuje definovat doménový model pomocí anotovaných Java tříd (tzv. entit) a z nich následně generovat schéma databáze. Jako konkrétní databázový systém byla zvolena *PostgreSQL*.

Při návrhu databáze byl tedy nejprve definován doménový model pomocí JPA entit, které představují klíčové doménové objekty (např. Company, User, GroceryCredential apod.) a s využitím anotací byly také definovány jejich atributy, ale především vztahy mezi nimi (@OneToMany, @ManyToOne a @JoinColumn). Pomocí anotace @UniqueConstraint jsou nastavená příslušná omezení.

```
@Getter
@Setter
@ToString
@RequiredArgsConstructor
@Entity
@Table(name = "GroceryCredential", uniqueConstraints = {
        @UniqueConstraint(name = "uc_grocerycredential_companyid_userid",
                          columnNames = {"company_id", "user_id"})
})
public class GroceryCredential {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String username;

    private String password;

    @ManyToOne
    @JoinColumn(name = "company_id", nullable = false,
                foreignKey=@ForeignKey(name="fk_grocerycredential_companyid"))
    private Company company;

    @Override
    public final boolean equals(Object o) {
        // hidden - not relevant for this demo
    }

    @Override
    public final int hashCode() {
        // hidden - not relevant for this demo
    }
}
```

Zdrojový kód 1: Ukázka entity GroceryCredential; Zdroj: vlastní tvorba

Na příkladu entity *GroceryCredential* (viz Zdrojový kód 1) je vidět použití zmíněných anotací, a to včetně @Getter, @Setter (a dalších), které umožňují s použitím knihovny *Lombok* psát stručnější a přehlednější kód.

Z takto anotovaných entit byl následně pomocí Hibernate vygenerován fyzický datový model pro PostgreSQL. Schéma databáze tedy nebylo navrhováno ručně, ale bylo automaticky odvozeno z definic zmíněných JPA entit s ohledem na specifika PostgreSQL. Výsledný fyzický model obsahuje tabulky pro každou entitu (company, user, grocery_credential apod.) a cizí klíče, které představují vzájemné vztahy.

Použití přístupu „JPA First" s frameworkem Spring Boot a Hibernate značně zjednodušilo a zefektivnilo proces návrhu a implementace datové vrstvy. Odpadla tak nutnost ručního psaní SQL skriptů nejen při vytváření tabulek, ale také v pozdější fázi při jakýchkoliv úpravách. Případné změny v modelu (typicky přidání atributu nebo případně změna vztahu) se totiž automaticky promítnou do schématu databáze. Díky tomu se tak při vývoji bylo možné soustředit jen na definici a úpravu JPA entit a provázání datové vrstvy s aplikační logikou.

První diagram (viz Diagram 7) zobrazuje datový model v podobě diagramu tříd. Každá entita je reprezentována třídou s atributy a vztahy mezi entitami jsou znázorněny pomocí asociací. Tento diagram tříd poskytuje pohled na databázi z hlediska objektového návrhu, a proto také zobrazuje směr asociací. Fyzický model databáze je zobrazen na diagramu (viz Diagram 8) formou ER[20] diagramu, který se zaměřuje na specifickou strukturu pro konkrétní databázi — v tomto případě PostgreSQL.

[20] ER – entity-relationship

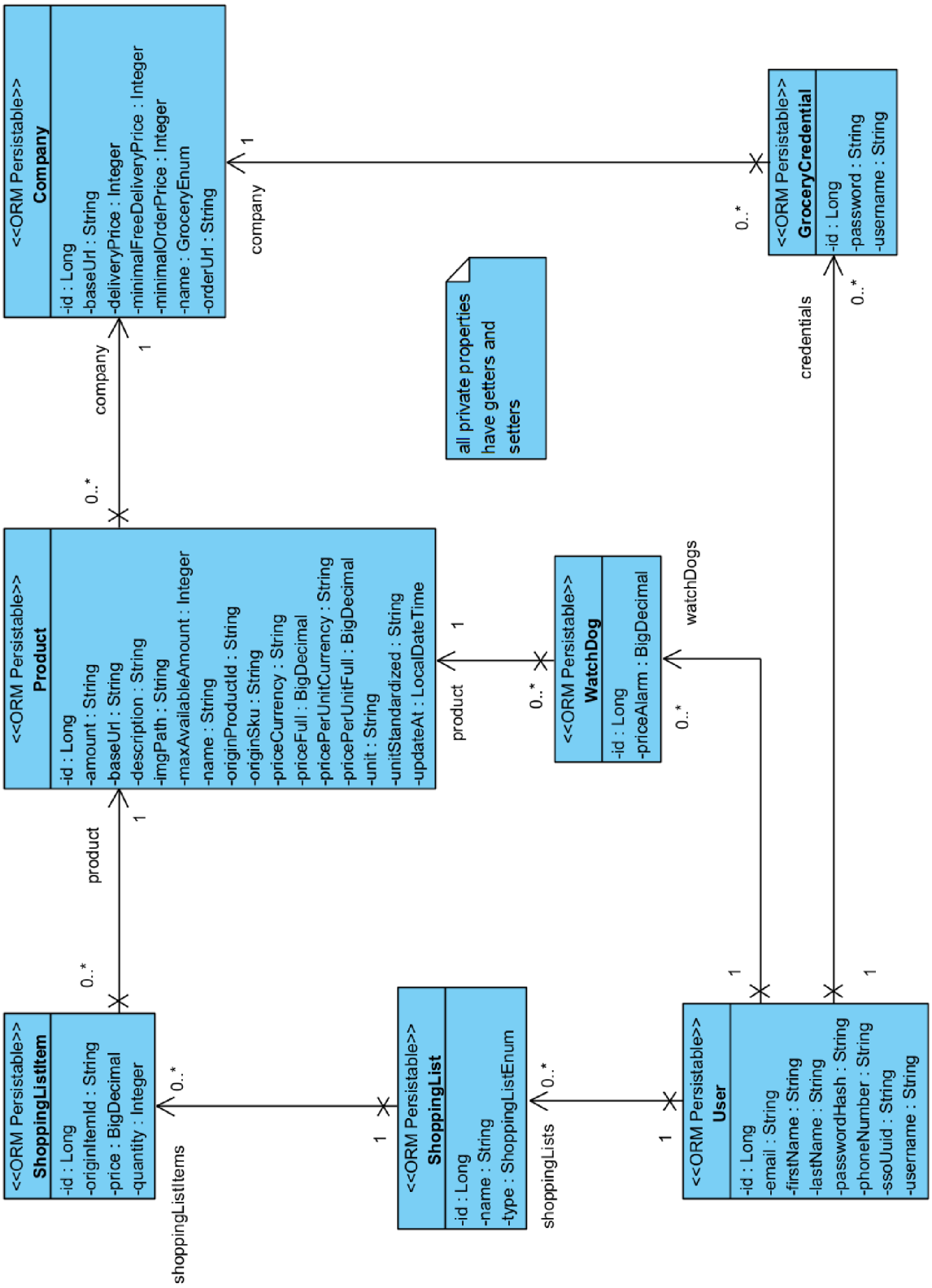


Diagram 7: Class diagram; Zdroj: vlastní tvorba

Diagram 8: ER diagram – fyzický model; Zdroj: vlastní tvorba

4.7 Vytvoření testovacích dat

Vzhledem k neexistenci veřejné dokumentace API třetích stran bylo pro účely vývoje a testování této aplikace velmi vhodné vytvořit testovací data. Díky tomu tak bylo možné simulovat API připojených e-shopů k agregátoru bez jejich zbytečného zatěžování a ihned si tak ověřovat funkčnost implementovaného řešení. K tomuto účelu byl s úspěchem využit nástroj Mockoon, který umožňuje vytváření tzv. mock serverů a mockování API s využitím zmíněných testovacích dat.

Mockování také umožnilo simulovat různé scénáře a odezvy API, což vedlo k rychlejšímu vývoji a důkladnějšímu otestování funkcionality agregátoru bez nutnosti zasahovat do systémů třetích stran. Zároveň, jak již bylo zmíněno, bylo i velmi důležité minimalizovat zatížení externích API během vývoje, ale také tím odpadla nutnost spoléhat se na jejich dostupnost a správnou funkčnost.

Kromě mockování API třetích stran byl s pomocí zmíněného nástroje mockován také backend navrženého agregátoru. Tento přístup umožnil testování frontendové části aplikace, a to nezávisle na vývoji backendu. Díky tomu byla umožněna lepší kontrola nad daty, které poskytoval mockovaný backend. Výhodné to ovšem bylo i z výkonnostních důvodů, protože backend nemusel zpracovávat žádná data. Mockování backendu tak opět napomohlo k rychlejšímu vývoji prezentační části aplikace.

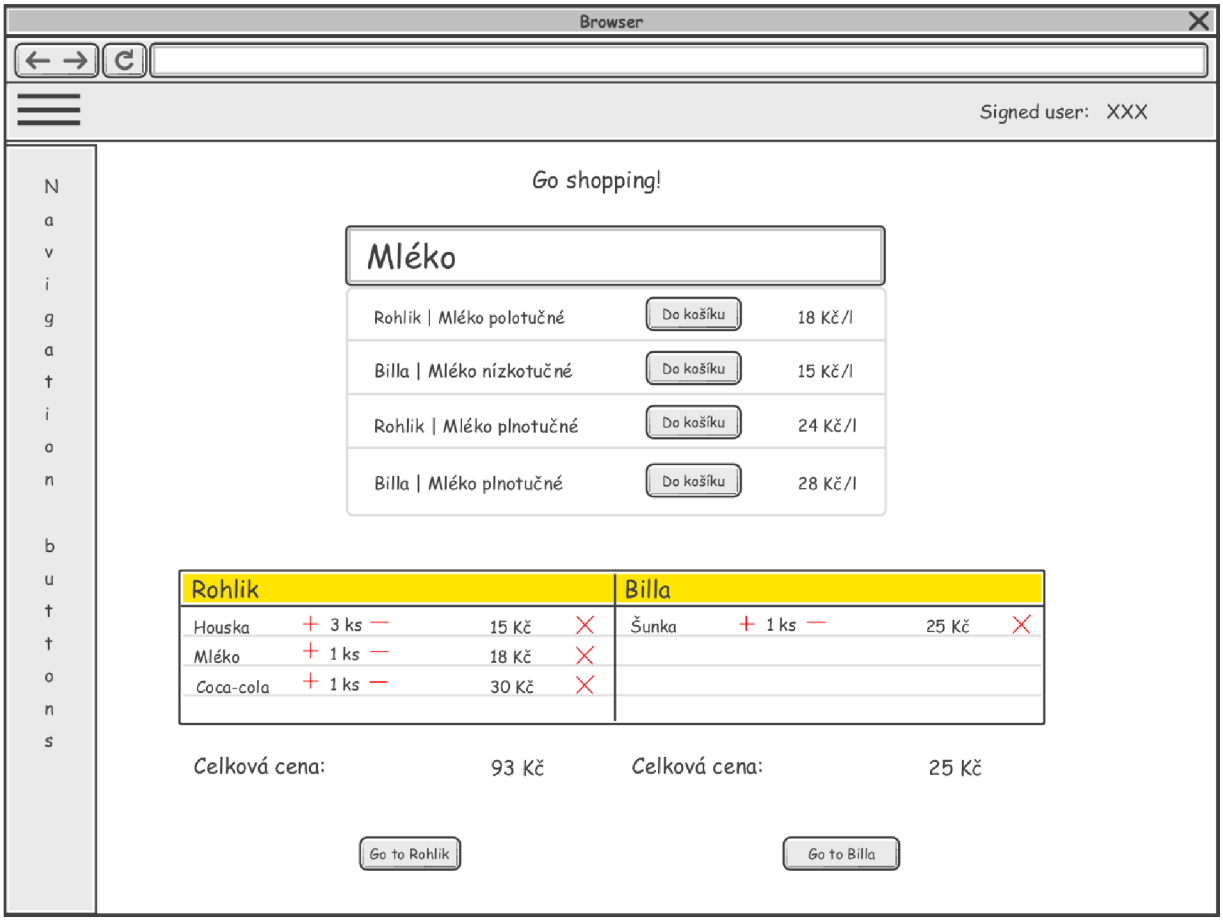
Tento přístup vedl k rychlému odhalení a opravě chyb v rané fázi vývoje, poskytl větší kontrolu nad vývojovým prostředím, snížil rizika ze závislosti na externích systémech a celkově tak výrazně přispěl k vyšší kvalitě vývoje aplikace. Použití nástroje Mockoon je hodnoceno jako velice intuitivní, jednoduché, a přitom nabízí i širokou škálu funkcí. Jeho použití je možné doporučit i pro tvorbu dalších podobných aplikací.

4.8 Návrh uživatelského rozhraní (UI)

Tato kapitola se zabývá návrhem uživatelského rozhraní pro agregátor e-shopů s potravinami. Cílem bylo vytvořit intuitivní a uživatelsky přívětivé rozhraní, které usnadní zákazníkům vyhledávání a nákup potravin z různých e-shopů na jednom místě. Z toho

důvodu jsou tyto hlavní funkcionality rozvrženy na jednu hlavní (klíčovou) obrazovku, ke které, jako jediné, byl navržen také příslušný wireframe[21]. Zároveň bylo při návrhu a pozdějším vývoji bráno v potaz, aby byla aplikace použitelná i na různých zařízeních s využitím responzivního designu.

Pro implementaci uživatelského rozhraní ve frameworku Angular byla zvolena komponentová knihovna *Angular Material UI*[22]. Tato knihovna poskytuje vývojářům sadu předpřipravených komponent a stylů, které usnadňují vytvoření moderního a konzistentního designu.



Obrázek 12: Wireframe hlavní stránky; Zdroj: vlastní tvorba

[21] Wireframe – slouží při návrhu jako základ pro implementaci a poskytuje poměrně jasnou představu o rozmístění klíčových prvků a funkčnosti stránky

[22] Angular Material UI – knihovna komponent uživatelského rozhraní, která implementuje Material Design v systému Angular

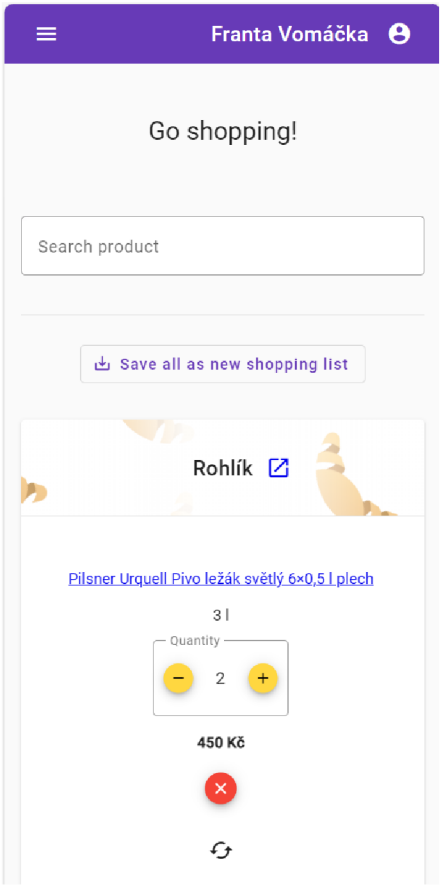
4.8.1 Wireframe hlavní nákupní stránky

Hlavní nákupní stránka (viz Obrázek 12) s názvem „Go shopping“ obsahuje centrálně umístěné vyhledávací pole pro agregované vyhledávání produktů. Po zadání hledaného výrazu, například „mléko“, se zobrazí našeptávač s agregovanými relevantními výsledky. Pod vyhledáváním se nacházejí dva oddělené košíky z e-shopů Rohlik.cz a Billa.cz. Každý košík zobrazuje seznam vložených produktů s jejich počtem a jejich celkovou hodnotou.

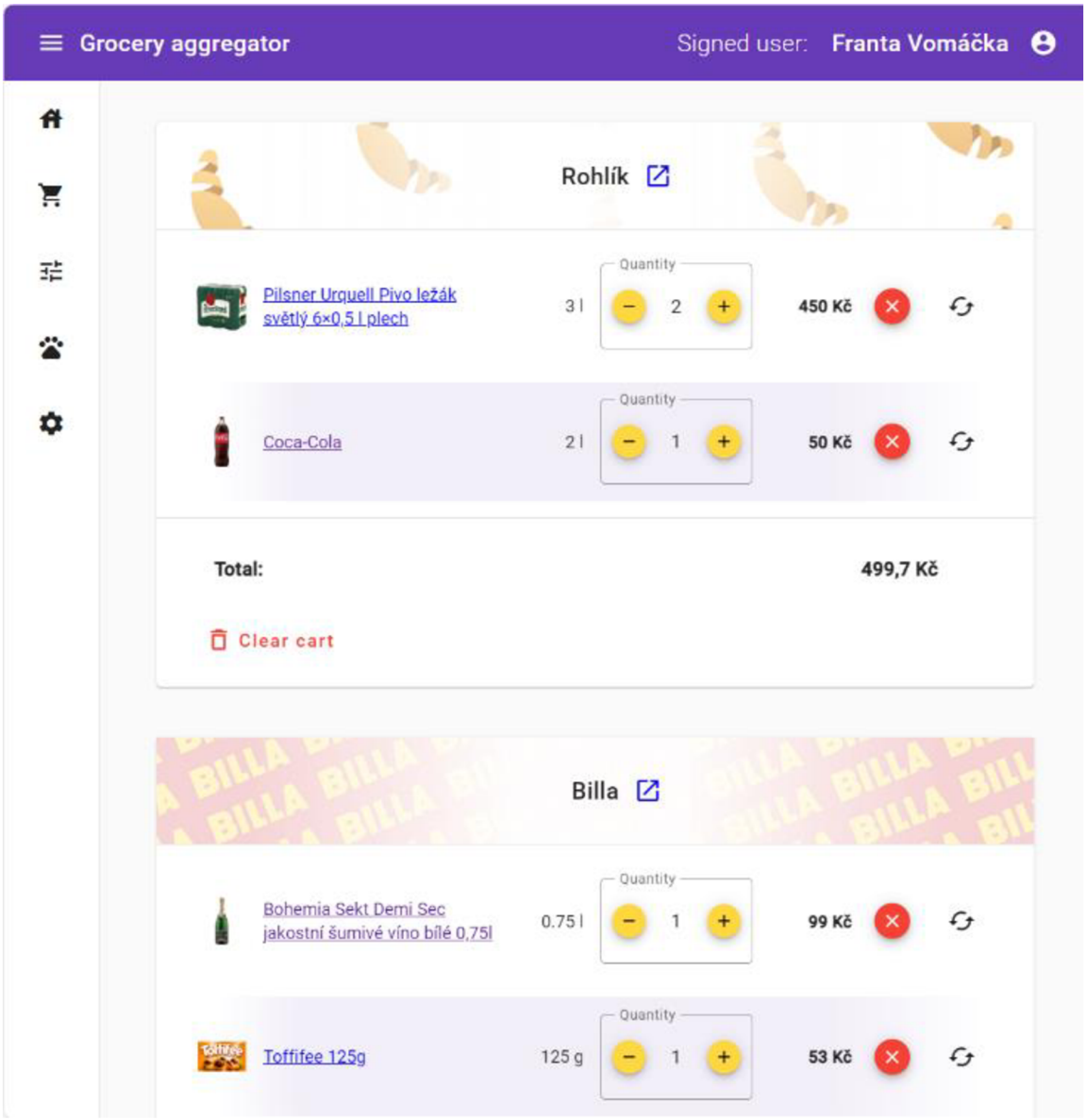
4.8.2 Hlavní nákupní stránka – desktop

Snímek obrazovky (viz Obrázek 15) zachycuje finální implementovanou hlavní nákupní stránku podle návrhu wireframu v desktopovém zobrazení. Na obrazovce lze vidět, že uživatel je aktuálně přihlášený a má zboží v obou nákupních košících.

4.8.3 Responzivní zobrazení



Obrázek 13: Snímek obrazovky hlavní stránky (chytrý telefon); Zdroj: vlastní tvorba



Obrázek 14: Snímek obrazovky hlavní stránky (tablet); Zdroj: vlastní tvorba

Ze snímků obrazovek (viz Obrázek 13 a Obrázek 14) je zřejmé, že hlavní nákupní stránka disponuje responzivním designem, a díky tomu se rozložení prvků na stránce automaticky přizpůsobuje velikosti obrazovky konkrétního uživatele, případně se některé prvky nezobrazují vůbec.

Obrázek 15: Snímek obrazovky hlavní stránky (desktop); Zdroj: vlastní tvorba

4.9 Implementace Backendu

Backendová část je naprogramována v jazyce Java a s využitím známého frameworku Spring Boot, který byl zvolen pro jeho předpřipravenou konfiguraci a širokou dostupnost knihoven a jejich snadnou integraci. Backend komunikuje s ostatními systémy na principech REST. Pro persistenci dat byla zvolena relační databáze PostgreSQL, pro autentizaci uživatelů bylo vybráno robustní open-source řešení Keycloak. Jak relační databáze, tak Keycloak jsou nasazovány jako kontejnerové řešení s využitím Docker platformy.

Díky využití Spring Security, vlastním konvertorům pro převod JWT tokenů na autentizační objekty, bylo možné v příslušných kontrolerech jednoduše nastavit autorizační pravidla (viz Zdrojový kód 2), a navíc díky vlastním anotacím (viz Zdrojový kód 3) bylo možné jednoduše získat instanci uživatele.

```
@Slf4j
@Transactional
@RestController
@PreAuthorize("hasRole('user') and hasAuthority('SCOPE_read')")
@RequestMapping("/grocery")
public class GroceryController {
```

Zdrojový kód 2: Ukázka zabezpečení kontroleru; Zdroj: vlastní tvorba

```
@Target({ElementType.PARAMETER, ElementType.TYPE})
@Retention(RetentionPolicy.RUNTIME)
@Documented
@AuthenticationPrincipal(expression =
        "@userServiceImpl.getUserFromJwt(#this)")
public @interface CurrentUser {
}
```

Zdrojový kód 3: Ukázka vlastní anotace; Zdroj: vlastní tvorba

Aplikaci je možné snadno konfigurovat a spouštět pro různá prostředí, mj. díky přítomnosti konfiguračních souborů „`application.yaml`“ a „`application-{profile}.yaml`“.

Pro vývojové prostředí jsou k dispozici profily „`dev-live`“ (pro práci s „živými“ API potravinových e-shopů) a „`dev-mock`“ (pro použití mockovaných dat – viz 4.7). Pro nasazení na produkčním prostředí by však nejprve bylo nutné (z logických důvodů) upravit příslušný konfigurační soubor.

Kód backendu je strukturován do balíčků podle své funkcionality a zodpovědnosti, což přispívá ke zvýšení přehlednosti a snazší udržovatelnosti. Aplikace byla také navržena s ohledem na budoucí rozšiřitelnost a snazší integraci dalších potravinových e-shopů.

4.10 Implementace Frontendu

Frontendová část aplikace byla implementována s využitím jednoho z nejpoužívanějších frontendových frameworků, a to Angularu (ve verzi 17). Jak již bylo zmíněno v kapitole 3.6.1, tak Angular je open-source framework založený na jazyku Typescript. Je vyvíjen společností Google a je určen k tvorbě tzv. single-page aplikací. Díky své komplexitě umožňuje vývojářům tvorbu i velmi složitých webových aplikací.

Pro uživatelské rozhraní byla využita knihovna komponent *Angular Material*, která obsahuje již sadu předpřipravených prvků ve stylu *Material Designu* od společnosti Google. Díky tomu má aplikace vzhled odpovídající dnešní trendům v oblasti UI. Při stylování jednotlivých komponent byl také využit preprocesor Sass[23], který rozšiřuje možnosti dnes běžně využívaného CSS[24].

Za zmínku stojí komponenta zajišťující produktové vyhledávání („`ProductSearchComponent`“), která je použita na mnoha stránkách („Go shopping!“, „WatchDog settings“, „ShoppingLists settings“), a komponenta zobrazující obsah nákupního košíku („`CartsComponent`“), která se skládá z mnoha dalších vzájemně nezávislých komponent.

---

[23] Sass — Syntactically Awesome Style Sheets
[24] CSS — Cascading Style Sheets

Velmi praktická je knihovna „`angular-auth-oidc-client`“, která značně odstiňuje vývojáře od autentizace uživatelů pomocí protokolu *OpenID Connect* oproti Keycloak serveru. Díky tomu je možné mnohem jednodušeji zajistit uživatelům jednotné přihlášení (více viz 3.5). Další zajímavostí jsou některé tzv. HTTP interceptory. Například třída „`LoadingInterceptor`“ zajišťuje zobrazení načítacího indikátoru (tzv. spinner) při každém HTTP požadavku a při vrácení odpovědi ze serveru opět zajistí jeho skrytí.

Podobně jako v případě backendové části aplikace je možné pro spuštění zvolit vhodnou konfiguraci, tak i v tomto případě byly připraveny konfigurační soubory pro různá prostředí — „`dev-live`“ pro nasměrování na funkční backendový server a „`dev-mock`“ pro spuštění nad mockovanými daty.

5 Zhodnocení výsledků

Výsledkem této diplomové práce je funkční webová aplikace (agregátor potravinových e-shopů), která umožňuje uživatelům především rychlé porovnání online cen potravin a nakupování na dvou vybraných e-shopech (Rohlik.cz a Billa.cz) z jednoho místa.

Pro tvorbu této webové aplikace byly využity moderní komplexní frameworky a další technologie. Frontend byl implementován pomocí frameworku *Angular*, backend pomocí frameworku *Spring Boot*, k ukládání dat byla použita databáze *PostgreSQL* a pro autentizaci uživatelů robustní řešení *Keycloak*. Jak *PostgreSQL*, tak Keycloak se nasazují samostatně, a každý je tak provozován ve svém vlastním *Docker* kontejneru, což samozřejmě zlepšuje mj. jejich správu a škálovatelnost. Celkově tato architektura zajišťuje dobrou správu, škálovatelnost, udržovatelnost kódu a také rozšiřitelnost aplikace do budoucna.

Uživatelské rozhraní je z autorova pohledu považováno za moderní, intuitivní a přehledné, mj. díky využití *Material Designu* od společnosti Google. Uživateli se po přihlášení zobrazí tzv. dashboard[25] se čtyřmi dlaždicemi (odkazy): „Go shopping!“, „Shopping lists“, „Watch Dogs“ a „Grocery Credentials“.

Hlavní funkcionalita aplikace se skrývá pod odkazem „Go shopping!“, kde uživatel má ihned k dispozici aktuální obsahy dvou jeho nákupních košíků a nad nimi také agregované vyhledávání (pro přidání vybraných produktů). Uživatel může s položkami v košících libovolně pracovat (tzn. měnit jejich počet a odebírat je) a průběžně sledovat také celkovou cenu nákupu v každém z košíků.

Ačkoliv cíle práce byly splněny, tak i přesto zde ještě existuje prostor pro zlepšení. Asi největším přínosem pro uživatele by bylo rozšíření počtu podporovaných online supermarketů, což by uživatelům poskytlo větší možnost výběru produktů, a tím i rozsáhlejší srovnání online cen. Uživatel by v takovém případě mohl ocenit i širší

[25] Dashboard – v doslovném překladu „nástěnka“ nebo také řídící panel

nabídku služeb (např. v oblasti dodávky zboží). Pro přidání dalšího potravinového e-shopu by autor této práce v takovém případě doporučoval postupovat podle stejné metodiky jako v praktické části této práce.

Dalším možným zlepšením by mohlo být dokončení objednávky přímo v agregátoru, aby uživatel nemusel dokončovat objednávku ve vybraném e-shopu (a opouštět tak prostředí aplikace). V takovém případě by však bylo nejprve nutné detailně analyzovat API třetích stran a přesvědčit se, zdali je to vůbec proveditelné. A také zda je to účelné, protože tato část procesu se může často měnit a nemusí být také snadné tento proces navrhnout sjednocený, protože doručovací okna konkrétních e-shopů se mohou vzájemně značně lišit.

6 Závěr

Cílem této diplomové práce bylo navrhnout a vytvořit aplikaci umožňující srovnání vybraných online e-shopů s potravinami (tzv. agregátoru), které působí v České republice. Vedlejším cílem bylo vytvoření této aplikace za účelem usnadnění porovnání cen potravin a na to navazujícího nákupního procesu na různých e-shopech.

V teoretické části práce byla nejprve provedena rešerše e-commerce v oblasti potravin v ČR. Poté byl čtenář seznámen se všemi použitými technologiemi a nástroji jako je grafická notace pro procesní modelování (BPMN), programovací jazyk Java, framework Spring Boot a Angular, databáze PostgreSQL ad. Nebyla vynechána ani teorie zabývající se softwarovou architekturou a aplikačním rozhraním (API).

Výše uvedených cílů bylo možné dosáhnout přesným dodržením stanovené metodiky, jejíž výsledky jsou detailně popsány v praktické části této práce. Nejprve byla provedena analýza požadavků, aby bylo zajištěno, že výsledná aplikace bude splňovat očekávání uživatelů. Dále bylo nutné provést analýzu API vybraných online supermarketů (byl zvolen *Rohlik.cz* a *Billa.cz*) a poté také analýzu jejich vlastních nákupních procesů.

Po této analýze bylo přistoupeno k návrhu aplikace, čímž byl splněn jeden z dílčích cílů práce. Návrh aplikace se skládal podle metodiky z navržení nového společného nákupního procesu, softwarové architektury agregátoru a také z návrhu struktury databázového modelu. Také byla dle metodiky připravena tzv. mockovaná (testovací) data a navrženo uživatelské rozhraní.

Poté byla s využitím získaných poznatků aplikace také implementována, a to v obou oblastech, tj. jak v backendové (serverové) části, tak i frontendové (klientské) části. Dále bylo ještě nad rámec zadání zprovozněno moderní a bezpečné přihlašování uživatelů do aplikace pomocí softwarového produktu Keycloak, který umožňuje jednotné přihlášení pomocí správy identit a přístupu. A i tento nástroj byl čtenáři představen v teoretické části.

Vytvořený agregátor splňuje stanovené cíle a poskytuje uživatelům možnost pohodlného porovnání cen a nákupu potravin z různých e-shopů na jednom místě. Tvorba této aplikace

přinesla autorovi práce další zajímavé profesní zkušenosti s návrhem a implementací komplexní webové aplikace postavené na moderních technologiích.

7 Seznam použitých zdrojů

1. **Cushman & Wakefield.** *ONLINE PRODEJ POTRAVIN SE V ČESKU KAŽDOROČNĚ ZDVOJNÁSOBÍ, ZVYŠUJE POPTÁVKU PO SKLADOVÝCH PROSTORÁCH A NÁJMY V NICH.* [Online] [Citace: 16. 1. 2024.] Dostupné z: https://www.cushmanwakefield.com/cs-cz/czech-republic/news/2021/10/online-food-sales.
2. **VELKÁ PECKA s.r.o.** *Rohlik.cz zkracuje dobu doručení nákupu ve vybraných částech Prahy a Brna na 60 minut a stává se tak světovým unikátem. Zákazníci si mohou vybrat z více než 21 000 produktů, které jim kurýr přiveze už do hodiny.* [Online] [Citace: 23. 1. 2024.] Dostupné z: https://rohlik.mediaboard.site/p/rohlikcz-zkracuje-dobu-doruceni-nakupu-ve-vybranych-castech-prah-34.
3. **Economia, a.s.** *Kompletně digitalizovaný sklad Košík.cz firmě šetří pohonné hmoty i obalový materiál.* [Online] [Citace: 20. 1. 2024.] Dostupné z: https://archiv.hn.cz/c1-66980190-kompletne-digitalizovany-sklad-kosik-cz-firme-setri-pohonne-hmoty-i-obalovy-material.
4. **Internet Info, s.r.o.** *Košík.cz prodělal další více než půl miliardu. Stále je finančně závislý na Křetínském.* [Online] [Citace: 20. 1. 2024.] Dostupné z: https://www.lupa.cz/aktuality/kosik-cz-prodelal-dalsi-vice-nez-pul-miliardu-stale-je-financne-zavisly-na-kretinskem/.
5. **Seznam Zprávy, a.s.** *Další řetězec se vrhl na online prodej. Pilotní prodej ladil půl roku.* [Online] [Citace: 24. 1. 2024.] Dostupné z: https://www.seznamzpravy.cz/clanek/ekonomika-firmy-billa-se-pridala-do-online-zavodu-sni-o-stribrne-medaili-236746.
6. **International Organization for Standardization.** *ISO/IEC 19510:2013.* [Online] [Citace: 9. 2. 2024.] Dostupné z: https://www.iso.org/standard/62652.html.
7. **Object Management Group.** *BUSINESS PROCESS MODEL & NOTATION™ (BPMN™).* [Online] [Citace: 14. 2. 2024.] Dostupné z: https://www.omg.org/bpmn.
8. —. *Business Process Model and Notation (BPMN).* [Online] [Citace: 10. 2. 2024.] Dostupné z: https://www.omg.org/spec/BPMN/2.0/PDF.
9. **Carnegie Mellon University.** *What is your definition of software architecture?* [Online] [Citace: 20. 2. 2024.] Dostupné z: https://insights.sei.cmu.edu/documents/2544/2010_010_001_513810.pdf.
10. **Profinit EU, s.r.o.** *Softwarová architektura. Co to vlastně je?* [Online] [Citace: 21. 2. 2024.] Dostupné z: https://profinit.eu/blog/softwarova-architektura-co-to-vlastne-je/.
11. **Stack Exchange, Inc.** *Stack Overflow Developer Survey 2023.* [Online] [Citace: 21. 2. 2024.] Dostupné z: https://survey.stackoverflow.co/2023/#technology-most-popular-technologies.
12. **WALLS, Craig.** *Spring in Action.* 5. vyd. New York : Manning Publications, 2018. s. 4. ISBN 978-1617294945.
13. —. *Spring Boot in Action.* 1. vyd. New York : Manning Publications, 2016. s. 8. ISBN 978-1617292545.
14. **Fowler, Martin.** *Richardson Maturity Model.* [Online] [Citace: 14. 2. 2024.] Dostupné z: https://martinfowler.com/articles/richardsonMaturityModel.html.
15. **Red Hat, Inc.** *Keycloak.* [Online] [Citace: 4. 3. 2024.] Dostupné z: https://www.keycloak.org.
16. **WILKEN, Jeremy.** *Angular in Action.* 1. vyd. New York : Manning Publications, 2018. s. 1. ISBN 978-1617293313.

17. **Valenta, Michal.** *Co je databáze.* [Online] [Citace: 1. 3. 2024.] Dostupné z: https://users.fit.cvut.cz/valenta/doku/lib/exe/fetch.php/bivs/dbs2_02_architektura_srbd.pdf.
18. **Heureka Group a.s.** *Vánoční nákupy vrátily e-commerce v posledním kvartálu k 2% růstu, celkově skončila -6 % oproti loňsku.* [Online] [Citace: 27. 1. 2024.] Dostupné z: https://heureka.group/cz-cs/o-nas/tiskove-centrum/tiskove-zpravy/vanocni-nakupy-vratily-e-commerce-v-poslednim-kvartalu-k-2-rustu-celkove-skoncila-6-oproti-lonsku/.
19. **Visual Paradigm.** *BPMN Notation Overview.* [Online] [Citace: 3. 2. 2024.] Dostupné z: https://www.visual-paradigm.com/guide/bpmn/bpmn-notation-overview.
20. **VMware LLC.** *Introduction to Spring Framework.* [Online] [Citace: 25. 2. 2024.] Dostupné z: https://docs.spring.io/spring-framework/docs/3.2.x/spring-framework-reference/html/overview.html.
21. **JavaTpoint.** *Spring Boot Tutorial.* [Online] [Citace: 27. 2. 2024.] Dostupné z: https://www.javatpoint.com/spring-boot-tutorial.

8 Seznam obrázků, tabulek, grafů, diagramů, kódů a zkratek

8.1 Seznam obrázků

Obrázek 1: Znázorněný vývoj BPMN; Zdroj: (19) 20
Obrázek 2: Rozdělení aktivit dle BPMN; Zdroj: (8) 21
Obrázek 3: Rozdělení událostí dle BPMN z procesního pohledu; Zdroj: (8) 21
Obrázek 4: Rozdělení událostí dle BPMN z pohledu příčinného; Zdroj: (8) 22
Obrázek 5: Rozdělení bran dle BPMN; Zdroj: (8) 22
Obrázek 6: Rozdělení spojovacích objektů dle BPMN; Zdroj: (8) 23
Obrázek 7: Monolitická architektura; Zdroj: (10) 27
Obrázek 8: Mikroservisní architektura; Zdroj: (10) 27
Obrázek 9: Přehled Spring framework; Zdroj: (20) 30
Obrázek 10: Jednoduché grafické znázornění pojmu Spring Boot; Zdroj: (21) 31
Obrázek 11: Richardson Maturity Model; Zdroj: (14) 34
Obrázek 12: Wireframe hlavní stránky; Zdroj: vlastní tvorba 64
Obrázek 13: Snímek obrazovky hlavní stránky (chytrý telefon); Zdroj: vlastní tvorba 65
Obrázek 14: Snímek obrazovky hlavní stránky (tablet); Zdroj: vlastní tvorba 66
Obrázek 15: Snímek obrazovky hlavní stránky (desktop); Zdroj: vlastní tvorba 67

8.2 Seznam tabulek

Tabulka 1: Mapování funkčních požadavků na případy užití; Zdroj: vlastní tvorba 49
Tabulka 2: Dostupné endpointy Rohlik API; Zdroj: vlastní tvorba 50
Tabulka 3: Dostupné endpointy Billa API; Zdroj: vlastní tvorba 51

8.3 Seznam grafů

Graf 1: Roční tržby společnosti Rohlik Group (tržby v mil. EUR); Zdroj: vlastní tvorba z různých veřejných zdrojů 15
Graf 2: Obrat české e-commerce (v mld. Kč); Zdroj: (18) 15
Graf 3: Podíl e-commerce v jednotlivých kategoriích maloobchodu v ČR; Zdroj: (1) 16

8.4 Seznam diagramů

Diagram 1: Use case diagram; Zdroj: vlastní tvorba 43
Diagram 2: Nákupní proces Rohlik.cz; Zdroj: vlastní tvorba 53
Diagram 3: Nákupní proces Billa.cz; Zdroj: vlastní tvorba 54
Diagram 4: Nákupní proces potravinového agregátoru (1. část): Zdroj: vlastní tvorba 55
Diagram 5: Nákupní proces potravinového agregátoru (2. část): Zdroj: vlastní tvorba 56
Diagram 6: Architektura potravinového agregátoru; Zdroj: vlastní tvorba 58
Diagram 7: Class diagram; Zdroj: vlastní tvorba 61
Diagram 8: ER diagram – fyzický model; Zdroj: vlastní tvorba 62

8.5 Seznam zdrojových kódů

Zdrojový kód 1: Ukázka entity GroceryCredential; Zdroj: vlastní tvorba.....59
Zdrojový kód 2: Ukázka zabezpečení kontroleru; Zdroj: vlastní tvorba 68
Zdrojový kód 3: Ukázka vlastní anotace; Zdroj: vlastní tvorba.....68

8.6 Seznam použitých zkratek

AOP – Aspect Oriented Programming

API – Application Programming Interface

BPMI – Business Process Management Initiative

BPMN – Business Process Model and Notation

CLI – Command Line Interface

CRUD – Create, Read, Update, Delete

CSRF – Cross-Site Request Forgery

CSS – Cascading Style Sheets

DI – Dependency Injection

ER – Entity-Relationship

FR – Functional Requirement

HATEOAS – Hypermedia As The Engine Of Application State

HTTP – Hypertext Transfer Protocol

IAM – Identity and Access Management

IdP – Identity Provider

IoC – Inversion of Control

ISO – International Organization for Standardization

JPA – Java Persistence API

JPQL – Java Persistence Query Language

JSON – JavaScript Object Notation

JVM – Java Virtual Machine

JWT – JSON Web Token

NFR – Non-Functional Requirement

OIDC – OpenID Connect

ORM – Object–relational mapping

POX – Plain Old XML

RAD – Rapid Application Development

REST – REpresentational State Transfer

RMM – Richardson Maturity Model

Sass – Syntactically Awesome Style Sheets

SOAP – Simple Object Access Protocol

SP – Service Provider

SQL – Structured Query Language

SSO – Single Sign-On

UI – User Interface

UML – Unified Modeling Language

URI – Uniform Resource Identifier

URL – Uniform Resource Locator

XML – Extensible Markup Language

Přílohy

Příloha A – Zdrojové kódy aplikace GroceryAggregator (backend a frontend),
mock data (Mockoon), docker-compose config files (Keycloak, PostgreSQL)