ИКОНОМИЧЕСКИ УНИВЕРСИТЕТ – ВАРНА
ФАКУЛТЕТ „ИНФОРМАТИКА“
КАТЕДРА „ИНФОРМАТИКА“

Йордан Иванов Йорданов

Облачна информационна система за управление на поръчките от клиенти в производствено предприятие

ДИСЕРТАЦИЯ

за присъждане на образователна и научна степен „доктор“

по докторска програма „Информатика“

професионално направление 4.6. „Информатика и компютърни науки“

Научен ръководител: проф. д.н. Павел Петров

Варна
2025

С Ъ Д Ъ Р Ж А Н И Е

Списък на използваните съкращения

Съкращение	Пълно наименование
ИТ	Информационните технологии
ПОСУП	Персонализирана облачна система за управление на поръчките
ACID	Atomicity, Consistency, Isolation, Durability
AMQP	Advanced Message Queuing Protocol
API	Application Programming Interface
CQRS	Command Query Responsibility Segregation
CQS	Conveyancing Quality Scheme
CRUD	Create, Read, Update and Delete
DDD	Domain Driven Design
ERP	Enterprise Resource Planning
ES	Event Sourcing
gRPC	Google Remote Procedure Calls
HTTP	Hypertext Transfer Protocol
IoT	Internet of Things
MRP	Material Requirements Planning
PaaS	Platform as a Service
REST	Representational State Transfer
RPC	Remote Procedure Call
SaaS	Software as a Service
SCM	Supply Chain Management
SLA	Service Level Agreement
SLI	Service Level Indicator
SLO	Service Level Objectives
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
TMS	Transport Management System
UML	Unified Modeling Language

Въведение

Използването на облачни технологии в производствените предприятия създават условия за оптимизиране на комуникационните процеси и поддържане на постоянни и стабилни взаимоотношения с бизнес клиенти. При въвеждането в експлоатация на нови системи за управление на ресурси и информационна логистика, обаче, е възможно в производствените предприятия да възникнат проблеми, свързани с обработката на поръчки от бизнес клиенти в рамките на вътрешната си верига на доставки. Въпреки това, използването на съвременни информационни технологии дава възможност на предприятията да оптимизират работните си процеси, да удовлетворяват успешно нарастващите клиентски изисквания, както и да се адаптират към бързо променящите се пазарни условия.

Организационните и технологични проблеми във веригите на доставка са разнообразни и обхващат управлението на множество клиентски поръчки и съответни доставки, оптимизацията на информационните потоци, координацията и комуникацията между отделните участници във веригата, както и непрекъснатото приспособяване към пазарна среда, което изисква въвеждане на гъвкави стратегии за управление и оптимална логистична организация. В условията на четвъртата индустриална революция[1], характеризираща се с все по-широко приложение на дигитални технологии в производството, възникват редица проблеми, свързани с взаимодействието между различни вътрешни и външни информационни системи, които изискват осигуряване на оперативна съвместимост, надеждни механизми за обмен на данни, високо ниво на киберсигурност и прилагане на стандартизирани методологии за управление на информационните потоци.

[1] Четвъртата индустриална революция (Industry 4.0) е концепция, обхващаща интеграцията на технологии като интернет на нещата, изкуствен интелект, машинно обучение, роботика и анализ на големи данни. Обединяването на производството със съвременни информационни и комуникационни технологии може значително да повиши ефективността и автоматизацията в бизнеса.

Например, при интеграцията на вътрешни корпоративни системи за планиране на ресурсите с външни бизнес платформи могат да се появят затруднения, свързани с преноса и формата на данни, както и увеличаване на нивото на риск от уязвимости, свързани със сигурността.

Необходимостта от надеждно съхранение и ефективна обработка на големи обеми от данни поражда допълнителни проблеми, свързани с ограничения в капацитета на информационната инфраструктура, която може да е разположена на територията на няколко държави. Международните компании с производствени предприятия и търговски организации в различни държави, са длъжни да се съобразяват със законите и стандартите на държавата, в която работят. Това изисква, както добро познаване на местните нормативните актове, така и тяхното интегриране в използваните информационни системи. Част от тези проблеми могат да се решат чрез използване на облачни технологии.

Актуалността на изследваната тема се обуславя от тенденцията облачните технологии да се превръщат в стратегически инструмент за бъдещ растеж, модернизация и цифрова трансформация на производствените предприятия. Очаква се тази тенденция да продължи, тъй като технологичните средства, предоставени от облачните платформи, дават възможности за сравнително бързо реализиране на иновативни идеи, целящи подобряване на конкурентоспособността. Настоящото проучване разглежда основните проблеми и решения, свързани с внедряването на облачни информационни системи, както и някои съпътстващи ги технологични аспекти.

Исторически погледнато, логистиката и управлението на веригата на доставки се основават на сложни и понякога ръчно управлявани процеси, които включват множество участници – от диспечери, доставчици и превозвачи до крайни потребители. Създадените през 90-те години на миналия век системи за планиране на ресурсите на предприятието (Enterprise Resource Planning – ERP) и за управление на веригата на доставки (Supply

Chain Management – SCM), понякога функционират отделно от останалите информационни подсистеми на едно предприятие, което затруднява комуникацията и обмена на информация между различните отдели и партньори. Тази изолация възпрепятства надграждането на процесите по въвеждане и обработка на информация и създава затруднения за ефективното управление на бизнес процесите.

В периода след 2000-та година до наши дни, водещи технологични компании като Amazon, Microsoft и Google започват да разработват и предлагат т.нар. „облачни услуги“, използвайки мащабируеми инфраструктури за съхранение и обработка на данни в териториално отдалечени и разпределени центрове. В началото водеща роля има компанията Amazon Web Services (AWS), която през 2006 г., става един от първите доставчици на облачни услуги, които дават достъп до виртуални уеб сървъри. Това дава възможност на компаниите да избегнат разходите за поддръжка на физически компютри и да се съсредоточат върху основните си бизнес процеси.

Широкото приложение на облачни технологии през последното десетилетие доведе до съществени промени в производството и логистиката. Облачни информационни системи, като SAP S/4HANA, Oracle Fusion, Microsoft Dynamics 365 и Blue Yonder, дават възможност за оптимизиране на процесите по управление на поръчките чрез широк набор от инструменти, подпомагат намаляването на оперативните разходи и подобряват обслужването на бизнес клиентите.

Развитието на облачните технологии и по-специално въвеждането на моделите: „инфраструктура като услуга“ (от английски: Infrastructure as a Service – IaaS), „платформа като услуга“ (Platform as a Service – PaaS) и „софтуер като услуга“ (Software as a Service – SaaS) откриват нови възможности за дигитализация на бизнес процеси в логистиката (Петров и др., 2020). Прилагането на тези модели премахва необходимостта от поддържане на локална инфраструктура, характерна за т.нар. „on-premise“

услуги, и предоставя на компаниите възможност да внедряват софтуерни продукти, съобразени с техните конкретни нужди. Гъвкавостта на тези модели дава възможност на производствените предприятия да се адаптират към променящите се пазарни условия, улеснявайки бързото вземане на решения, базирани на данни в реално време, и подобрявайки възможностите за планиране и прогнозиране.

Основната **теза**, която застъпваме е, че процесите, свързани с управлението на клиентски поръчки, поддържани и реализирани чрез различни информационно-технологични софтуерни продукти, могат да бъдат интегрирани в персонализирана облачна система, базирана на облачните модели IaaS, PaaS и SaaS. Конфигурирана според нуждите на конкретно производствено предприятие, тази система би допринесла за усъвършенстване на бизнес операциите. Облачната информационна система може да рационализира процесите по управление на поръчки за продажба, както и цялостната верига на доставки, като предостави адаптивни мобилни и уеб приложения.

За разлика от универсалните системи (например SAP S/4), персонализираната система специално се разработва и конфигурира, за да отговаря на специфичните нужди и процеси на конкретна компания, като същевременно се интегрира със съществуващи или ново въведени подсистеми. Въпреки, че първоначалната инвестиция е сравнително по-голяма, в дългосрочен план персонализирането намалява зависимостта от външни доставчици и осигурява непрекъснат контрол върху развитието и поддръжката на системата.

В настоящото проучване се анализират методите, чрез които производствените предприятия управляват информационните потоци, свързани с клиентски поръчки. Съхранената и обработена информация от тези потоци е от съществено значение за организирането на доставките – от натоварването и транспортирането, до разтоварването на продуктите. Проучването разглежда основни системи и процедури, като някои

второстепенни аспекти остават извън неговия обхват.

Обект на изследване са процесите във веригите на доставки в производствено предприятие, което произвежда, продава и доставя собствени продукти чрез отделни търговски организации в множество държави. В рамките на проучването се разглеждат информационни системи и процедури, свързани с управлението на потока от данни, от момента на производство до достигането до крайния потребител. Заедно с това се изследва практическото прилагане на облачни технологии за управление на поръчки, логистични операции и веригите на доставка, като се спазват специфичните изисквания на бизнес клиентите и оптималното използване на ресурсите.

Предмет на изследване са методите за рационализация и автоматизация на бизнес процеси, осъществявани чрез съвременните възможности на облачни платформи и технологии. Въз основа на методите за рационализация и набор от софтуерни архитектурни подходи се разработва персонализирана информационна система, която дава възможност за динамична адаптация към променящи се във времето изисквания и параметри.

Целта на настоящото изследване е да се проектира и апробира облачна информационна система за управление на поръчките от бизнес клиенти чрез използване на мобилни и уеб приложения, както и да се оцени нейната приложимост в конкретно производствено предприятие.

За постигане на поставената цел са дефинирани следните задачи:

1. Да се изследват основни принципи на управление на веригите за поръчки и доставки, както и да се анализират проблемите, свързани с информационното осигуряване чрез корпоративни ERP и SCM системи.
    1.1. Да се проучат възможностите за дигитализация и рационализация на бизнес процесите, свързани с управлението на поръчки за продажба от бизнес клиенти, чрез персонализирана информационна система, адаптирана към специфични нужди на

производствено предприятие.

1.2. Да се определи същността на облачните услуги и да се изследват принципи и практики за внедряване на сложна бизнес логика в програмния код на информационната система.

2. Да се предложат концептуален, логически и комуникационен модел, които да послужат като основа на архитектурата на облачната информационна система. Моделите трябва да съответстват на критериите за сигурност, мащабируемост и интеграция.
3. Да се проучат основни функционалности, които да внесат необходимите подобрения по отношение на управлението на поръчките от бизнес клиенти.
4. Да се изготви план за внедряване на системата и да се изберат подходящи технологични средства за нейната реализация. Резултатите от изследването да се апробират в производствено предприятие.

Методика на изследването – за постигане на целта и изпълнение на поставените задачи, в дисертационния труд са използвани редица научно-изследователски методи. Сред тях са: метод на моделиране, логически, сравнителен и системен анализ. Фактите и данните са представени чрез графични, схематични и фигурални подходи, а при апробацията на резултатите са приложени техники за виртуализация. Комбинираното използване на тези методи и техники за събиране, анализ и интерпретация на данни, цели постигане на обосновани заключения и валидиране на получените резултати.

Обхвата на изследването се **ограничава** до конкретни проблеми, архитектури и системи, въз основа на тяхната значимост за дейността на производствено предприятие. В този смисъл, извън обхвата на настоящото изследване остават други възможни подходи, стратегии за вериги на доставка, разработка и внедряване на облачни услуги.

Глава 1. Проблеми на информационното осигуряване при управление на поръчките от клиенти

В настоящата глава се изследват теоретичните принципи и проблеми, отнасящи се до информационното осигуряване в процеса на управление на клиентски поръчки. Анализират се същността и принципите на веригата на доставки, както и тяхното интегриране в корпоративните системи, използвани в производствено предприятие. Разглеждат се възможностите за рационализиране на процесите посредством персонализирана система, конфигурирана според конкретни нужди и изисквания на дадено предприятие. Също така се изследват подходи за централизиране на процесите чрез облачни технологии, както и за управление на бизнес логиката чрез ориентиран към домейн дизайн.

1.1. Управление на веригите от поръчки и доставки чрез корпоративни системи за планиране на ресурси

В съвременната пазарна икономика производствените предприятия са изправени пред засилваща се конкуренция и динамично променяща се бизнес среда. За да запазят и развият конкурентните си предимства, те трябва да се адаптират към настъпващите промени. Управлението на веригите от поръчки и доставки, обхващащо снабдяването, производството, дистрибуцията на продукти и удовлетворяването на заявките и нуждите на бизнес клиентите, е от основно значение за организациите, които се стремят към устойчиво конкурентно предимство.

В научната литература съществуват множество различни дефиниции за термина „*верига на доставките*“. Според Моллов (2017) веригата на доставки са „*етапите, които пряко или непряко участват в изпълнението на заявките на клиента. Веригата на доставки включва не само производителя и доставчиците, но и превозвачите, складовете, търговците на дребно и*

самите клиенти“. Други автори (Khan & Yu, 2019) дефинират веригата на доставки като: „*мрежа от съоръжения и възможности за дистрибуция, която изпълнява функциите на доставка на материали, превръщането на тези материали в междинни и готови продукти и разпространението на тези готови продукти на клиентите*“. Според друга дефиниция (Раковска, 2021) веригата на доставки представлява „*съвкупност от процеси и ресурси, необходими за извършване и доставка на продукт на крайния потребител*“ или също „*канал за ефективно движение на материали, продукти, услуги или информация от доставчици към клиенти*“.

В настоящото изследване приемаме определението на Matinheikki et al. (2022), дефиниращо понятието като „*ясно очертана верига от свързани двойки логистични звена „доставчик – получател“ (структурирани подразделения на фирмата и/или логистичните ѝ партньори), по която конкретната стока и/или услуга се доставя на крайния потребител в съответствие с неговата заявка и изисквания*“.

Транспортирането на продуктите по веригата на доставки се осъществява от производителя до потребителя, докато обратната логистика представлява процеса на движение на стоките в обратна посока – от потребителя обратно към производителя. Обратната логистика включва различни дейности: връщане на стоки от страна на потребителите, които не отговарят на техните очаквания, връщане на стоки с цел ремонт, рециклиране и замяна (Gupta, 2016). Според дефиниция, предоставена от *Асоциацията по обратна логистика*[2], обратната логистика обхваща всички дейности, свързани с продукт или услуга след точката на продажба, като крайната цел е да се оптимизира или направи по-ефективна следпродажбената дейност, което спомага за намаляване на разходите.

В табл. 1.1 са сравнени характеристиките на права и обратна верига на

[2] Асоциацията по обратна логистика е глобална търговска асоциация, създадена през 2002 г. Служи като централен орган за насърчаване на добри практики, предоставяне на сертификати и улесняване на възможностите за работа между производители, компании за търговия и доставчици на услуги.

доставки.

Таблица 1.1

Сравнение между права и обратна верига на доставки

Показатели	Права верига на доставки	Обратна верига на доставки
Оптимизация	Базирана на оптимизиране на печалбата и разходите.	Базирана на екологични принципи и политики за качество на продуктите.
Прогнозиране	Сравнително по-лесно прогнозиране на търсенето на продукти.	По-трудно прогнозиране за връщане на продукти.
Качество на продукта	По-малко вариации в качеството на продукта.	Големи различия във върнатите продукти.
Време за обработка	Времето и стъпките за обработка са предварително дефинирани.	Времето и стъпките за обработка зависят от състоянието от върнатия продукт.
Транспорт	Стоките се транспортират от едно място до много други места.	Върнатите продукти се събират от много места и се връщат към едно.
Оценка на разходите	Сравнително лесна, основаваща се счетоводни отчети.	Сравнително сложно определяне и представяне на разходи.
Опаковка	Стандартна структура на продукт.	Модифицирана структура на продукт.
Прозрачност на процесите	Проследяване на движението в реално време.	Липса на възможности за обратна връзка.

Източник: Gupta, 2016.

Може да се обобщи, че при правите вериги на доставки основно внимание се отделя на движението на продукти от производителите към

крайните потребители, с цел максимизиране на печалбата и минимизиране на разходите, докато при обратните вериги на доставки се проследява движението на продукти от потребителите към производителите, с цел връщане или рециклиране. За да се постигне цялостен цикъл на управление на материалите и продуктите, двата вида вериги трябва да работят координирано.

Важен елемент в управлението на веригата на доставки е логистиката (Василев, 2015). Европейската логистична асоциация[3] дефинира логистиката като „*организация, планиране, контрол и реализация на придвижването на стоковия поток от проектирането и закупуването, през производството и разпределението, до крайния потребител с цел удовлетворяване изискванията на пазара с минимални операционни и капиталови разходи*“. Дейностите, свързани с логистиката, се характеризират с различни параметри: начална и крайна точка на движение, изминато разстояние, скорост, продължителност на придвижването, време на престой, вид на използваните транспортни средства и условия на транспортиране (Bisogni et al. 2021). Авторският колектив Василев, Николаев и Милкова (2023) разглежда логистичната система като устойчива мрежа от звена, които са взаимно свързани и управлявани централно чрез административни системи, които подпомагат управлението на целия логистичен процес. Целта е да се удовлетворят заявките и нуждите на клиентите, да се поддържа баланс между предлагането и търсенето.

В някои източници се използва понятието „*логистичен мениджмънт*“, което обхваща планирането, организацията, координацията и контрола на всички операции, необходими за удовлетворяване на изискванията на клиентите и осигуряване на ефективно придвижване на стоките от мястото на натоварване до мястото на разтоварване (Сълова, 2020;

[3] Европейската логистична асоциация е федерация на националните логистични асоциации в Европа, създадена през 1984 г. и базирана в Брюксел. Тя е отговорна за създаването и поддържането на система от стандарти за компетентност в областта на логистиката, които са в съответствие с европейската квалификационна рамка.

Molamohamadi et al., 2021). Логистичният мениджмънт включва вземане на стратегически, тактически и оперативни решения, насочени към развитието на логистичните дейности и ефективното взаимодействие с доставчиците и клиентите по веригата на доставки. Стратегическият логистичен план има за цел да реализира поставена стратегия и да осигури функционирането на логистичната мрежа (Sandberg, 2025). Редица автори и изследователи (Calabrò et al. 2020; Zając & Swieboda, 2023) посочват, че оптимизационните задачи в логистичното планиране могат да бъдат разграничени в различни компоненти на веригата на доставки, с насока към управлението на поръчки от бизнес клиенти. Тези компоненти включват оптимизация на множество етапи в цикъла на изпълнение на поръчките, включително приемането, обработката и доставката (Tukamuhabwa et al. 2021).

Според Chen (2020) за да се изяснят компонентите на веригата на доставки, е необходимо да се разгледат различни стратегии за управление, които предлагат за използване на процедури, свързани с трансформацията на суровини и материали в готови за доставка стоки. Тези процедури са насочени към създаване на стойност както за производителите, така и за потребителите. Съществуват различни стратегии за управление, като някои от основните включват избягване на излишъци, намаляване на отпадъците, постигане на гъвкавост и бърза реакция на промените в търсенето, както и доставка „точно навреме“ за производство или продажба (Alzoubi et al., 2020). Тези стратегии могат да се комбинират една с друга или да бъдат адаптирани според спецификите на определена компания. Всяка от тях има своите предимства и недостатъци, но всички споделят една обща цел: да подобрят качеството и времето за доставка, да увеличат ефективността, да намалят излишните запаси и разходи. Това в крайна сметка носи ползи както за клиентите, така и за бизнеса.

За целите на настоящото изследване, във връзка с дейностите по доставка и продажба, по наше мнение, е подходяща за прилагане стратегията за доставка на продукция „точно навреме“ (от английски – just in time).

Стратегията може да се разглежда и като управленски подход, насочен към доставка на готовите продукти точно в момента, когато са необходими. Редица автори (Hahn, 2019; Althabatah et al., 2023) описват необходимостта от координирано изпълнение на разнообразни функции и операции, разделени в няколко SCM компонента. Някои от тях са: стратегии за планиране на веригата на доставки, оперативно управление на доставките, управление на активи и жизнения цикъл на продуктите, снабдяване, както и корпоративни приложения, които поддържат управлението на информационните потоци. В тази връзка, таблица 1.2 представя описание на някои от компонентите, считани за основни в стратегията за управление, според Runkler et al. (2019). Съществуват и други компоненти, които също оказват влияние върху SCM, но те не са разгледани в настоящия труд, тъй като имат косвено значение към разглежданите проблеми.

Таблица 1.2

Основни компоненти на стратегията за доставка на продукция „точно навреме“

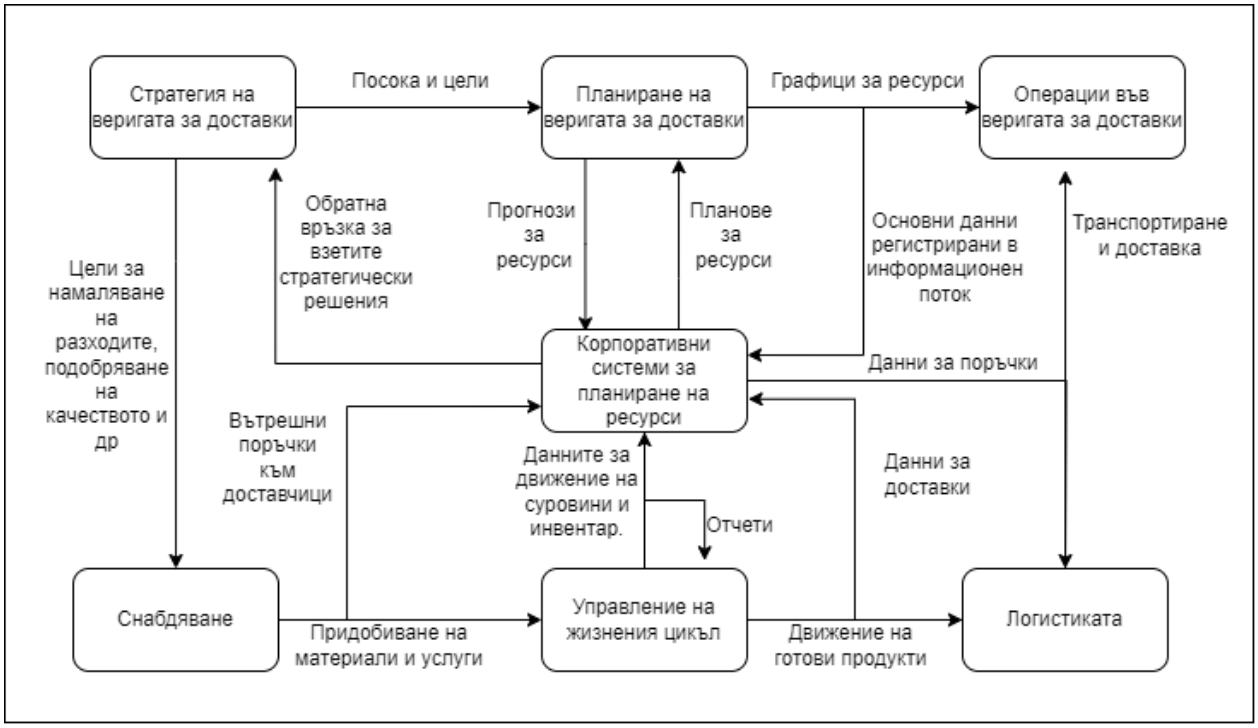
Компонент	Описание
Стратегия на веригата на доставки	Компонент, който установява целите и подхода на SCM, насочвайки как планирането и операциите следва да се съгласуват.
Планиране на веригата на доставки	Компонент, включващ прогнозиране на търсенето, разпределение на ресурсите и планиране на доставките. Той има за цел да балансира търсенето и предлагането и да подготви организацията за бъдещи нужди.
Операции във веригата на доставки	Компонент, в който се изпълняват SCM планове. Включва ежедневни дейности като изпълнение на поръчки, производство и транспортиране на стоки и продукти.
Корпоративни приложения във веригата на доставки	Технологични инструменти, които поддържат оперативните аспекти на SCM, включително системи за управление на отношенията с доставчици и клиенти, системи за управление на поръчки и системи за

	управление на инвентара.
Снабдяване	Процес на осигуряване на стоки и услуги, необходими за SCM, тясно обвързан със „стратегията на веригата на доставки“.
Управление на жизнения цикъл на продукта	Управлението на продукт от проектирането, производството, обслужването и приключването. Част е от цялостната стратегия и влияе върху „планиране на веригата на доставки“ с информация за състоянието и продуктите.
Логистика	Транспортиране на стоки в рамките на SCM, свързан с „операции във веригата на доставки“. Осигурява се физическият поток от продукти в синхрон с планирането и стратегията.

Източник: Runkler et al., 2019.

Въз основа на таблица 1.2 може да бъде създаден модел, описващ взаимосвързаността между основните компоненти на стратегията за доставка на продукция. Според изследване на Türkay et al. (2016) корпоративните ERP системи заемат централно място, управлявайки потока от стоки, материали, информация и капитал. Това дава възможност за интегрирането на различни процеси и операции в единна платформа. Използването на ERP системи в SCM се свързва с оптимизация на инвентара, планиране на производството, управление на складовите запаси и логистика (Димитров, 2020). ERP подкрепя сътрудничеството между различните отдели в организацията, както и с външни партньори. ERP системите улесняват автоматизацията на повтарящи се дейности, намалявайки вероятността от човешка грешка (Vasilev & Stoyanova, 2019).

На фиг. 1.1 е представен модел на взаимосвързаност между основните компоненти на SCM стратегията за доставка на продукция.



Фиг. 1.1. Модел на взаимосвързаност между основните компоненти на SCM стратегията за доставка на готова продукция

Адаптация по: Türkay et al., 2016; Vasilev & Stoyanova, 2019.

В модела, представен на фиг. 1.1, стратегията на веригата на доставки има водеща роля при определяне на посоката на взаимовръзките между процесите. Съществуват множество софтуерни подсистеми, които подпомагат модела, като например, софтуер за бизнес анализ (Verdouw, 2010). Съществуват и подсистеми за снабдяване, управление на склада и изпълнение на производството. Всяка от тези подсистеми изпълнява специфични функционалности – подсистемите за снабдяване отговарят за придобиването на материали, подсистемите за управление на склада регистрират данните за наличности, запаси и изготвят отчети за инвентаризация, а подсистемите за управление на производството агрегират информация за суровини, прекъсвания и работни инструкции (Văcar, 2019). Те също са свързани с ERP и са част от SCM.

При управлението на жизнения цикъл се следи движението на готовите продукти в производството и продажбите. Към тази част на модела

могат да се причислят подсистеми за управление на качеството, околната среда и взаимоотношенията с клиентите (от английски: Customer Relationship Management – CRM). Тези подсистеми работят с разнообразни видове данни. Подсистемите за управление на качеството използват данни от тестове, дефекти, рекламации и исторически записи за производствени процеси, като също така следят за съответствие със стандарти и регулации. Чрез подсистема за управление на околната среда се регистрират данни за емисии от вредни вещества, рециклиране, управление на отпадъци, енергийна ефективност и спазване на екологични норми. CRM подсистемата обработва данни за контакти и комуникация с клиенти, исторически данни за продажби и поръчки, предпочитания и поведение на клиентите, договорни условия и анализи на удовлетвореността.

Както бе отбелязано, логистиката е основен компонент в модела на взаимосвързаността на SCM стратегията, отговаряйки за доставката на стоки и продукти и работейки в синхрон с други компоненти (Vasilev & Cristescu, 2019). Логистиката в производствено предприятие обхваща планирането, организацията и контрола върху доставките, управлението на материални запаси, транспортирането и дистрибуцията на суровини и готова продукция. Тя е свързана с оптимизация на транспортните мрежи и насърчава внедряването на иновативни технологии, като устройства от типа „Интернет на нещата“ (от английски: Internet of Things – IoT) и услуги за машинно обучение (Sazanavets, 2022). За подпомагане на логистичните процеси се използват подсистеми за управление на транспорта (от английски: Transport Management Systems – TMS). Според Bier et al., (2019) тези подсистеми укрепват цялостната структура на веригата на доставки, работейки с разнообразни типове данни, включително:

- Географски данни: GPS координати за проследяване на превозните средства, карти и маршрути за планиране на пътуванията, данни за трафика за избягване на задръствания и закъснения;
- Данни за превозните средства: идентификационни данни на

превозни средства като: регистрационен номер, вид гориво, обем на резервоара, среден разход на гориво на час и в километри, поддръжка и сервизна история на превозните средства;

- Транспортни данни: разписания и графици за доставки, информация за товари (тип, размер, тегло, особености), информация за шофьорите (работно време и почивки);
- Финансови данни: разходи за транспорт (гориво, винетна и тол такси).

Когато се интегрират с ERP, системите от тип TMS съгласуват записаните данни с информация за:

- Инвентаризация и наличности на складовете;
- Статус и приоритети за поръчки и заявки;
- Фактуриране и плащания;
- Данни за местоположение на складови бази и разпределителни центрове;
- История на поръчките и предпочитания на клиентите.

Въз основа на централната роля на ERP в модела на взаимосвързаност между основните компоненти на SCM, в дисертацията се изследват основните аспекти на ERP: същност, модулност и възможности за интеграция. Подсистемите за управление на склада, качеството и жизнения цикъл се разглеждат като второстепенни, предназначени да подпомогнат работата на ERP. Поради тази причина тяхното детайлно проучване остава извън обхвата на настоящия труд. Въпреки това, тези подсистеми следва да бъдат взети под внимание при последващо надграждане на ERP и SCM.

Основният положителен ефект от прилагане ERP системи се състои в намаляване на количеството материали, незавършено производство и готова продукция, както и в съгласуване на графика на доставките с работата на отделните производствени звена и процесите по закупуване и доставка (González et al. 2024). Подобренията от въвеждането на ERP водят до увеличаване на броя на изпълнените поръчки, повишаване на качеството на

логистичното обслужване към клиентите, възможности за динамични промени в обема на поръчките, съкращаване на времето от приемането на поръчка до извършването на доставка. ERP технологията представлява стандартизирана последователност за изпълнение на различни бизнес функции (Templar et al. 2020). Алгоритмите в ERP са предварително зададени стъпки и правила, които дават възможност за оптимизация и автоматизация на процесите (Radev, 2023). ERP модулите са получили нормативна регламентация, което осигурява тяхната ефективност и съответствие с международно признати стандарти. Примери са Material Requirements Planning[4] (MRP) I и MRP II, за които са разработени и утвърдени международни ISO стандарти. Нормативната регламентация на MRP I и MRP II алгоритмите представляват унифицирани процедури, които установяват стандарти при интеграция между различни системи и организации.

На базата на класации на S&P Global Ratings[5] за компании за строителни материали към 2025 г., внедряването на ERP системи е от ключово значение за поддържане на интеграция на бизнес процесите (Cataldo et al., 2022; Милушева, 2023). Например, компании като CRH plc, Vulcan Materials Company, Martin Marietta Materials, Inc., Anhui Conch Cement и Heidelberg Materials AG използват SCM и ERP софтуер, включително SAP S/4HANA, Oracle SCM Cloud, Blue Yonder, Microsoft Dynamics 365 и Kinaxis RapidResponse. Тези софтуерни системи се конфигурират спрямо нуждите и изискванията на конкретното предприятие, като се взема под внимание неговите логистични и оперативни проблеми (Parusheva, 2019). По този начин SCM и ERP могат да бъдат специално адаптирани за строителната

[4] MRP е система за планиране и управление на материалите, която помага на предприятията да определят колко материали са необходими и кога трябва да бъдат закупени или произведени. Основната цел на MRP I е да осигури материалните нужди в производството, докато MRP II включва аспекти като минимизиране на запасите, увеличаване на капацитета на производствените мощности, планиране на работната дейност.

[5] S&P Global Ratings прави оценка на компаниите в индустрията за строителни материали. Класирането на компаниите става по рейтинг, перспектива, самостоятелен кредитен профил, профил на бизнес и финансов риск и оценка на ликвидността.

индустрия. Те са проектирани с цел да усъвършенстват процеса по вземане на решения, да подпомагат сътрудничеството между доставчици, диспечери и клиенти, както и да насърчават инициативи за устойчивост (Tang & Xia, 2023).

Една от водещите ERP системи в света е SAP. В настоящото изследване терминът „SAP“ се отнася до софтуера SAP S/4HANA. Системата SAP има възможности за интеграция в облачни платформи и отговаря на изискванията за централизирано управление на данни и логистични операции. Според Gaur (2020) в рамките на тази система се управляват редица функционални области на даден бизнес: човешки ресурси, финанси и функции за закриване на отчетен период, продажби, управление на клиенти, фактуриране и задължения, управление на инвентара и логистика. В SAP е наличен набор от функционалности, които покриват различни нужди на производствено предприятие (Schneider, 2020). Към април 2024 г. 86% от компаниите от Fortune 500 и 92% от компаниите от Forbes Global 2000 са клиенти на SAP. Над 77% от приходните транзакции в света преминават през тази система, която се използва от над 400 000 организации в 180 държави (Duff, 2024).

Внедряването на SAP може да отнеме значително време и ресурси в зависимост от конкретната компания и редица други фактори. Независимо от това, този корпоративен софтуер интегрира различни бизнес процеси в една платформа. Според изследвания (Gargeya & Brady, 2005; Ojra et al., 2021), функционалностите в SAP са разпределени в отделни модули, които могат да бъдат групирани в три основни категории:

- Логистика: обхваща модули за управление на продажби и дистрибуция, управление на материалите, производство, планиране и управление на качеството.
- Счетоводство: счетоводните модули включват финансово счетоводство, контрол и одит. Те администрират информация за финансовото състояние на организацията и подпомагат анализа и

управлението на разходите.

- Човешки ресурси: в тази категория се управляват основни дейности и анализи в областта на човешките ресурси, ведомост на заплатите и планиране на работната сила.

В рамките на дисертацията се анализират възможностите за интеграция и автоматизация в модулите за управление на продажби и дистрибуция. Автоматизацията в тези модули се счита за ефективен начин за намаляване на вероятността от човешка грешка и повишаване на ефективността на логистичните и дистрибуторските операции (Barata, 2022). Модулите за управление на продажби и дистрибуция се поддържат данни за клиенти, продукти, поръчки, бизнес партньори и доставки.

Използването на основни и трансакционни данни в SAP модулите за управление на продажби и дистрибуция е предмет на изследване от редица автори (Hildebrand, 2018; Pjp, 2023). В своите проучвания, авторите използват понятието „мастър данни“, за да обозначат информация, която остава относително постоянна във времето и служи като референтна точка за разнообразни бизнес операции и процеси. Мастър данните са структурирани и организирани в т.нар. „организационни единици“, които формират рамката, в която модулите функционират.

Според Bilovodska et al. (2018), мастър данните са основни градивни елементи за транзакционните данни. Мастър данните остават относително статични, докато данните за транзакциите (продажби, доставки и фактури) се променят непрекъснато. SAP е интегрирана система, в която се улеснява обмена на данни между различните модули. Например, при създаване на клиентска поръчка в модула за управление на продажби, данните автоматично се предават към модула за управление на материалите за проверка на наличността и към модула за финансово счетоводство за фактуриране.

На базата на проучване на Magal и Word (2013), в табл. 1.3 са представени част от организационните единици, интегрирани в SAP

модулите за управление на финанси, продажби, дистрибуция и управление на материалите.

Таблица 1.3

Организационни единици в SAP модули за управление на финанси, продажби и материали

Финанси	Продажби и дистрибуция	Управление на материали
Сметкоплан (Chart of Accounts)	Търговска организация (Sales Organization)	Завод (Plant)
Компания (Company)	Дистрибуционен канал (Distribution Channel)	Местоположение на склада (Storage Location)
Код на компания (Company Code)	Дивизия (Division)	Организация на покупките (Purchasing Organization)
Бизнес област (Business Area)	Продажбена област (Sales Area)	Група покупки (Purchasing Group)

Източник: Magal & Word, 2013.

При продажбите и дистрибуцията, търговската организация (от английски – sales organization) е поставена на най-високото ниво на управление. Отчитане на продажбените дейности се извършва именно на това ниво (Becker et al., 2016; Von Aspen, 2020). Финансовите резултати се отразяват в сметкоплана на отделна „компания“, което представлява юридическо лице и самостоятелна счетоводна единица. Всяка компания може да има една или повече търговски организации, като продажбите на всички търговски организации, свързани с определена компания, трябва да бъдат консолидирани в нейните финансови отчети. От друга страна, на първо ниво в модула за управление на материалите се намира заводът и прилежащото производствено съоръжение или дистрибуторски център.

Подсистемите и организационните единици, могат да бъдат класифицирани както като SCM, така и като допълващи ERP, в зависимост от тяхната основна функционалност. На тази база, един от основните проблеми,

с които се сблъскват производствените компании, е необходимостта да се координират множество процеси и звена, които са пряко или косвено свързани с изпълнението на поръчките (Katsaliaki et al., 2021). Както бе показано в табл. 1.3, компаниите трябва да събират, обработват и анализират данни от различни източници – вътрешни системи за управление на производството, транспорт, склад, както и от външни доставчици.

ERP и SCM системите не винаги успяват да обхванат всички процеси в реално време, което може да доведе до закъснения, пропуски в изпълнението на поръчките и проблеми с точността на доставките. Например, ERP системите са ориентирани към вътрешното управление на ресурсите и производството, но при липсата на интеграция с външни логистични системи и вериги на доставка, координацията по изпълнението на поръчките може да се усложни (Frey, 2023). SCM системите са създадени да осигурят видимост на процесите във веригата на доставки, но когато не са интегрирани с ERP, актуалността на данните за производствения капацитет и наличността на продукти може да бъде нарушена. В резултат, възникват проблеми, които влияят негативно както върху бизнес клиентите, така и върху рентабилността на компанията.

Възможно е компаниите да изпитват затруднения при изпълнението на поръчките, което да доведе до закъснения с доставките, увеличени разходи и пропуски в изпълнението. Производствените предприятия разчитат на доставчици и партньори от различни части на света, което създава допълнителни проблеми в управлението на доставките и логистиката. „Удължаването“ на веригата на доставки и необходимостта от координация с международни партньори означава, че компаниите трябва да бъдат гъвкави и бързи в реакциите си на промените в пазарната среда. Т.нар. „глобални вериги на доставки“ изискват управление на различни транспортни и логистични канали, спазване на международни разпоредби и стандарти за качество. Глобализацията създава непредсказуемост в търсенето на продукти. Дружествата са принудени да реагират на сезонни промени или

икономически кризи (Василев, 2017).

При въвеждането на ERP системи, обикновено основно внимание се отделя на управлението на вътрешните операции и ресурси, като например производствен капацитет, складови наличности и финансова отчетност. В много случаи липсва свързаност на ERP с логистични канали или външни доставчици, което затруднява актуалното проследяване на поръчките. SCM системите предоставят инструменти за управление на логистиката, транспорта и доставките, но ако не получават точни данни за производствените възможности и наличните ресурси, те не могат да оптимизират напълно процесите (Schachenhofer, Kummer & Hirsch, 2023). Например, ако SCM системата не разполага с актуална информация за наличността на даден продукт, тя може да не успее да прогнозира точно времето за доставка или да предложи алтернативни решения, когато продуктът не е наличен. Проблемите могат да станат критични и при значителни промени в търсенето или затруднения с доставките. Липсата на интеграция между ERP и SCM създава необходимост от персонализирана система, разработена и конфигурирана спрямо нуждите и изискванията на конкретно производствено предприятие. Системата трябва да обединява данни и процеси от основните компоненти на веригата на доставка.

1.2. Рационализиране на процесите по управление на поръчките чрез персонализирана информационна система, конфигурирана към конкретна компания

На базата на анализа, представен в раздел 1.1, може да се заключи, че управлението на клиентските поръчки в обхвата на SCM се осъществява посредством разнообразни софтуерни системи, използвани в различни етапи от стратегията за доставка на продукция. Софтуерни решения като SAP S/4HANA, Oracle NetSuite, JD Edwards EnterpriseOne, Microsoft Dynamics 365 и Infor CloudSuite са базирани на ERP и SCM технологии, поддържащи

анализи и автоматизация на бизнес процеси.

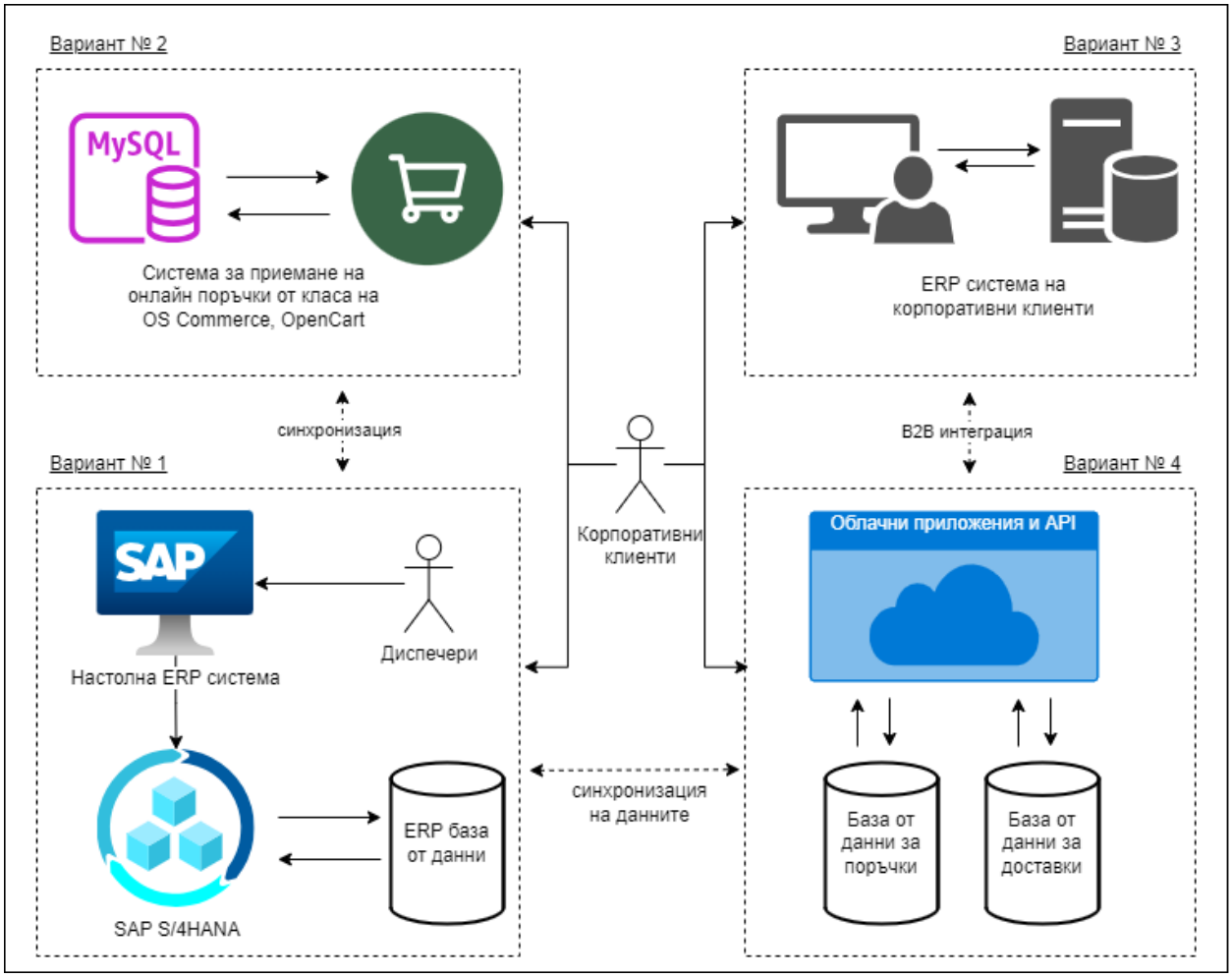
Oracle NetSuite представлява изцяло облачна ERP платформа, подходяща за предприятия с различен мащаб, която предлага набор от функционалности, сред които модули за финанси, CRM и електронна търговия. JD Edwards EnterpriseOne, също разработка на Oracle, може да бъде внедрена както локално, така и в облак, и е приложима в редица отрасли и бизнес процеси. Microsoft Dynamics 365 интегрира ERP и CRM функционалности. Microsoft Dynamics се свързва директно с други приложения на Microsoft като Word, Excel, OneNote, Outlook, OneDrive и Teams, за да осигури съвместна работа и автоматизация на работните процеси в рамките на единна платформа (Парушева & Александрова, 2022). Infor CloudSuite е облачна платформа с индустриално-специализирани функционалности, базирана на Infor OS, която дава възможност за надграждане на съществуващи ERP и CRM системи.

Софтуерни продукти за онлайн поръчки, например OS Commerce и OpenCart, служат като инструменти за електронно управление на търговските процеси в производствените предприятия. С помощта на подобни платформи организациите приемат и изпълняват поръчки, управляват продуктовите каталози, обработват плащанията и автоматизират комуникацията с бизнес клиентите.

Както бе отбелязано в раздел 1.1, компаниите, които разчитат единствено на традиционни ERP и SCM системи, могат да изпитват затруднения при адаптиране към бързо променящите се пазарни условия и технологични иновации. Тези системи обикновено не осигуряват необходимата гъвкавост, за да отговорят адекватно на нововъзникнали изисквания. В съвременната бизнес среда, все по-належащи стават достъпът до данни в реално време, прозрачността на операциите и способността за бърза реакция при непредвидени обстоятелства, поради което внедряването на персонализирани решения, съобразени с конкретните потребности на дадена компания, има все по-голямо значение.

Според изследване на Knolmayer (2012), разглеждащо софтуерните системи за управление на поръчки, ERP системите предоставят базова рамка, но сами по себе си не успяват да обхванат динамичния характер на цялостната верига на доставки. Констатацията е свързана с конкретно решение на дадено производствено предприятие да премине от една ERP система към друга. В такъв процес предприятието може да се сблъска с редица оперативни проблеми, тъй като интегрираните модули в действащата ERP платформа налагат пълна промяна на познатите интерфейси и процедури за въвеждане и извличане на данни.

Анализът на различни литературни и онлайн източници разкрива липсата на специално разработен технологичен модел, фокусиран върху управлението на клиентски поръчки в производствено предприятие, който да адаптира ERP и част от SCM стратегията за доставка на готова продукция. Преглед на статии, публикувани в научни списания като Journal of Supply Chain Management, International Journal of Production Economics и Supply Chain Management: An International Journal, както и на доклади от международни конференции (International Conference on Logistics and Supply Chain Management), разкрива разнообразни подходи за обработка на клиентски поръчки (Василев, 2018; Cichosz et al., 2020; Agarwal, 2021). Основавайки се на изводите от посочените проучвания, на фиг. 1.2 се представя модел, който обединява четири варианта за приемане на клиентски поръчки в производствено предприятие.



Фиг. 1.2. Технологичен модел, представящ различни варианти за управление на клиентски поръчки в производствено предприятие

Адаптация по: Василев, 2018; Cichosz et al., 2020; Agarwal, 2021.

На фиг. 1.2 са представени четири различни варианта за управление на поръчки от бизнес (корпоративни) клиенти. Всеки от тях предлага собствен подход за обработка и синхронизация на данните, но общата им основа е ERP системата. При първият вариант поръчките се въвеждат ръчно от диспечери директно в настолна ERP система. Данните се съхраняват в база на ERP. Макар че този подход осигурява контрол върху процеса, той изисква постоянен човешки ресурс, който в определени моменти може да се окаже недостатъчен. Това би довело до забавяния и проблеми при обслужването на множество клиенти.

Във втория вариант се интегрира платформа за електронна търговия

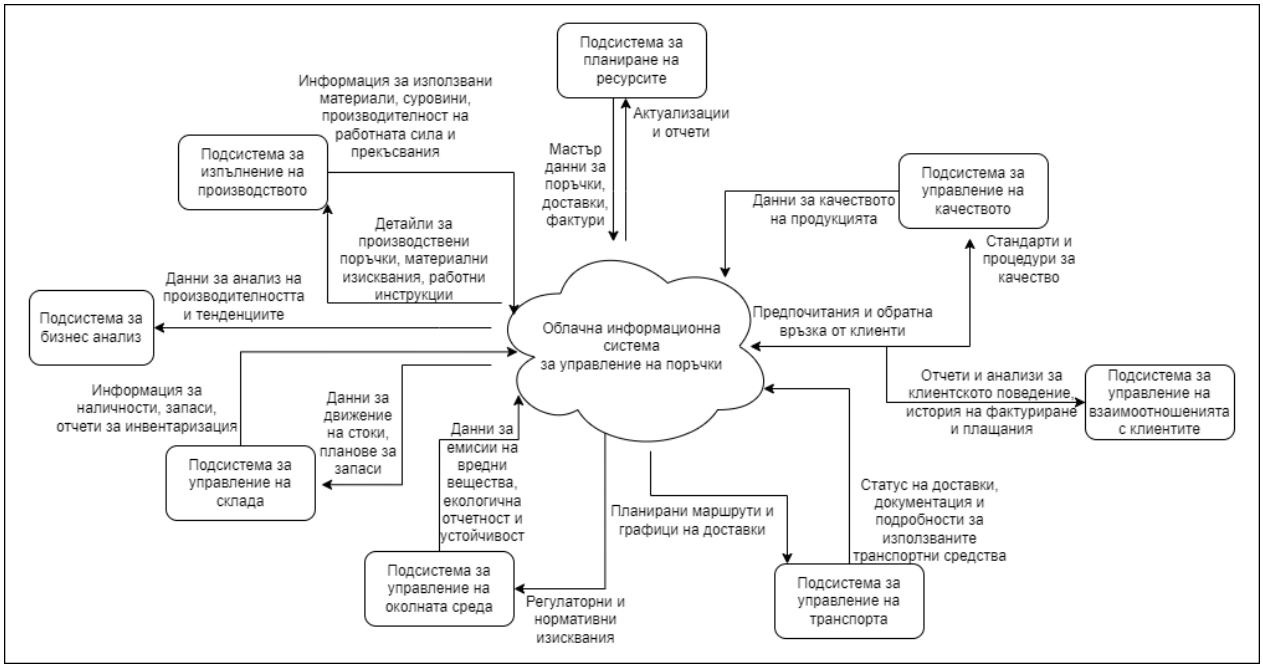
от типа на OpenCart или OS Commerce. Платформите работят на отделни сървъри, със собствени бази от данни (например MySQL), които трябва да се синхронизират със съответните бази от данни на ERP системата. Платформата за електронна търговия дава възможност на бизнес клиентите сами да въвеждат своите поръчки. По този начин се облекчава натоварването на диспечери и оператори. При този подход се изисква допълнителна поддръжка и функционалността е ограничена до възможностите на самата платформа за електронна търговия. Необходимо е редовно да се актуализират всички софтуерни компоненти, за да се избегнат потенциални уязвимости.

Третият възможен подход е интеграция от тип Business-to-Business (B2B), при която ERP системата на производственото предприятие комуникира директно с ERP системите на клиентите. Този метод има висока степен на автоматизация, но изисква съвместимост между различните платформи. Въпреки че тази практика се прилага често, основният недостатък е свързан с необходимостта от предоставяне на достъп до програмния интерфейс на основната ERP система. Според доклад на Kaspersky (2024), отворените интерфейси на водещи ERP платформи значително повишават риска от кибератаки.

При интеграция на външни партньори във вътрешната корпоративна мрежа, се налага прилагане на многостепенна защита, тъй като при наличие на уязвимости или неправилни конфигурации може да се стигне до пробиви в системата. В доклада на Kaspersky се подчертава важността на специализирани системи за откриване на аномалии, които следят за подозрителни активности, тъй като подобни проблеми оказват негативно въздействие върху цялостната дейност на ИТ отдела, който носи отговорност за сигурността на данните и поддръжката на различни системи.

Липсата на единно становище и недостатъците на първите три варианта насочва към управление чрез персонализирана информационна система, работеща като междинен софтуер. В доклад на Шишманов и Маринова-Костова (2024) се проучва централизиран слой, интегриращ набор

от корпоративни приложения. Това проучване подкрепя идеята за конфигурация според нуждите и изискванията на конкретното предприятие. На базата на доклада и изследванията на други автори (Verwijmeren, 2004; Caserio & Trucco, 2018), фиг. 1.3 представя модел на централизирана облачна система за управление на поръчки, която осигурява адаптивност и интеграция с различни външни и вътрешни подсистеми.



Фиг. 1.3. Модел на централизирана облачна система за управление на поръчки

Адаптация по: Verwijmeren, 2004; Caserio & Trucco, 2018; Шишманов & Маринова-Костова,2024.

На фиг. 1.3, част от вариант №4 от технологичния модел за управление на клиентски поръчки в производствено предприятие, е представена персонализирана облачна система. Тя интегрира набор от външни и вътрешни подсистеми. Чрез нея се осъществява приемането на заявки и се предоставят данни на бизнес клиентите чрез мобилни приложения. Системата е проектирана да консолидира данни от подсистемите на компонентите на SCM стратегията, изложени в таблица 1.2 от преходната точка, както и да приема информация на крайните

потребители.

При по-нататъшното развитие на системата е необходимо да се отчетат специфичните изисквания на конкретното предприятие, да се предвиди бъдещият растеж и цялостното развитие на бизнеса. В системата се интегрират различни процеси за оптимизация на потока от информация между различните звена – от производството до потребителя. Централизираното информационно управление стои в основата на системата, което дава възможност за рационализация и подобряване на достъпа до актуална информация (Сълова, 2019).

Подсистемите обхващат различни дейности, свързани с управлението на вътрешнофирмени ресурси – логистика и складови наличности. Също така обхващат и външни взаимодействия: клиентски връзки и контрол на качеството на продуктите. Подсистемите са взаимосвързани по такъв начин, че да осигуряват постоянен поток на информация в организацията (Aleksandrova, 2021; Sullivan & Kern, 2021).

Както бе отбелязано, ERP допринася за събирането на информация за наличните ресурси, планиране на производството и поддържане на баланс между наличностите и търсенето (Kakhki & Gargeya, 2019; Rajapakse, 2023). Подсистемата за управление на качеството осигурява наличието на постоянен мониторинг и контрол върху качеството на продуктите и процесите в организацията. CRM подсистемата се използва за връзка с клиентите и управление на маркетингови стратегии (Hasim et al., 2018; Александрова, 2020). TMS подсистемата координира превозните средства, използвани за доставка. Подсистемата за управление на склада администрира складовите процеси, инвентара и управлява наличностите. Подсистемата за бизнес анализ извлича и агрегира данни от другите подсистеми, за да определи производителността и пазарните тенденции. Посредством тази подсистема се дефинират бизнес метрики, които допринасят за стратегическото развитие (Ren et al., 2019; Schniederjans et al., 2020). Тя дава прогнози, които значително подпомагат вземането на бизнес решения

(Ramakrishna, 2022). Подсистемата за мониторинг на производството контролира производствените процеси и разпределя информацията към останалите подсистеми, следейки текущото състояние на производствените операции.

В рамките на предложената персонализирана информационна система, която има за цел да оптимизира управлението на поръчки, е предвидена конфигурация към определена SAP S/4Hana ERP система, както и частична интеграция с CRM и TMS, чрез технически модули от категорията на SAP Netweaver Gateway[6] (Bönnen et al. 2018). Основното предназначение на системата е да отговори на потребностите на бизнес клиентите за достъп до актуална информация за поръчките и доставките, както и да осигури възможност за въвеждане или промяна на данни.

Персонализираната система дава възможност за внедряване на автоматизирани процеси и алгоритми за непрекъснато подобрение, които да се адаптират към динамиката на пазара и променящите се изисквания. Тя се базира на взаимодействията между различни софтуерни компоненти, поддържа непрекъсната обратна връзка с бизнес клиентите и извършва анализ на техните предпочитания.

За да се постигне това, основните ERP, CRM и TMS подсистеми, работещи на локални сървъри в предприятието, откриват към персонализираната информационна система както входящи, така и изходящи интерфейси. Например, заявките от клиенти, подадени чрез мобилно приложение, се обработват чрез входящ интерфейс. В предходната точка, те бяха отбелязани като транзакционни данни. Статични мастър данни, например име на доставчик и номер на превозно средство, се предават чрез изходящ интерфейс. Персонализираната информационна система синхронизира основните и трансакционните данни в свои собствени бази чрез криптирани канали, за да ги предостави на бизнес клиентите. По този

[6] SAP NetWeaver Gateway е технология, която позволява свързване между SAP и персонализирани платформи. NetWeaver предоставя начин за интеграция чрез използване на API.

начин се способства защита на вътрешните подсистеми, избягва се потенциалният риск от уязвимост при евентуален пробив или хакерска атака.

Наличието на централизирана система дава възможност за интегриране на отделните търговски организации (или подразделения) на производственото предприятие. Анализът в реално време на различни показатели като наличност на ресурси и време за изпълнение подпомага тяхната съвместна работа. В случай че една от организациите се окаже претоварена в даден момент, друга може да поеме и обслужи част от поръчките, като е необходимо процесите по пренасочване да бъдат автоматизирани от персонализираната система. При получаване на заявка от бизнес клиент, системата проверява наличните ресурси. Ако те не са достатъчни или не отговарят на изискванията, се генерира известие към операторите или се задейства процес по преназначаване. Алгоритми за прогнозиране правят предложения кои доставчици е най-подходящо да обслужат поръчките за деня. Тези прогнози се предоставят като план-график на диспечерите.

За постигане на целта, в реално време се анализират данните, получени от IoT устройства, прикрепени към превозните средства, които предоставят постоянна обратна връзка относно местоположение, скорост и състояние на товара. Въз основа на тази информация, при възникване на непредвидени обстоятелства (например: закъснения поради трафик на пътя или повреда на превозното средство), може да се актуализира първоначалният план-график и да се предложат алтернативни маршрути. Докато диспечерите следят движението на превозните средства, клиентите получават своевременни насочени известия (нотификации) за приблизителното време на пристигане или евентуални промени във времевите интервали за доставка. При необходимост, персонализираната система може да пренасочи ресурси, за да осигури доставянето на продукцията навреме.

Производствените предприятия могат да комуникират с клиентите си

едновременно чрез имейли, телефонни обаждания до центъра за обслужване, мобилни и уеб приложения, за да се улесни достъпа до актуална информация и да се осигури бърза обратна връзка. Ако каналите за връзка не са интегрирани по подходящ начин, могат да възникнат определени проблеми. Например жалба, подадена по имейл, да бъде пропусната от диспечера на смяна. Централизираната система дава възможност за интегрирането на различните комуникационни канали посредством т.нар. „омниканален“ (от английски – omnichannel) подход. Целта на омниканалния подход е да осигури последователно и персонализирано взаимодействие (Димитрова, 2023). По този начин производственото предприятие може да осигури потребителско обслужване, което съответства на предпочитанията на бизнес клиентите.

В проучване, Thomas et al. (2024) установяват, че компаниите, използващи този подход, получават 30% по-висока оценка от клиентите си в сравнение с компании, които разчитат на отделни, не интегрирани канали. В статията се подчертава и ролята на изкуствения интелект и машинното обучение в обслужването на клиенти.

Чрез персонализираната система могат да се внедрят алгоритми за извличане на съдържание от входящи имейли. Автоматично извлечените данни следва да бъдат въведени в системата, но всеки запис трябва да бъде маркиран със специален статус „изисква се преглед от диспечер“. Това осигурява възможност за верификация и минимизиране на потенциални грешки. При потвърждение от страна на диспечера, данните се синхронизират с ERP подсистемата. По този начин се избягва ръчно въвеждане. Системата автоматично генерира и изпраща отговор на имейла, добавя хипервръзка към мобилно приложение, което да предоставя информация за конкретните поръчки и доставки в реално време.

По този повод, въз основа на емпирични данни като история на поръчките, свързаната информация за доставки и алгоритми на изкуствения интелект от типа „обучение с утвърждение“, могат да се създадат модели на

поведение на бизнес клиентите, които дават възможност за прогнозиране на техните нужди и откриване на подходящи продукти и услуги за тях. На базата на тази модели може да се приложи и динамично ценообразуване. То се формира на базата на взаимоотношения, търсене и предлагане. Друг фактор при ценообразуването са отказаните поръчки. Чрез използването на алгоритми за времеви редове може да се определи вероятността клиент да откаже поръчка, да се прогнозира бъдещото търсене или вероятността да премине към конкурентно предприятие.

Персонализираната информационна система следва да се внедри поетапно. Възможно е да се започне с пилотно тестване в малка част от компанията, по-конкретно в отделите по „Обслужване на клиенти“ и „Приемане на поръчки за продажба“. За самото внедряване следва да се използват „*гъвкави*“ (от английски – agile) подходи, които дават възможност за плавна адаптация и се придържат към текущите нужди и обратната връзка от потребителите (Маринова, 2015).

В началния етап, достъп до мобилните и уеб приложения може да бъде предоставен на ограничен брой клиенти, като 15% до 20% от поръчките и доставките се обслужват чрез персонализираната система. Очаква се, в определен момент 100% от поръчките да се заявяват, променят и отхвърлят чрез системата, а всички превозни средства да могат да се проследяват в реално време.

Въз основа на разгледаните по-горе корпоративни софтуерни системи от типа на SAP S/4HANA, Oracle NetSuite и Microsoft Dynamics 365, може да се заключи, че използването на облачни услуги за внедряване на персонализирана информационна система е надежден подход, който подпомага адаптацията към нуждите на бизнеса и рационализира оперативните процеси. При този подход се насърчават непрекъснати подобрения и иновации. Предоставя възможност за надграждане и интеграция на нови функционалности в различни компоненти на системата.

1.3. Възможности за централизация на процесите по управление чрез прилагане на облачни технологии

Редица автори и изследователски компании (Partida, 2023; Roy, 2023; Microsoft Research, 2024) изследват значението на облачните технологии за постигане на оптимално функциониране на процесите във веригите на доставки. Все повече производствени предприятия предпочитат да развиват и внедряват своите софтуерни продукти в облачна среда. Чрез облачните услуги се разработват и внедряват информационни системи, даващи възможност на множество участници и подсистеми във веригата на доставки да работят съвместно и координирано. Софтуерната архитектура на това софтуерно решение дава възможност за обмен на данни в реално време и осигурява контролиран достъп до основна информация.

Може да разграничим общия термин „*облак*“, който често се използва като синоним на „*облачни изчисления*“, от термина „*облачни технологии*“, описващ технологични компоненти, софтуерни инструменти и архитектурни подходи за реализиране. Облакът и облачните изчисления се асоциират с модела за предоставяне на изчислителни ресурси от тип „*pay as you go*“, при който се заплаща само за реално използваните ресурси. Облачните технологии са по-общ термин, включващ виртуализация, контейнеризация и автоматизация на инфраструктурата. Понятието „*облачната инфраструктура*“, от своя страна, представлява хардуерната и софтуерната основа на физическата и виртуалната среда за работа на облачните услуги.

Концепцията за облачни технологии варира, например *National Institute of Standards and Technology*[7] (2011) определя облачните изчисления като „модел за осигуряване на мрежов достъп, при поискване, до споделен пул от конфигурируеми изчислителни ресурси, които могат бързо да бъдат предоставени и внедрени с минимални усилия“.

[7] National Institute of Standards and Technology (NIST) е американска правителствена агенция, която е част от Министерството на търговията на САЩ. Основни функции и дейности са: разработване на стандарти, изследвания, развитие и киберсигурност и технологии за информация.

Според дефиницията на организацията *Cloud Native Computing Foundation*[8] (2018) „облачните технологии дават възможност на организациите да разработват и изпълняват приложения в съвременни, динамични среди – публични, частни и хибридни облаци – използвайки мрежи от услуги и микроуслуги. Сред характеристики на тези системи са устойчивостта, високата наличност, достъпността, мащабируемостта и управляемостта, които са от съществено значение за разнообразни бизнес единици. Автоматизирането на процесите дава възможност на инженерите да внедряват софтуерни промени с минимални усилия“.

Посочените определения предлагат различни тълкувания, но преобладава схващането, че системите, базирани на облачни технологии, се характеризират предимно с висока производителност и ниска латентност (Smith, 2025).

Производителността измерва времето между Интернет заявката на потребителя и последващия отговор на системата. Следователно, производителността служи като показател за ефективност, свързан с удовлетвореността на потребителя. Бързото време за реакция обикновено означава оптимална производителност на системата, което води до положително потребителско изживяване, докато забавянето може да е показател за неефективност. Heusser (2019) представя общ метод за концептуализиране на производителността чрез следното уравнение:

Време за отговор = Време за обработка + Време на изчакване.

В посоченото уравнение *времето за отговор* е общото време от момента, в който потребителят изпрати заявка, до момента, в който получи отговор. Счита се, че това представлява интервалът от време, в което потребителят изчаква, за да получи резултат след започване на действие (Esposito, 2016). *Времето за обработка* е времето, необходимо на системата

[8] Cloud Native Computing Foundation (CNCF) е основана през 2015 г. като част от Linux Foundation. Целта ѝ е да помогне на предприятията и разработчиците да изграждат и управляват приложения в облачна среда. CNCF насърчава интегрирането на облачни технологии чрез предоставяне на инструменти, стандарти и практики.

за изчисляване на резултата след получаване на заявката. То включва задачи, свързани със заявка към база от данни, обработка и връщане на резултата към потребителя. *Времето на изчакване* представлява времето, в което заявката се намира в „опашка“, преди да бъде обработена. В система с голям трафик от данни могат да постъпят няколко заявки едновременно. Ако системата не може да ги обработи наведнъж, някои заявки трябва да изчакат. По този начин се увеличава времето за изчакване, така че чрез разделяне на времето за отговор на неговите компоненти системните администратори и разработчиците могат да определят областите за подобрение. Например, ако времето за обработка е дълго, може да е необходима оптимизация на алгоритми или програмен код. Ако времето за изчакване е дълго, това може да служи като показател, че системата се нуждае от по-добро балансиране на натоварването или увеличен капацитет за обработка.

Проучвания на източници в областта (Betts et al. 2013; Garrett et al. 2013) показват, че способността на система да управлява ефективно увеличеното работно натоварване се отнася до мащабируемостта. Описват се две измерения на мащабируемостта: вертикална и хоризонтална. Според Henning & Hasselbring (2022) вертикалната мащабируемост се осъществява чрез подмяна на хардуерни устройства с по-добри параметри (процесор, памет или пропускателна способност на мрежата). За разлика от това, хоризонталната мащабируемост се постига чрез добавяне на допълнителни хардуерни модули. Вместо да се подобрява един сървър, за да се разпредели натоварването, се създават множество виртуални сървъри, работещи на различен физически хардуер. Този подход допринася за висока достъпност и толерантност към грешки.

По този повод, редица автори разглеждат показателя за оперативна достъпност (от английски – high availability) като измерител за качество (Sheldon et al., 2024). Ниво на достъпност се определя като частта от времето, през което дадена услуга е функционална и достъпна. Според Atchison (2020) нивото на достъпност може да бъде изразено като процент от времето на

работа (uptime) спрямо сумата от времето на работа и времето, в което облачната услуга не функционира (downtime):

$$Availability = uptime / (uptime + downtime).$$

Според някои автори, абсолютната 100% наличност е нереалистична поради необходимостта от поддръжка и настройки (Martin, 2017). Забавянето в обслужването може да бъде индикатор за проблеми, които възникват при определени условия, като конкуренция за ресурси или хардуерни проблеми. Статистически, 90% наличност се равнява на над 2 часа дневно или 36 дни годишно, в които облачните услуги не функционират. 95% наличност се равнява на около час дневно или 18 дни годишно. Облачните услуги обикновено поддържат наличност от 99% или дори 99.9% (известни като "три деветки"), което означава, че системата е офлайн по-малко от 1,5 минути на ден.

За да се следят времената за отговор, обработка и изчакване, предприятия и облачните доставчици определят нива на обслужване (от английски: Service Level Agreement – SLA), чрез договорни споразумения между двете страни. Според Debski et al. (2018) SLA включват ангажименти за производителност, латентност и време за реакция. Индивидуалните цели, определени за една система, се наричат цели за ниво на обслужване (Service Level Objective – SLO). Всеки SLO определя целева стойност или диапазон за специфични системни дейности, например време за реакция под 100 ms за 90-ия процент. Наред с това, индикаторът за ниво на обслужване (Service Level Indicator – SLI) е количествена мярка за определяне на съответствието със SLO. Той администрира данни за ефективността в реално време, които се събират и оценяват дали се постигат SLO. Според нас, SLA, SLO и SLI са ключови инструменти за поддържане на качество при облачните услуги. Докато SLA обикновено се формулират от юридическите екипи, SLO и SLI спадат към отговорностите на софтуерните архитекти.

Изхождайки от казаното дотук, можем да обобщим, че облачните технологии осигуряват инфраструктура за приложения чрез ресурси за

сървъри, операционни системи, защитни стени и балансиране на натоварването. Хардуерът е разположен в център за данни, в който ИТ специалисти създават виртуални ресурси, без необходимост от закупуване или поддръжка на устройства, за разлика от традиционния подход, при който хардуерът е собствен и изисква цялостно управление и поддръжка от ИТ отдела (Endo et al. 2016).

В областта на облачните технологии можем да разграничим публични, частни и хибридни облачни архитектури. Публичните облачни услуги предоставят изчислителни ресурси чрез Интернет, което дава възможност за достъпност в голям мащаб, без да са необходими значителни капиталови инвестиции в реална инфраструктура. За разлика от това, частните облачни услуги представляват специално изградени среди, проектирани и управлявани за конкретни предприятия. Това осигурява повече защита и контрол върху данните, но също така води до по-голяма отчетност за администриране и поддръжка на инфраструктурата (Радев, 2015). Хибридните облачни услуги са резултат от комбинацията на публични и частни, което дава възможност на предприятията да комбинират мащаба и рентабилността на публичните облаци с персонализираната сигурност и контрола на частните. Като обобщение, в табл. 1.4 е направен обзор на някои от основните характеристики и ограничения на трите вида.

Таблица 1.4

Сравнение между публични, частни и хибридни облачни услуги

Характеристика	**Публичен**	**Частен**	**Хибриден**
Хостинг	Услуги, предоставяни извън организацията през интернет.	Инфраструктура, проектирана и за нуждите на определена организация.	Комбинация от обществени и частни облаци, позволяваща обмен на данни и приложения между тях

Характеристика	Публичен	Частен	Хибриден
Икономическа ефективност	Висока, поради модела плащане според употребата и липсата на инвестиции във физическо оборудване.	По-ниска в сравнение с обществените облаци поради началната инвестиция в хардуер и разходи за поддръжка.	Осигурява комбиниран подход, при който организациите могат да се възползват от икономическите предимства на публичните облаци за некритични дейности, като същевременно запазват критичните задачи в частни облачни среди.
Мащабируемост	Висока мащабируемост, с ресурси на разположение от тип „при поискване“ за удовлетворяване на завишени търсения и обратно.	Мащабируемостта е ограничена от капацитета на физическата инфраструктура, изискваща планиране и инвестиции за увеличение.	Подход, при който част от операциите се поддържат в частния облак, а друга част в публичния, според нуждата от сигурност.
Сигурност	Високи нива на сигурност, основаващи се на водещи доставчици на услуги.	Най-високо ниво на сигурност сред трите вида, тъй като ресурсите не се споделят с други организации.	Предоставя комбиниран модел, като запазва чувствителните данни и приложения в защитения частен облак, докато други операции могат да се разполагат в публичната среда.
Контрол	Ограничен контрол над изчислителната среда и основната инфраструктура.	Пълен контрол над средата и инфраструктурата, позволяващ персонализирани настройки.	Най-сложен за изпълнение и поддръжка, тъй като ресурсите се споделят между различни видове услуги.

Източник: Dotson, 2019.

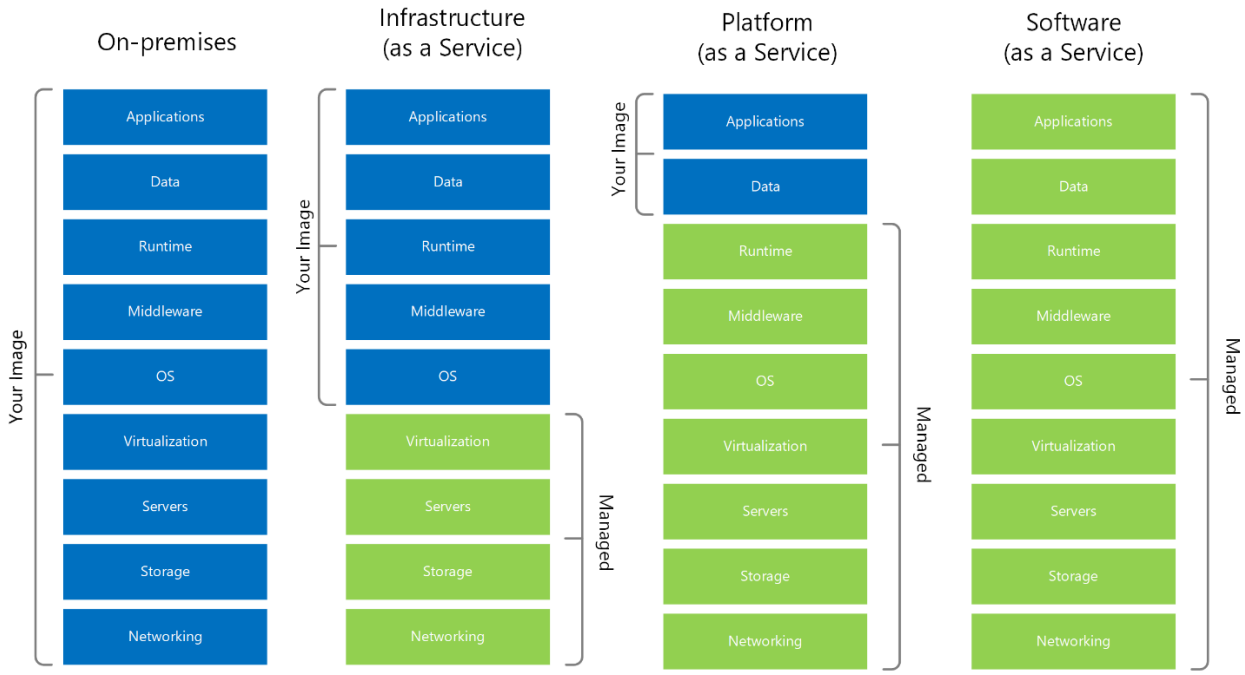
От трите вида облачни услуги, публичните изглеждат най-подходящи поради своята висока степен на мащабируемост, икономическа ефективност и публичния характер на услугите, насочени към крайни потребители. Публичните облачни услуги дава възможност за мащабиране на ресурсите в отговор на променящото се търсене, предоставяйки необходимата компютърна инфраструктура без нужда от инвестиция в сървърен хардуер.

Макар да съществуват потенциални рискове, свързани със сигурността, те могат да бъдат минимизирани чрез избор на доставчик на облачни услуги, сертифициран по ISO/IEC 27001[9]. Следователно, предимствата на публичните облачни услуги надхвърлят потенциалните недостатъци.

Съществува друга класификация на облачните услуги, която разграничава три модела на облачни изчисления: IaaS, PaaS и SaaS. При IaaS модела ИТ специалистите имат достъп до виртуализирани изчислителни ресурси, виртуални машини и мрежи. При IaaS моделът се възлага отделна физическа среда, освобождавайки компаниите от пряко управление на слоевете за виртуализация. Същевременно се запазва контрол върху операционните системи и приложения. Надграждайки над IaaS, при PaaS доставчикът на облачната услуга отговаря за ОС, междинния софтуер и среди за изпълнение. По този начин се дава възможност на разработчиците да се съсредоточат единствено върху създаването и внедряването на приложения. В третия SaaS модел се администрират цялостни приложения, достъпни през интернет. Благодарение на този модел, разработчиците не се ангажират с инсталация, поддръжка или управление на софтуера, тъй като посочените дейности се извършват от доставчика на услугата.

При внедряването на облачните модели, предприятията могат да изберат най-подходящата комбинация от ресурси и услуги спрямо специфичните си нужди и бизнес цели. Изборът между публични, частни и хибридни облаци, комбиниран с моделите IaaS, PaaS и SaaS, може да подобри капацитета за обработка на данни, да повиши сигурността и да намали техническите разходи. Всеки от моделите осигурява различно ниво на контрол и сложност на управление, което ги прави подходящи за различни организационни изисквания и възможности, представени на фиг. 1.4.

[9] ISO/IEC 27001 е международен стандарт за управление на информационната сигурност. Стандартът описва изискванията за създаване, прилагане, поддържане и непрекъснато подобряване на система за управление на информационната сигурност.



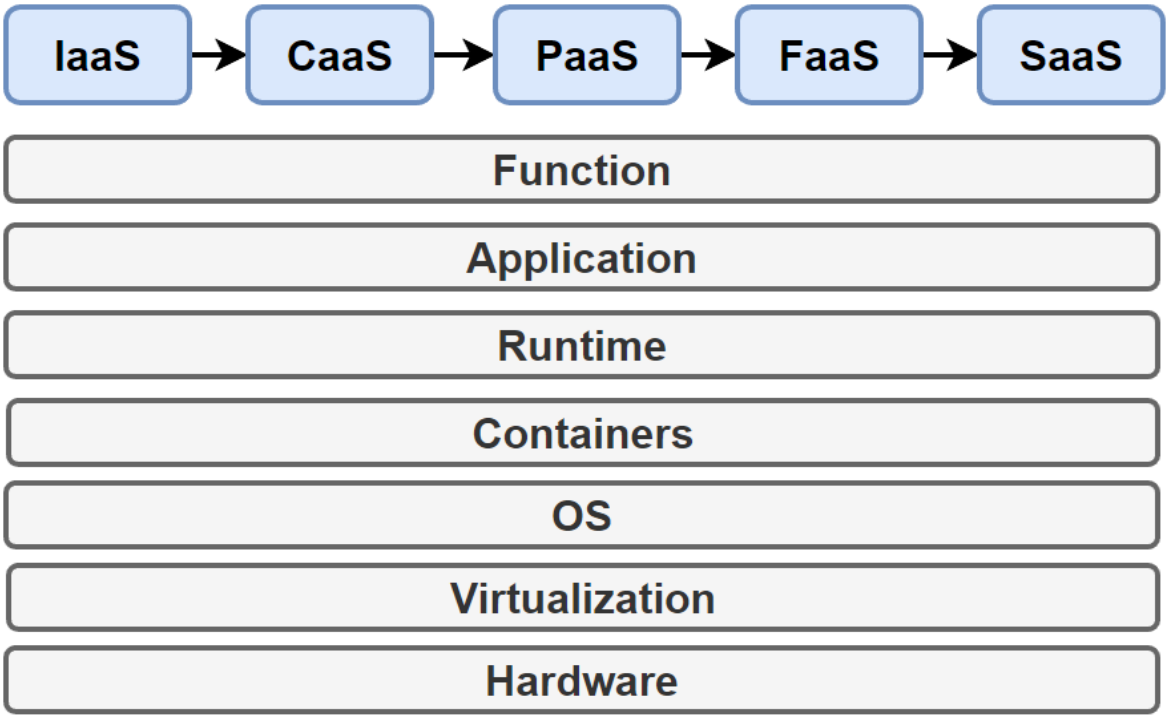
Фиг. 1.4. Сравнение между моделите на облачни услуги (IaaS, PaaS, SaaS) и традиционната локална инфраструктура, определяйки отговорностите по управление

Източник: Mohammed & Zeebaree, 2021.

На фиг. 1.4 се представя сравнение между различните модели на облачни изчислителни услуги (IaaS, PaaS, SaaS) и традиционната локална инфраструктура. Определят се отговорностите за управление и представя както оперативните, така и икономическите ползи. Fields et al. (2009) посочват, че IaaS, PaaS и SaaS предлагат по-ефективно разпределение на ресурсите спрямо традиционната локална инфраструктура, тъй като работят върху модел на ценообразуване, базиран на потреблението. Този подход дава възможност на организациите да плащат само за ресурсите и лицензите за софтуер, които действително използват, подобрявайки предвидимостта и управлението на разходите. Като основен недостатък, който някои автори считат за съществен е, че при преминаването от локални центрове към IaaS, PaaS или SaaS, компаниите могат да се сблъскат с проблеми, свързани с контрола върху данните, регулаторното съответствие и необходимостта от експерти с конкретни умения за управление на облачни услуги (Kumar &

Agnihotri, 2021). Облачните услуги за възстановяване „след бедствия“, описани от Guo (2013), прилагат защитна мрежа, която дава възможност за възстановяване на операциите при загуба на данни или системни повреди. За справянето с този проблем при локалните услуги компаниите трябва да поддържат дублирани хардуерни и софтуерни среди, които може никога да не бъдат използвани.

Различните видове облачни услуги подсигуряват различни нива на абстракция на компютърната инфраструктура. Същевременно дават възможност за надграждане чрез междинни модели от типа „Containers as a Service“ (CaaS) и „Function as a Service“ (FaaS).



Фиг. 1.5. Слоеве на облачна абстракция при основни и междинни модели

Източник: Garverick и McIver, 2023; Likness & Phillip, 2024.

Описани от Garverick и McIver (2023) като „изчисленията без сървър“ (от английски – serverles), CaaS и FaaS представляват друг вид облачни услуги, при който отделни функции се изпълняват в отговор на конкретни събития. Този модел допълнително намалява необходимостта от управление на ресурси, предоставя икономически ефективна опция за приложения, които

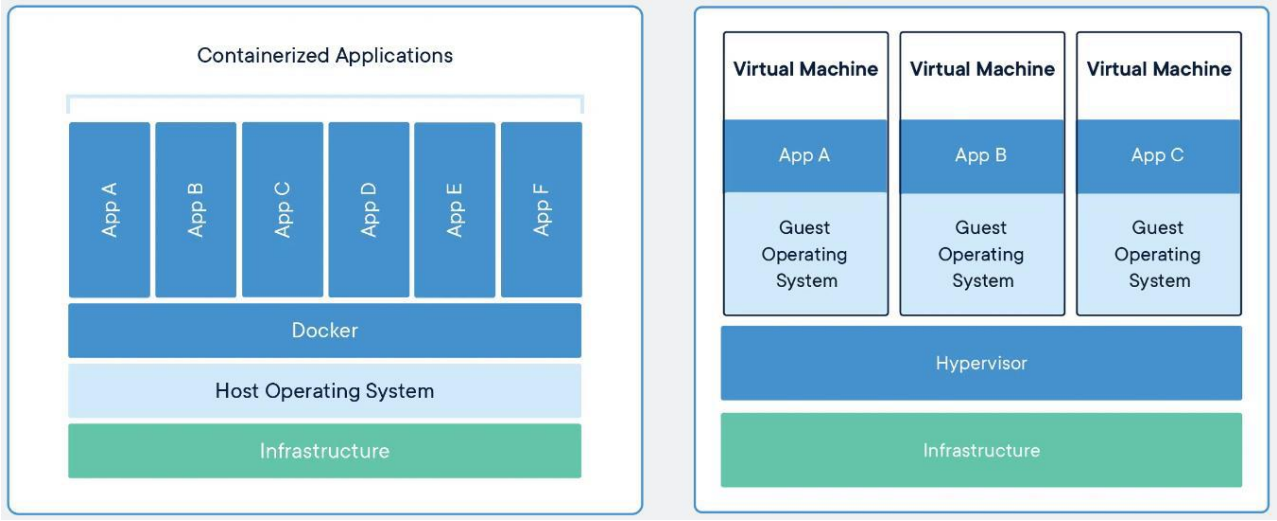
се нуждаят от специфична функционалност. Както е показано на фиг. 1.5, CaaS използва услуги за разработка, управление и изпълнение на контейнери (Likness & Phillip, 2024). FaaS е модел за облачни услуги, който дава възможност на потребителите да изпълняват функции (малки части от програмен код), без да управляват сървъри или инфраструктура.

За изграждане, доставка и изпълнение на облачни системи, експертите в областта препоръчват използването на контейнеризирани технологии (Lano & Tehrani, 2023; Toub, 2024). Контейнеризацията представлява подход в сферата на разработката на софтуер, при който кодът на приложението, всички негови зависимости и конфигурации са пакетирани в двоичен файл, наречен изображение. Според документацията, изображенията се съхраняват в регистър, който функционира като хранилище или библиотека (Garg, 2019). Облачната платформа трансформира изображението в работеща услуга (контейнер), който може да бъде стартиран, спрян, преместен или премахнат.

За различните части на едно приложение се създават отделни контейнери: клиентско приложение, уеб услуга и база от данни. Софтуерните контейнери се възприемат като стандартна единица за внедряване на софтуер, която може да съдържа различен код и зависимости. Контейнеризацията на софтуера дава възможност на разработчици и ИТ специалисти да прилагат промени в различни среди (продукционни и не-продукционни – Dev, Stage, Prod).

Контейнерите изолират приложенията едно от друго в споделена операционна система. Поради тази причина контейнерите предлагат предимства на изолация, преносимост, гъвкавост и контрол в целия жизнен цикъл на приложението. Според експерти в областта, най-използваната технология е Docker (Soper, Addie & Dembovsk, 2024), която представлява проект с отворен код за автоматизиране на внедряването на приложения като преносими, „самодостатъчни контейнери“, които работят еднакво както локално така и в облак. Docker контейнерите могат да работят върху Linux или Windows. На фиг. 1.6 е представено сравнение между компонентите на

виртуална машина и Docker контейнер.



Фиг. 1.6. Сравнение между компонентите на традиционна виртуална машина и Docker контейнер машина

Източник: Soper, Addie & Dembovsk, 2024.

Docker контейнерите включват приложението и всички негови зависимости. Споделят ядрото на операционната система с други контейнери и функционират като изолирани процеси. Изключение правят Hyper-V контейнерите, при които всеки контейнер работи в отделна виртуална машина. Контейнерите са основен инструмент в облачния софтуер. Тяхното управление се извършва чрез специализиран софтуер, наречен „*оркестратор*“.

Един от най-популярните оркестратори на контейнери е Kubernetes, който поддържа набор от функции за автоматизация на внедряването и управлението на контейнеризирани приложения (виж приложение 1). Kubernetes способства за високо ниво на автоматизация и оперативно управление на облачните приложения чрез програмния им код. Той работи с инструкции, които се изпълняват върху групи от виртуални машини, на които се разполагат и самите приложения.

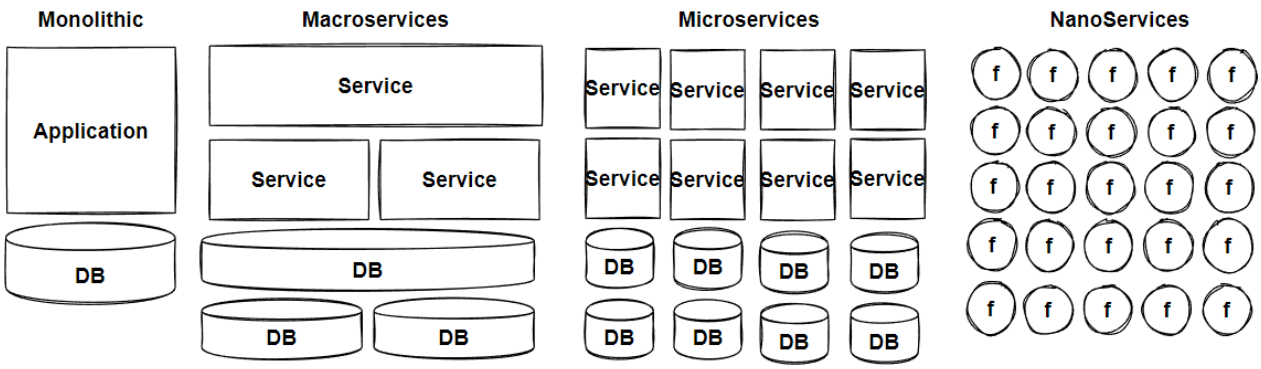
Проучвания на източници в областта (Li et al. 2021) показват, че за конструиране на облачни системи е подходящ ориентирания към

микроуслуги архитектурен стил (от английски – microservices). Микроуслугите осигуряват механизъм за взаимодействие между приложения, за разлика от уеб сайтовете, които са ориентирани към комуникация с потребителя и се използват чрез браузър (Nacheva, 2020). Микроуслугите представляват метод за разработване на сървърни приложения чрез използване на множество малки под услуги. Съответно, клиентите на сървърните услуги могат да бъдат отделни приложения, които да се поддържат и управляват самостоятелно.

За разлика, монолитните приложения, представляващи традиционен модел на софтуерна архитектура, при който всички компоненти на приложението са тясно интегрирани и разгърнати като едно цяло. Тази архитектура, преобладаваща в разработката на софтуер от много години, обхваща унифициран модел, при който различни функции (въвеждане на данни, обработка и потребителски интерфейс) са интегрирани в една програма. Монолитните програми показват висока степен на вътрешно свързване и взаимозависимост между компонентите, което предполага нарастваща бизнес (или домейн) сложност с течение на времето. Нарастването на бизнес сложността води до неструктурирана и трудна за поддръжка база от програмен код. Промените или модификациите, направени в една част от системата, могат неочаквано да повлияят и на други, несвързани компоненти. Следователно отстраняването на грешки се усложнява, което възпрепятства въвеждането на нови функционалности. Това поведение е описано в различни казуси за разработка на софтуер (Elgheriani & Ahme 2022), където се откроява недостатъкът на монолитните архитектури в сравнение с подхода на микроуслугите. Целта на микроуслугите е да се достави функционален продукт, изискващ постоянна поддръжка и връзка с клиента.

Фигура 1.7 показва развитието на софтуерните архитектури стилове за разработка и внедряване на облачни услуги, преминавайки от монолитна структура към микроуслуги, като се проследява постепенно разделяне на

модулността и автономността.



Фиг. 1.7. Развитие между различни софтуерни архитектурни стилове за разработка и внедряване на облачни услуги

Източник: Novais et al., 2019.

Изследователи в областта (Laszewski et al. 2018) анализират редица фактори и разработват методология, наречена „дванадесет фактора“ (Twelve-Factor), приложена в една от първите облачни платформи – Heroku. Тази методология дава набор от принципи и практики, към които разработчиците да се придържат, когато създават приложения, оптимизирани за съвременни облачни среди (табл. 1.5). Според практици (Grafiati, 2022), дванадесет факторната методология служи за основа при изграждането на облачни системи, тъй като може да се приложи към всякакъв тип уеб, настолни или мобилни приложения.

Таблица 1.5

Обобщение на методологията на дванадесетте фактора

Фактор	Описание
Code Base	Единна база, в която се съхранява изходният код на всяко софтуерно приложение, разположено в отделно хранилище (GitHub, GitLab, Azure DevOps). Благодарение на контрола на версиите, всяко приложение може да бъде внедрено в различни среди (Dev, Staging, Production).
Dependencies	Всяка микроуслуга изолира и пакетира свои собствени зависимости, като обхваща промени, които да не засягат цялата система.

Фактор	Описание
Configurations	Конфигурационната информация се управлява чрез инструмент, извън кода на микроуслугата. Тя може да бъде различна за различните страни.
Backing Services	Допълнителните ресурси (бази от данни, услуги за кеширане, брокери на съобщения) трябва да бъдат достъпни през уникален URL. Това разделение между ресурс и приложение дава възможност те да се заменят при нужда.
Build, Release, Run	Всяка нова версия следва да премине през няколко етапа на изграждане и изпълнение чрез използване на технологии за автоматизация. Резултатът от това е минимизиране на възможностите за допускане на човешки грешки и стандартизиране на цялостния процес.
Processes	Всяка облачна услуга трябва да се изпълнява в свой собствен процес, изолиран от другите.
Port Binding	Всяка услуга трябва да бъде самостоятелна, да има собствени интерфейси и да бъде конфигурирана да работи през определен порт.
Concurrency	Когато капацитетът на услуга трябва да се увеличи, мащабирането следва да бъде от хоризонтален тип, ориентирано към увеличение на процеси.
Disposability	Екземплярите на услугите трябва да благоприятстват бързото стартиране, както и изключване. Контейнерите, заедно с приложение оркестратор, по своята същност отговарят на това изискване.
Dev/Prod Parity	Различните среди е необходимо да се поддържат възможно най-сходни през целия жизнен цикъл на приложението. Тук контейнеризацията може значително да допринесе чрез насърчаването на същата среда за изпълнение.
Logging	Лог файловете, генерирани от различните услуги, следва да се третират като потоци от информация. За публикуване и архивиране на данни могат да се използват инструменти за управление на логове (например Azure Monitor или Splunk).
Admin Processes	Изпълняване на административни задачи, като почистване на вътрешни данни или рестартиране на услуга.

Адаптация по: Laszewski et al. 2018.

Микроуслугите са свързани с **фактор #6** от принципите на методологията на дванадесетте фактора, който свързва притежанието на

всяка услуга със своя собствена логика и данни в рамките на автономен жизнен цикъл. Концептуалните модели, технологиите и проблемите се различават между подсистемите или микроуслугите. Този принцип е заложен в дизайна, управляван от домейн, където всяка услуга притежава свой модел на домейн (данни + логика и поведение). Hoffman (2016) разглежда подробно всеки от първоначалните 12 фактора и добавя три нови, в съответствие със съвременния дизайн на облачните приложения към определен момент от време (табл. 1.6).

Таблица 1.6

Допълнение на методологията на дванадесетте фактора

Фактор	Описание
API First	Всеки ресурс трябва да бъде разгледан като приложно-програмен интерфейс, който да бъде интегриран към основната система.
Telemetry	Дизайнът на системата трябва да включва събирането на специфични за домейна данни, както и за текущото състояние на системата.
Authentication/ Authorization	Удостоверяването се използва за проверка на самоличността на потребителя, като обикновено се прилагат идентификационни данни (потребителско име и парола). Упълномощаването, от своя страна, определя нивото на достъп и привилегии, които получава вече удостоверения обект.

Адаптация по: Hoffman 2016.

В допълнение, на по-късен етап Karthikeyan (2021) предлага набор от ръководни принципи, които да се използват за подобряване на качеството на работното натоварване, което е показано в табл. 1.7 и представлява „петте стълба“ на т.нар. „добра облачна архитектура“.

Таблица 1.7

Добри практики на облачната индустрия

Фактор	Описание
Управление на разходите	Обхваща процеса на планиране, оценка, бюджетиране и контрол на разходите, целящ завършване на проект в рамките на одобрен бюджет. Ефективните стратегии за управление на разходите помагат на организациите да оптимизират използването на облачни ресурси като намаляват ненужните разходи.
Оперативно съвършенство	Автоматизиране на работната среда и операциите, за да се увеличи общата производителност и да се намалят човешките грешки.
Ефективност	Отговаряне на изискванията, поставени върху работни натоварвания, чрез тестове за производителност и натоварване, за да се идентифицират потенциалните затруднения.
Надеждност	Отнася се до концепцията за висока производителност, разгледана в тази глава, както и до функционалностите на мобилните и уеб приложения да „предвиждат“ и справят с неочаквани проблеми.
Сигурност	Тъй като облачните архитектури по своята същност разпределят ресурси между множество локации, съществува риск от различни заплахи за сигурността – от изтичане на данни до неоторизиран достъп. Въпреки че протоколите за криптиране и управление на самоличността укрепват защитата в облачна среда, променливият характер на киберзаплахите изисква постоянна бдителност. Алгоритмите за откриване на аномалии подпомагат ранното разпознаване и ограничаване на възможни нарушения.

Адаптация по: Karthikeyan, 2021.

1.4. Управление на бизнес процесите чрез ориентиран към домейн дизайн

При изграждането на дистрибутирана облачна система могат да възникват редица проблеми, свързани с комуникацията между отделните услуги, обработката на информация, бързодействието, бизнес логиката и

технологичното развитие. При въвеждането на комплексна бизнес логика за управление на клиентски поръчки, разработчиците могат да се сблъскат със проблеми при изграждането на алгоритми и структури от данни, предназначени за дефиниране на правила и осъществяване на валидации. Изхождайки от разгледаните в раздели 1.1 и 1.2 аспекти, може да се заключи, че управлението на поръчките се характеризира с относително висока степен на сложност, при което е подходящо да се използват утвърдени подходи разработка на софтуер, като например описания от Evans (2004) подход за разработка на софтуер, наречен „*ориентиран към домейн дизайн*“ (от английски: Domain-Driven Design – DDD). При този подход, структурата и езикът на имената, методите и променливите на класовете в програмния код на системата съответстват на бизнес изискванията.

Основните цели на DDD са:

• Поставяне на основния фокус на проекта върху неговата бизнес логика;

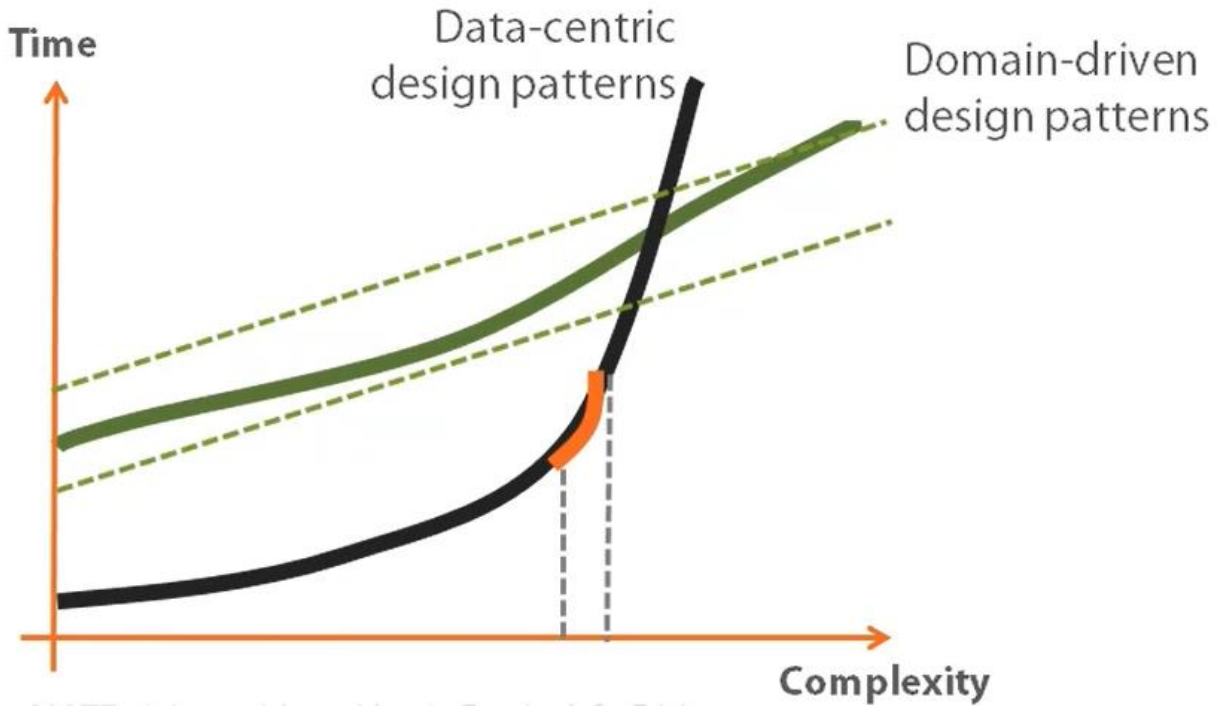
• Постигане на максимална адаптивност на софтуерната архитектура към бизнес целите и специфичните изисквания;

• Разработчици на информационна система и експерти в областта на SCM следва да сътрудничат с цел да създадат концептуален модел, който обхваща определени проблеми и функционалности.

De La Torre (2017) описва DDD като структуриран подход за проектиране на софтуер, който дава възможност за точно дефиниране на различните области в системата и улеснява разпределението на бизнес логиката в отделни, независими модули. Това помага за постигането на висока степен на модулност и повторна използваемост на микроуслугите. DDD насърчава използването на обща терминология, което подобрява комуникацията между различните екипи и участници в проекта, намалявайки риска от недоразумения и грешки (Millett & Tune, 2015). Този подход е подходящ за системи, подложени на чести промени, тъй като снабдява основа, която може да бъде адаптирана и разширена.

Друг подход за разработка на софтуер е „*дизайнът, управляван от данни*“, описан от Erl (2007). При този подход разделението между модулите и услугите се осъществява въз основа на данните, с които функционалностите оперират. Обикновено този подход започва с моделирането на базата от данни. Дизайнът, управляван от данни, е особено подходящ за проекти, при които данните заемат водеща роля. Например приложения, изпълняващи основни операции: създаване, четене, актуализиране и изтриване (от английски: Create, Read, Update and Delete – CRUD). Недостатъкът на този подход е, че има ограничена способност да имплементира бизнес логика, която интегрира различни технически процеси (Vernon, 2016).

Fowler (2019) сравнява DDD с дизайна, управляван от данни, с оглед на необходимото време и сложността при софтуерната разработка. Резултатът от това сравнение е онагледен на фиг. 1.8.



Фиг. 1.8. Сравнение на ориентиран към домейн дизайн с дизайн, управляван от данни, в контекста на времето и сложността при разработка на софтуер

Източник: Fowler, 2019.

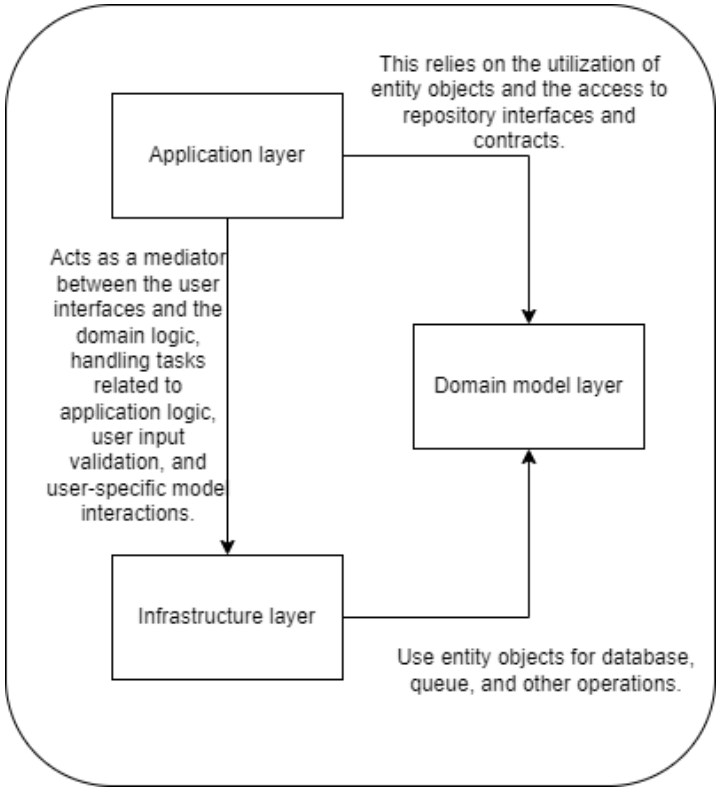
В тази диаграма по оста Y са представени времето и разходите, докато по оста X е измерена сложността при разработка на софтуер. Може да се забележи, че при дизайн, управляван от данни, след достигане на определено ниво на сложност, дори незначително увеличение на сложността може да доведе до значително увеличение на разходите и времето, необходимо за разработка. За разлика от това при DDD времето и разходите за проекта имат тенденция да нарастват линейно. Обикновено началните разходи, свързани с внедряването на DDD, са по-високи в сравнение с дизайна, управляван от данни.

Според принципите на DDD (Zimarev, 2019) случаите на употреба следва да се моделират въз основа на начина, по който реалният бизнес функционира, като се има предвид, че всеки бизнес постоянно се развива във времето. Това допринася за поддържане на качеството на софтуерната архитектура, при което сложността на бизнес логиката представлява индикатор за сложността на проблемната област, която софтуерът е предназначен да управлява. Според нас, DDD подходът има ограничен обхват и не е предназначен да предлага решения за проблеми, свързани с облачната инфраструктура на проекта или защитата от хакерски атаки. Поради това DDD е подходящо да включва само процесите, свързани с комуникация между отделните услуги и ефективната обработка на информацията. Тъй като посочените процеси са от съществено значение за цялостното функциониране на системата, те са разгледани допълнително и в следващата глава на дисертацията.

Основна характеристика на DDD е улесняване на комуникацията между експертите по домейна и софтуерните инженери, като се използва общ, **универсален език** (UL). Чрез този инструмент, който помага за обединяване на дизайнери и програмисти, могат да се създадат модели на домейна и да се приложат в практиката. Когато програмния код е написан чрез UL, той може да даде индикатори за случаи и изисквания, които не са били достатъчно изяснени предварително. За да функционира успешно,

класовете в кода и таблиците в базата от данни трябва да се именуват в съответствие с термините от UL. Тази обща номенклатура улеснява разбирането и съгласуването на изискванията между всички участващи страни. Batista (2022) изтъква необходимостта от универсалния език за предотвратяване на „недоразумения“ и неправилни предположения. UL може да се използва в различни области при разработката на софтуер, включително в документацията, комуникацията между екипите, създаване кода на приложението и кода за тестване. UL може да се развива и поддържа с течение на времето, като се използва за средство за събиране и организиране на знанията и бизнес логиката (Rademacher et al. 2018).

На базата на направени проучвания (Braun et al. 2021; Khononov, 2021) голямата част от корпоративните системи се разделят вътрешно на различни слоеве, което подпомага управлението на програмния код. Следвайки принципите на DDD, програмните класове и обекти могат да се организират в няколко отделни слоя, както е показано на фиг. 1.9.



Фиг. 1.9. Трислоен архитектурен модел

Източник: Vettor & Smith, 2024.

Счита се, че в приложния слой се координира потокът на изпълнение между отделните програмни класове и обекти (Армянова, 2018). В него се определят случаите на използване и операциите, които се изпълняват в рамките на облачна услуга. В този слой се организира взаимодействието между потребителския интерфейс и основните елементи. Обикновено се реализира като уеб API или MVC проект (Сълов, 2022). Приложният слой е зависим от домейн слоя и инфраструктурния слой.

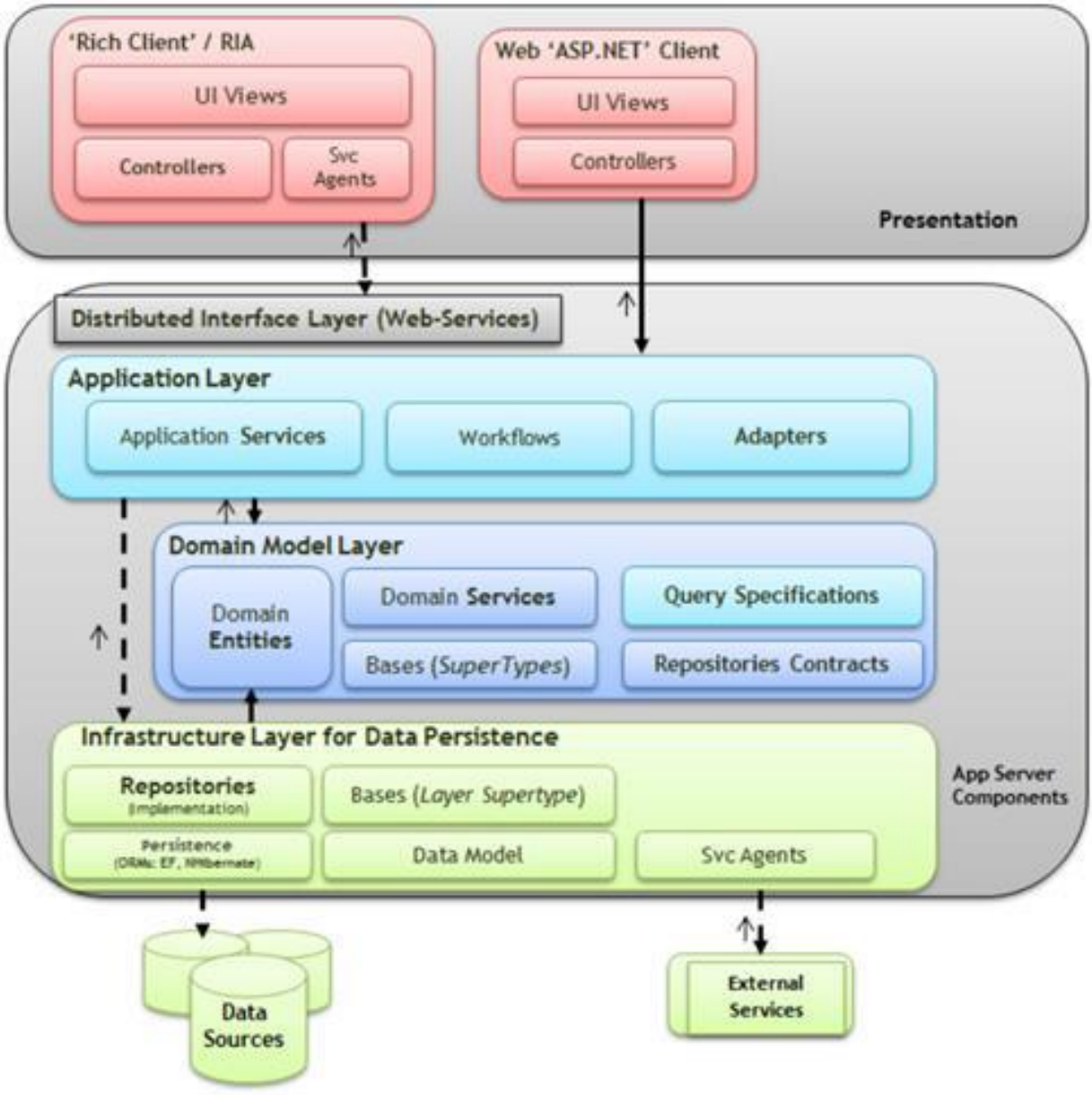
В домейн слоя се решават бизнес проблеми, свързани със съхранение на данни, обработка на съобщения, мрежова комуникация и интеграция с външни услуги. От гледна точка на програмния код, този слой съдържа напълно изолирани обекти, за да няма зависимост към други компоненти. Инфраструктурният слой е зависим от обектите на домейн слоя. Програмният код в инфраструктурния слой включва конфигурации, свързани с физическото и логическото управление на данните. В този слой се намират и класове, отговарящи за мониторинга на приложенията, както и за управление на транзакциите. По този начин инфраструктурният слой действа като посредник между домейн логиката и специфични технологични програми.

Грег Йънг представя концепцията за разделяне на отговорността за команди и заявки (Command and Query Responsibility Segregation – CQRS) през 2010 г. като разширение на принципите на DDD. Тази идея се базира на принципа на Meyer (2009), наречен „разделяне на команди и заявки“ (Command and Query Separation – CQS). Съгласно CQRS принципа всеки метод в API трябва да бъде или команда (command), или заявка (query), но не и двете едновременно. Според Йънг командите са методи, които извършват операции, променящи състоянието на системата[10]. Те са отговорни за изпълнение на действия, които променят данни или файлове. Заявките се

[10] Изразът „промяна състоянието на системата“ се отнася до процесите и събитията, които предизвикват изменения в текущото състояние на различните компоненти или обекти в рамките на системата или базата от данни. Тези изменения могат да включват промени в данни, статуси или състояния на потребителските акаунти, поръчки и доставки.

използват за извличане на информация, но без да ги променят.

Фиг. 1.10 представя диаграма, която надгражда трислойния архитектурен модел от фиг. 1.9, добавяйки CQRS.



Фиг. 1.10. Надграждане на трислойния архитектурен модел

Източник: Vettor, Molenkamp & van Wijk, 2024.

Една от съществените страни на CQS е, че методите би трябвало да връщат стойност, само ако са „референтно прозрачни“ и нямат „странични ефекти“, което прави кода по-четлив и предсказуем (Indrasiri & Suhothayan, 2021). Въпреки това, не винаги е възможно или практично да се спазват

напълно принципите на CQS. Има случаи, когато методите трябва да имат едновременно страничен ефект (промяна на състоянието) и да връщат стойност. Например, при работа със структура от данни „стек“, методът „pop“ премахва и връща последния елемент от структурата (Armiyanova, 2017). В този случай разделянето на действията в два отделни метода може да бъде нелогично. Поради това следва внимателно да се оценят конкретните изисквания на дадено приложение, преди да се приложи подходът на CQS. Подходът на CQRS прилага принципи, подобни на CQS, но акцентира върху крайните точки на определена облачна услуга. Основните операции се разделят на две: едните за управление на записите (командите), а другите – за обработка на заявките (четенето). Считаме, че чрез това разделение могат да се разработят различни стратегии, които да се фокусират върху конкретните нужди на облачната информационна система.

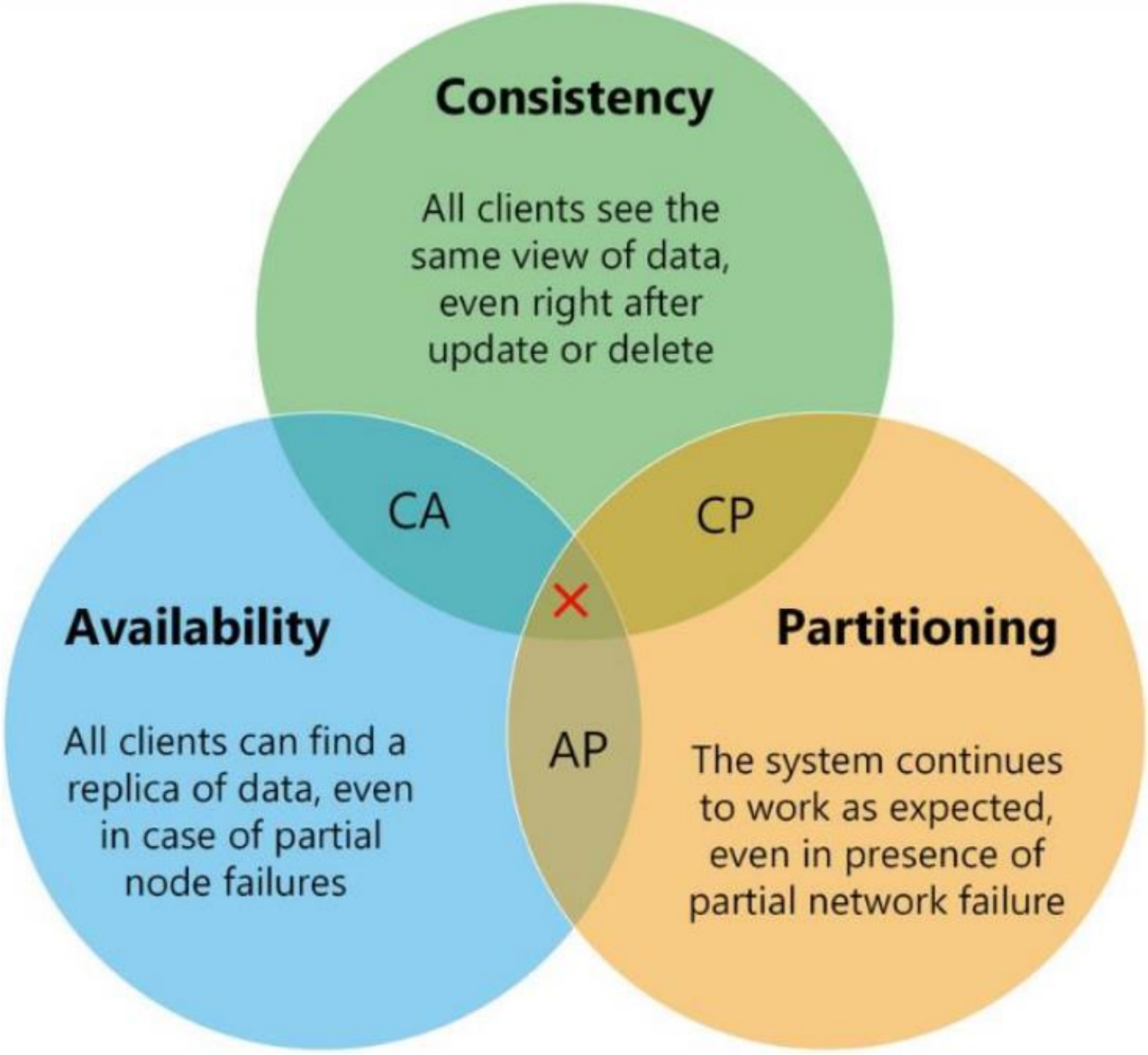
Приложният слой преобразува входните заявки и команди и ги изпраща по споделен комуникационен канал, известен като „манипулатор на съобщения“. Чрез него, командите се използват, за да инструктират приложението да изпълни определено действие, заявките се прилагат, за да се поиска информация или данни от приложението, а информационните съобщения от двата типа са регистрирани като „*събития*“. Командите активират процеси в домейн моделите на системата, докато събитията са резултат от тези процеси. Именуването на операциите следва стандартизирани указания на UL.

Според Brewer (2012) теоремата CAP (известна още като теоремата на Брюър) е основен принцип в областта на архитектурата на микроуслуги и има тясна връзка с CQRS. Според теоремата CAP, една разпределена система не може да осигури едновременното наличие на всичките три свойства:

1. **Консистентност (Consistency):** всички клиенти виждат един и същ изглед на данните, дори след актуализация или изтриване;
2. **Наличност (Availability):** всички клиенти могат да намерят реплика на данните, дори при частични неизправности в

микроуслугите;

3. **Разделяне (Partitioning):** системата продължава да работи нормално, дори при частични проблеми в мрежата от микроуслуги.



Фиг. 1.11. Диаграма, представяща теоремата CAP

CA: Консистентност + Наличност

CP: Консистентност + Разделяне

AP: Наличност + Разделяне

Източник: Brewer, 2012.

Считаме, че чрез внедряването на CQRS разработчиците могат да създават облачни услуги, които обработват големи натоварвания от HTTP

заявки и същевременно поддържат съгласуваност на данните чрез обработка на команди. CQRS е междинен етап между DDD и подходът за „източник на събития“ (Event Sourcing – ES). Извличането на събития допълва CQRS, тъй като всички промени в състоянието на системата се записват последователно и могат да бъдат използвани за съгласуване и анализ на данните.

В заключение, DDD и CQRS все по-често се прилагат при проектиране и изграждане на облачни услуги. Капсулирането на основния бизнес домейн в предварително дефинирани модули помага за правилното създаване на подсистеми и обекти. Чрез комбинирането на двата подхода производствените предприятия могат да изградят системи, които са не само технически стабилни, но и съобразени с бизнес целите и изискванията. Прилагането на DDD и облачни архитектури следва да помогне на организациите да внедряват иновации, да предоставят услуги на своите клиенти и да бъдат конкурентоспособни в бързо променящата се верига на доставки.

Изводи и обобщения към първа глава

В настоящата глава са изследвани основни проблеми, свързани с информационното осигуряване при управлението на поръчки от бизнес клиенти. Представени са теоретични основи и дефиниции от различни автори, които разкриват ролята и значението на веригите на доставки в производствените предприятия. Проучени са разлики между права и обратна верига на доставки, основни компоненти на стратегията за доставка на продукция, както и взаимосвързаността между основните компоненти. Установено е, че информационните потоци между компонентите на веригата на доставки се поддържат от набор от корпоративни софтуерни подсистеми, включително ERP, CRM, TMS, както и подсистеми за управление на склада, качеството и жизнения цикъл. Основна е ERP, която дава възможност за интеграция на останалите подсистеми. Разгледани са характеристики на SAP, една от водещите ERP системи, в това число: организационни единици,

основните и трансакционните данни в модулите за управление на продажби и дистрибуция.

Въз основа анализа на научни доклади и статии, е представен технологичен модел, представящ различни варианти за приемане и управление на клиентски поръчки в производствено предприятие, както и модел на централизирана система, интегрираща различни видове външни и вътрешни подсистеми. Основната цел на модела е подобряване на достъпа до актуална информация и надграждане на текущите ERP, CRM, TMS подсистеми на предприятието. Приемаме че, интегрирането на посочените подсистеми е от съществено значение за успешната дигитализация на бизнес процесите.

В допълнение са представени основните характеристики на облачните технологии, които дават възможност за управление на компютърни ресурси чрез различни видове облачни услуги – публични, частни и хибридни, както и модели като IaaS, PaaS и SaaS. Мобилните и уеб приложения, използващи облачни технологии, следва да бъдат разработени чрез контейнеризация и микроуслуги, за да осигурят висока производителност. За адаптиране на сложната бизнес логика и изискванията в програмния код на облачната информационна система са проучени принципите и практиките на DDD. Те помагат за създаването на отделни, независими модули, които отразяват реалните бизнес процеси и улесняват управлението и развитието на системата. Чрез използването на UL се подобрява комуникацията между експертите в областта на SCM и разработчиците на информационната система. Теоретичната рамка, представена в първа глава, реализира първата изследователска задача и е съществен принос на настоящия дисертационен труд.

Резултатите от литературния обзор разкриват редица проблеми, свързани с управлението на поръчките в реално време. Сред тях са нуждата от адаптиране към променящите се процеси във веригата на доставки, ограничената гъвкавост на традиционните ERP и SCM подсистеми,

потенциалните затруднения при миграцията на съществуващи подсистеми, както и проблеми с интеграцията на външни партньори и използването на IoT технологии. Освен това са разграничени:

- рискове от кибератаки при директно разкриване на интерфейси към вътрешните подсистеми;
- необходими подобрения в клиентското обслужване;
- подобрения в комуникацията и обмена на информация.

Въз основа на проучените технологии и практики е формулирана теза за оптимизиране на процесите по управление на поръчките чрез персонализирана информационна система, конфигурирана съобразно нуждите на конкретното предприятие. За реализацията на такава система е избрано използването на облачни услуги, които подсигуряват надеждност, сигурност и автоматизирани процеси по актуализации. Във втора глава се представя архитектура на облачна система за управление на поръчки от бизнес клиенти, изградена на базата на посочената персонализация. Демонстрирани са концептуалният модел на системата, нейните функционални компоненти и връзките помежду им.

Глава 2. Архитектура на облачна система за управление на поръчки от клиенти

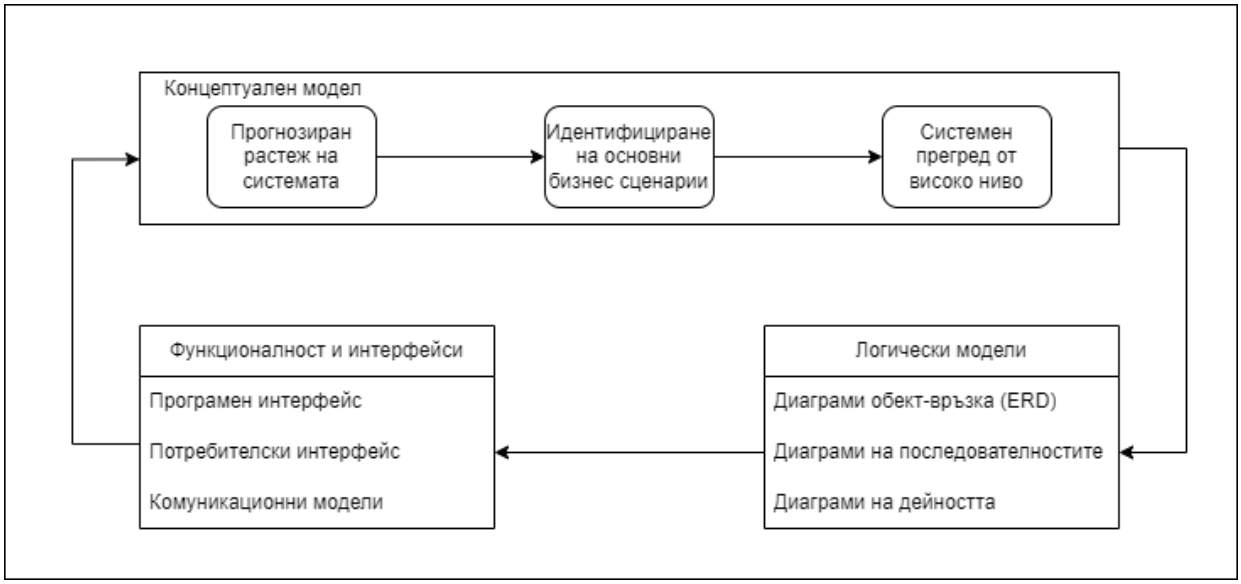
Въз основа на проведеното проучване, представено в предходната глава, в настоящата глава се разглежда архитектурата на облачна система за управление на поръчки. Изследват се концептуалният и логическият модел, които представят софтуерните елементи и интерфейси, изграждащи системата, както и комуникационният модел, ръководещ интеграцията на облачни услуги. Едновременно с това са представени случаи на употреба и бизнес сценарии, използвани за моделиране на приложения за обслужване на клиенти.

2.1. Концептуален модел на облачната система за управление на поръчките

Основните компоненти на персонализираната облачна система за управление на поръчките (ПОСУП), която надгражда функционалностите на съществуващите SCM системи и осигурява взаимодействие помежду им, са представени като част от концептуален модел на софтуерната архитектура. Този концептуален модел служи като основа за проектиране и внедряване на системата в производствено предприятие (Penchev, 2016).

За изграждането на концептуалния модел на системата е приложен итеративен процес (фиг. 2.1), предложен от Ingeno (2018). Итеративният процес представлява циклична поредица от етапи за разработване и усъвършенстване на софтуерната архитектура. Итеративният процес започва с концептуален модел, включващ прогноза за растежа на системата и дефиниране на основните бизнес сценарии, след което се извършва преглед от високо ниво. Вторият етап обхваща разработването на логически модели, включително ER диаграми (Entity Relationship Diagrams), диаграми на последователности и дейности, които да представят информационните

потоци, основните процеси и взаимодействията между компонентите в системата. Последният етап обхваща различни страни на взаимодействието на системата с крайните потребители, включително дефиниране на функционалности и интерфейси. Това дава възможност за определяне на очакваните резултати и възможности на системата.

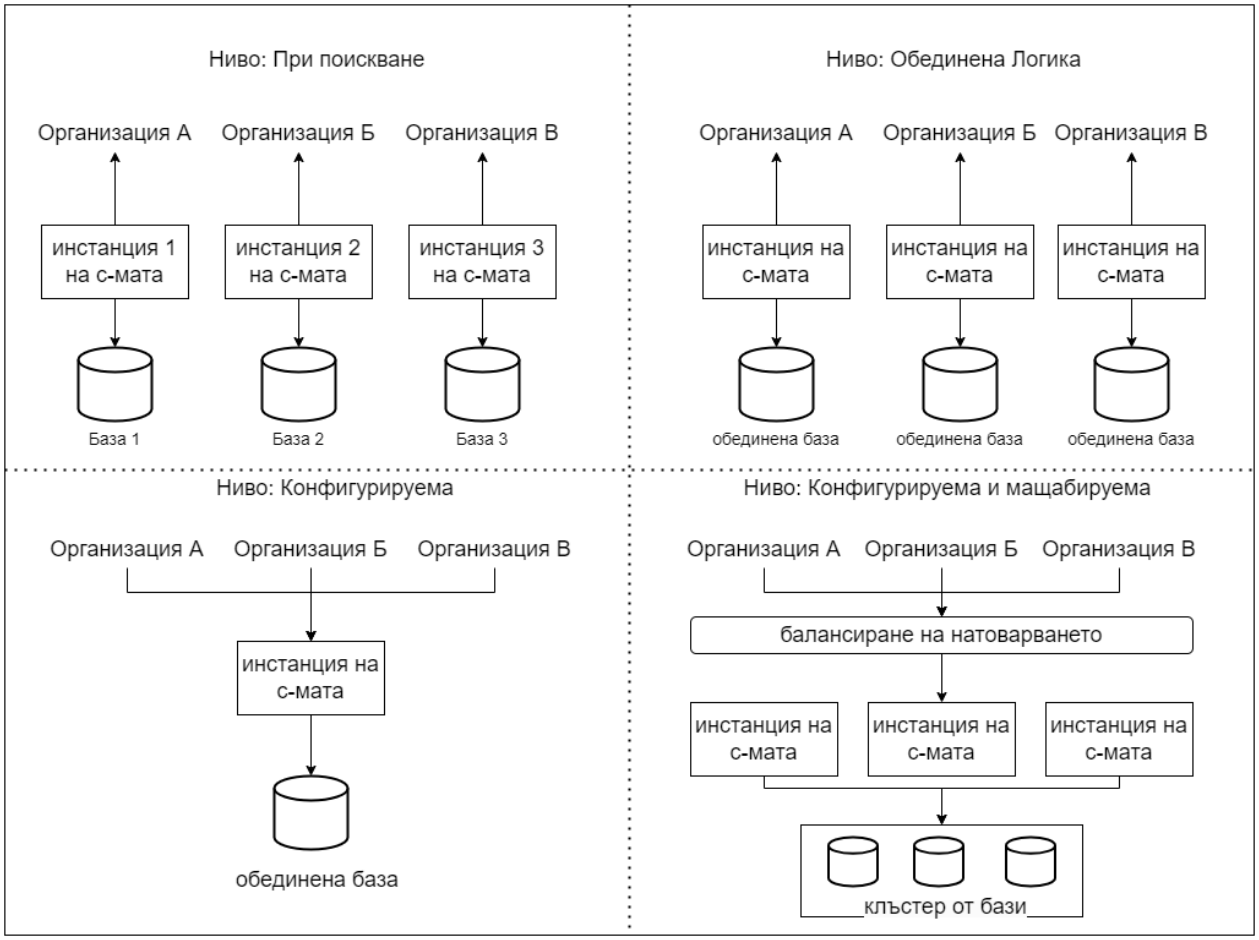


Фиг. 2.1. Итеративен процес за проектиране на концептуален модел

Адаптация по: Ingeno, 2018.

Според Stuckenberg (2014), за изграждането на архитектурата на облачна система за управление в съответствие с организационните единици, представени в глава първа, е от съществено значение да се представи модел на **прогнозиран растеж на системата**[11]. Моделът, представен на фигура 2.2, е адаптиран с цел да представи четири възможни нива, базирани на различни търговски организации, по подобие на ERP системи от класа на SAP S/4 HANA. Анализът на различните нива дава възможност на производственото предприятие да планира етапи за подобряване на съвместимостта с други корпоративни системи и бъдещи изисквания.

[11] Прогнозираният растеж се отнася до оценката или предвиждането за бъдещото развитие на дадена система. Тази прогноза може да е базирана на текущи данни, тенденции, анализ на външни и вътрешни фактори и проблеми, като например посочените в глава първа.



Фиг. 2.2. Модел на зрялост на архитектурата

Адаптация по: Stuckenberg, 2014.

В първото ниво, наречено „при поискване“, всяка организационна единица[12] използва специална версия на информационната система, която да бъде пригодена според специфичните изисквания на текущата организация на производственото предприятие. Данните за поръчки и доставки се записват в отделни бази от данни, а програмният код е различен за всяка отделна организация. Във второто ниво на „обединена логика“ програмният код се стандартизира. Всяка организационна единица използва идентични копия на софтуера, но адаптирани на специален хардуер. В тези два случая персонализирането е ограничено до предварително дефинирани функционалности, както и синхронизацията между отделните организации е

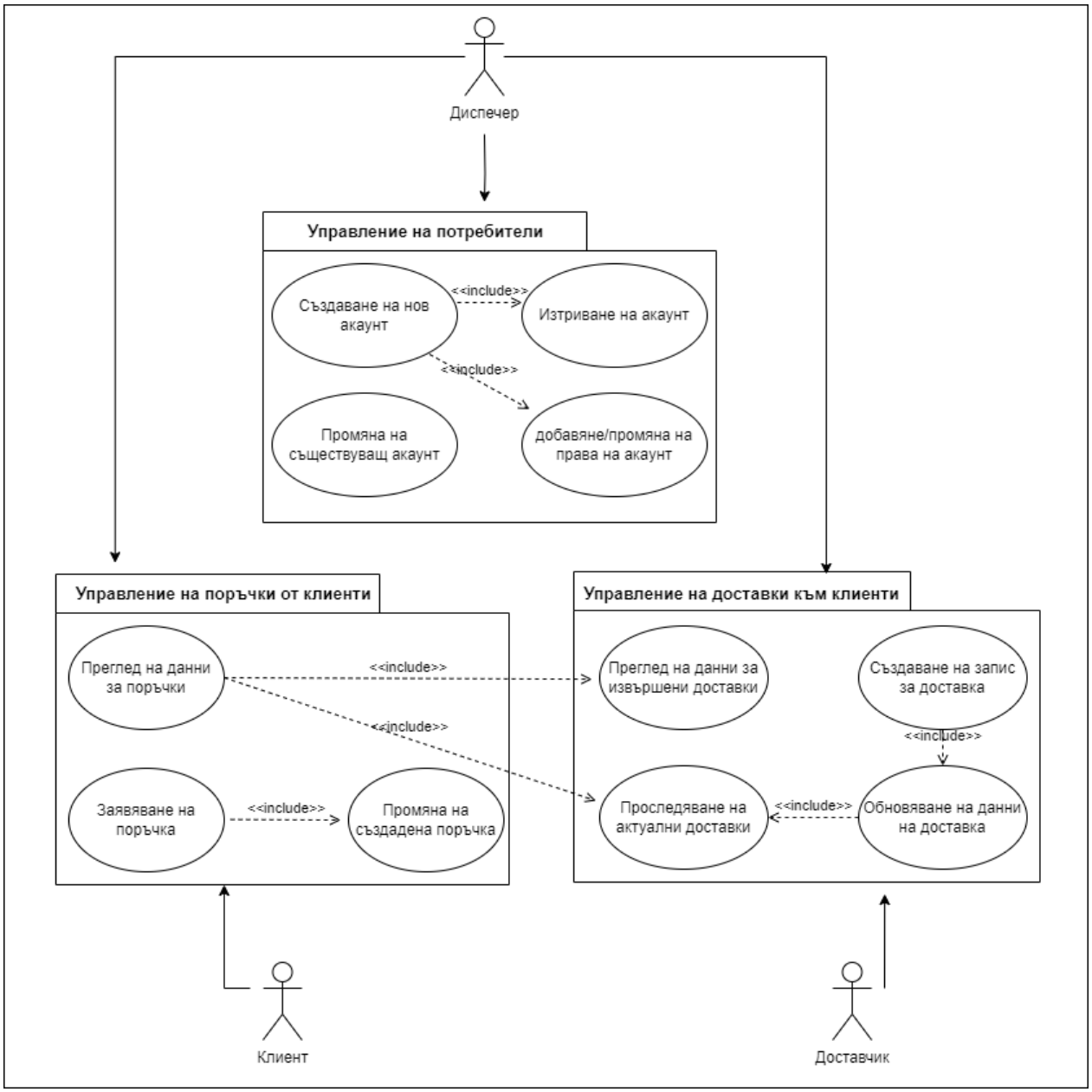
[12] Терминът „организационна единица” може да бъде дефинирана като подразделение или структурен компонент в рамките на една по-голяма корпорация (холдинг). В контекста на модула за продажби и дистрибуция на SAP се дефинира като „търговска организация”.

сложна за изпълнение. За да се преодолеят посочените проблеми, третото и четвъртото ниво предоставят по-висока степен на конфигурация, при което всеки клиент използва едно и също копие на един и същ хардуер. По този начин се дава възможност за постоянни обновления и реализиране на нови версии на системната. С оглед на това, четвъртото ниво е допълнително подобрено от балансьор на натоварването и клъстер от бази от данни.

Считаме, че архитектурата на облачната информационна система може да се базира на четвърто ниво на конфигурируемост и мащабируемост. Това дава възможност множество клиенти на различни търговски организации да използват една и съща версия на системата в обединена инфраструктура. По този начин оперативните разходи могат да бъдат разпределени между отделните организации на производственото предприятие. Четвърто ниво на конфигурируемост подпомага омниканалния подход и пренасочването на ресурси. За да се постигне оптимална производителност и ниско време за отговор (описани в глава първа), предложената архитектура включва механизми за кеширане и балансиране на натоварването. Кеширането намалява времето за достъп до често използвани данни, а балансирането на натоварването разпределя заявките между различни ресурси, за да се избегне претоварване. Във втория етап на итеративния процес се определя главния бизнес сценарий за използване на информационната система и се проучват различни функционални и нефункционални изисквания. Този етап е свързан и с интегрирането на система за управление на поръчки в оперативната рамка на производствено предприятие.

Бизнес сценариите описват основните възможности на системата: „управление на потребителски акаунти“, „управление на поръчки от клиенти“ и „управление на доставки до клиенти“. Те обхващат и участниците, които в нашия случай включват „диспечер“, „клиент“ и „доставчик“. Фиг. 2.3 представя UML диаграма на основен бизнес сценарий. Съобразена с потребителските изисквания, тази диаграма служи за визуална илюстрация на взаимодействията между основните роли, функции, обекти и

събития, които променят вътрешните състояния на данните в системата (Parusheva & Pencheva, 2021).



Фиг. 2.3. Диаграма на главен бизнес сценарий в ПОСУП

Разработка на автора

Диспечерът е представен като основен потребител. Той управлява оперативните задачи в системата. Достъпът на Диспечера до всички модули му дава възможност да създава, променя и изтрива акаунти, както и да задава или променя правата за достъп на другите потребители. Диспечерите имат достъп и до модула за управление на поръчки за продажби. Този модул

включва набор от функции: преглед на данни, заявяване на нови заявки и промяна на вече съществуващи поръчки. Основната му цел е осигуряване на взаимодействие с бизнес клиентите, които имат пряка връзка с този модул. Подобно на диспечерите, те могат да създават, обновяват или отменят заявки за поръчки.

За управлението на доставки, доставчикът има възможност да създава и обновява записи. По този начин информацията за доставките е актуална и може да бъде проследена в реално време, което е съществен фактор за своевременното и точно изпълнение. В същото време, диспечери и бизнес клиенти имат достъп до функционалности за преглед и проследяване на доставки. На диаграмата са показани някои случаи на употреба, при които са отразени зависимости между различни функции. Например, при активните поръчки е логично да се включат детайлите за определена поръчка и съответните доставки.

Според Василев (2015), ясно дефинираните изисквания са основата на успешния проект, тъй като включват набор от процеси: анализ, спецификация и валидиране. В този смисъл, във втория етап на итеративния процес е формулирането на **функционални** и **нефункционални изисквания**. Функционалните изисквания определят специфичното поведение и операциите на системата: автоматизация на обработката на поръчки, интеграция с планирането на ресурсите на предприятието в реално време и улесняване на взаимодействията при обслужването на клиенти. Нефункционалните изисквания определят оперативните атрибути и ограничения на системата, включително показатели за производителност, стандарти за сигурност, мащабируемост, надеждност и достъпност на потребителите. Функционалните изисквания описват характеристиките на продукта и са създадени да отговарят на променящите се нужди на служителите в производствените предприятия и техните клиенти. Основна част от изискванията е способността на системата да обединява различни бизнес функции: механизми за проследяване на доставки, координация

между отделните подразделения на предприятието и автоматизиране на административни задачи.

Изхождайки от анализа на литературата по управление на веригите за поръчки и доставки от първа глава, установихме, че основните изисквания към облачната система включват: регистриране и вписване на потребител, преглед на текущите поръчки, разглеждане на детайлите за определена поръчка и доставките към нея. Системата следва да поддържа събиране и актуализиране на данни в реално време от вътрешни и външни подсистеми като ERP и IoT. Допълнителните функционалности, които системата трябва да предлага, включват филтриране на елементи от потребителския интерфейс, създаване на нова поръчка или промяна на вече съществуваща, както и генериране на отчети и документи, включително електронно доказателство за доставка. Дава възможност за съставяне на план-график за доставките и предоставяне на възможности за пренасочване на ресурси.

Както бе споменато, нефункционалните изисквания определят оперативния капацитет на облачните услуги, докато функционалните изисквания описват какви действия изпълняват. Нефункционалните изисквания обхващат: надеждност, непрекъсната и коректна работа на компонентите на системата, мащабируемост, която дава възможност на системата да обслужва нарастващия брой потребители, сигурност, предпазваща чувствителните данни от неоторизиран достъп, както и висока производителност при обработка на голям брой HTTP заявки. Тези характеристики са от съществено значение за облачните системи, за да помогнат на производствените предприятия да постигнат прозрачност при операциите по управление на поръчките от клиенти. Нефункционалните изисквания са известни още като „атрибути за качество“ на системата.

Някои от нефункционалните изисквания към ПОСУП са следните:

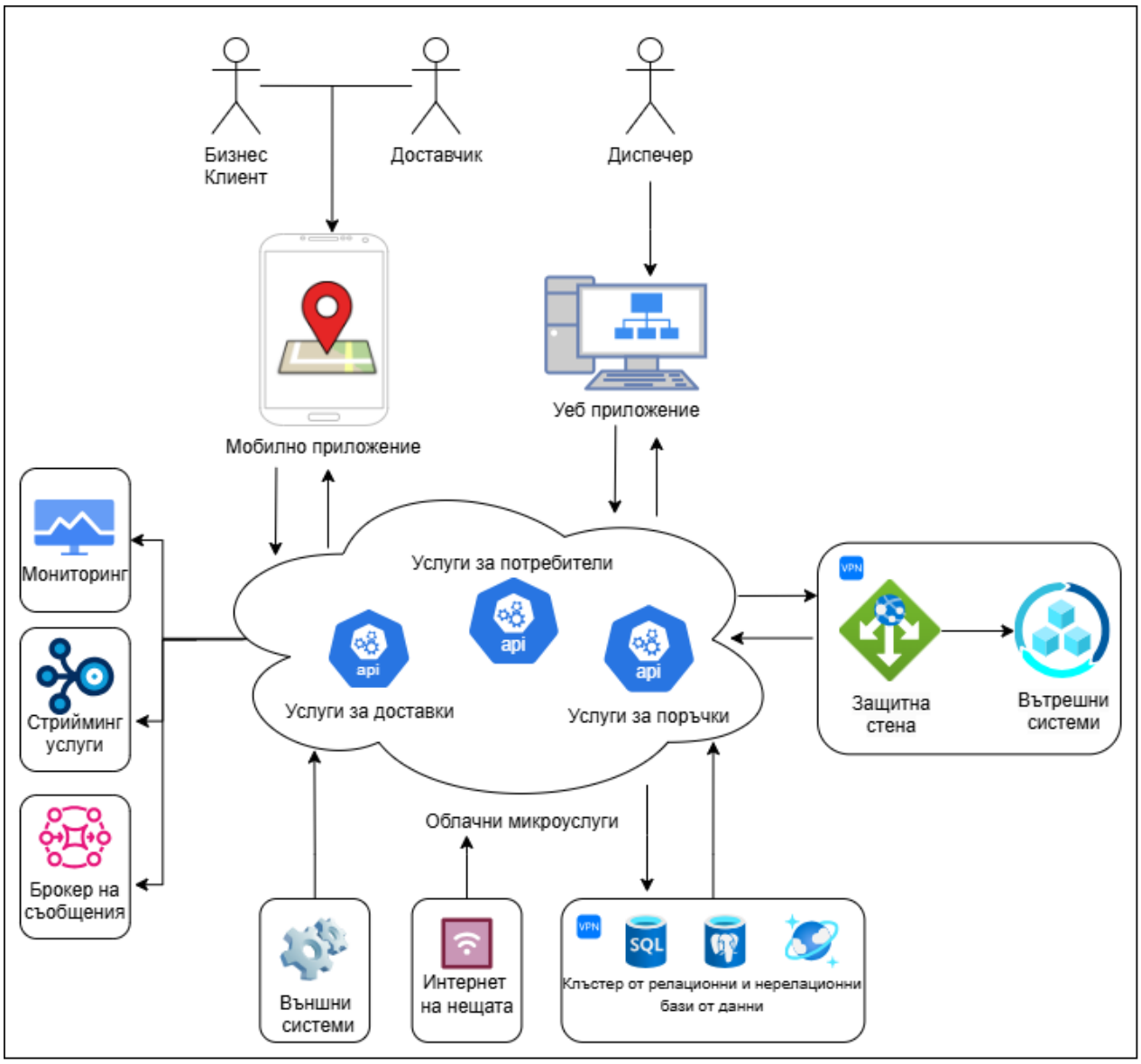
- Необходимо е системата да осигурява висока достъпност, като сама регулира разпределението на облачните ресурси в съответствие с нарастващия потребителски трафик. При последващо намаляване на трафика,

използваните ресурси трябва автоматично да се върнат към предишните нива;

- Да предоставя възможности за мониторинг и да води диагностични дневници, които подпомагат откриването и отстраняването на неизправности, както и на непредвидени проблеми по време на работа;
- Да интегрира т.нар. поддържащи услуги (от английски – backing services), като брокер на съобщения и стрийминг услуги;
- Необходимо е да се поддържат процеси на непрекъсната интеграция и внедряване (от английски – continuous integration and deployment);
- Системата следва да поддържа междуплатформен хостинг;
- Изисква се отговорът на системата да бъде върнат в рамките на секунди;
- Необходимо е да се спазват договорените в SLA показатели и предвидените санкции при неизпълнение;
- Да поддържа механизми за равномерно разпределяне на входящия мрежов трафик между множество сървъри и инстанции;
- Необходимо е в системата да са включени процедури за архивиране на данни;
- Необходимо е в системата да са включени инструменти за контрол на версиите и възможности за връщане към предишна версия;

Въз основа на анализираните бизнес сценарии, функционални и нефункционални изисквания, следва да представим концептуалния модел на ПОСУП.

Показан на фиг. 2.4, концептуалния модел от високо ниво обхваща клиентски приложения, облачни микроуслуги, вътрешни и външни корпоративни системи.



Фиг. 2.4. Концептуален модел от високо ниво на ПОСУП

Разработка на автора

Фиг. 2.4. представя концептуален модел, на който е показана структурата на приложенията в ПОСУП. Изследванията сочат, че мобилните приложения са подходящи за взаимодействие с крайните потребители, тъй като поддържат набор от функции: местоположение, камера и работят с уеб услуги. Клиентите на фирмата, които се явяват крайните потребители, управляват и проследяват поръчките и доставките в реално време с мобилно приложение. Предназначението на приложението е да помага в планирането и логистиката на работната площадка, да въздейства върху крайния резултат

с информация и данни. Информацията на мобилното устройство следва да е актуална. Текущото състояние на поръчка и местоположение на доставките да се проследяват на живо. Други възможности са преглед на история, създаване на нова, промяна или отказване на съществуваща поръчка. Приложението може да бъде публикувано в Google Play Store и Apple App Store. Двете платформи предлагат системи за оценяване и прегледи, с помощта на които се събира обратна връзка от крайните потребители и се подобряват функционалностите на приложението на база мненията и препоръките.

Уеб приложението е предназначено за диспечери и е част от интегрирана TMS подсистема. Посредством него могат да се създават заявки за поръчки и доставки, а генерираните данни се синхронизират с вътрешните подсистеми. Уеб приложението функционира под формата на инструмент за вземане на решения, предоставяйки предварителни варианти за работния график на доставчиците. Тези предложения могат да бъдат одобрени, отхвърлени или променени в зависимост от преценката на дежурния диспечер. Изхождайки от текущото състояние на превозните средства, подсистемите зад уеб приложението планират доставките според изискванията на клиентите. Те разчитат на точна и електронно удостоверена информация.

Обхватът на уеб приложението включва балансиране на работното натоварване на превозните средства, дава възможност за проследяване и коригиране както на поръчките, така и на доставките. Диспечерите имат възможност да поправят грешни данни, да комуникират с клиенти или доставчици. Същевременно, всички промени се отразяват в базата от данни. Уеб приложението използва техники за оптимизация, които се изпълняват непрекъснато на заден план. Уеб приложението генерира ежедневни отчети: за пробега на превозните средства, отхвърлени поръчки и извършени доставки. Мобилното и уеб приложения споделят едни и същи микроуслуги и бази от данни, които синхронизират информацията с ERP, CRM и TMS

подсистемите.

2.2. Логически модел на облачна система за управление на поръчки

Логическият модел на облачна система за управление е създаден въз основа на итеративния процес и концептуалния модел от високо ниво на ПОСУП, представени в предходния раздел. Логическия модел се състои от диаграми от типа обект-връзка, диаграми на последователности и дейности. Диаграмите представят архитектурата на облачната система, която е разделена на няколко модула. Всеки модул е проектиран да обработва специфични данни от процеса по управление на поръчки. Модулите работят съвместно, като всеки от тях е автономен и има специфични отговорности:

- Управление на поръчките: този модул включва набор от микроуслуги и NoSQL бази от данни за управлението на поръчки за продажба от бизнес клиенти. В него се обработва и съхранява информация за търговски поръчки чрез API, което интегрира вътрешна ERP подсистема.
- Управление на доставки: набор от микроуслуги и NoSQL бази от данни за управление на логистични процеси. Този модул включва данни за проследяване на превозни средства, оптимизация на графици за доставки, маршрути и координация между бизнес клиенти, доставчици и диспечери.
- Управление на потребителите: обслужва дейностите по регистрация, удостоверяване и оторизация на потребители. Използва API и релационна база от данни за достъп и управление на идентичности.

Бизнес клиенти, доставчици и диспечери използват мобилни и уеб приложения с интуитивен потребителски интерфейс, за да се свържат към облачната система. За осигуряване на висока производителност и стабилна работа на системата се прилагат подходи за разпределение на мрежовия

трафик и механизми за обработка на грешки. При неуспешни заявки се изпълняват повторни опити, както и се използва кеширане, за да се намали броят на заявките към базите от данни и вътрешните системи. За да се подпомогне надеждното функциониране на системата, се използват инструменти за мониторинг, които проследяват състоянието на клиентските и сървърните приложения.

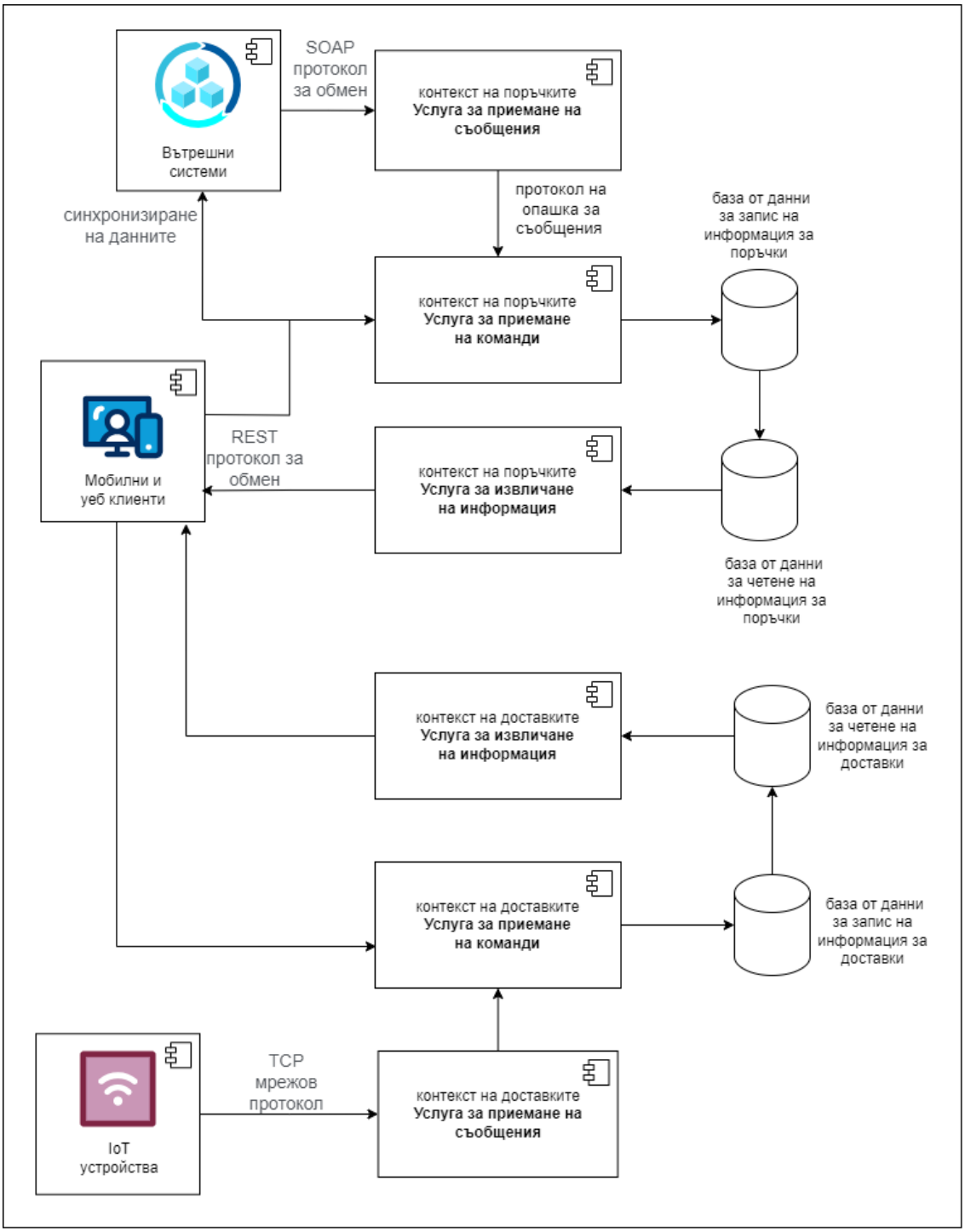
2.2.1. Модули за управление на поръчки и доставки

Въз основа на изследванията на Hartley&Sawaya (2019) е установено, че дейностите по управление на поръчки и доставки са взаимосвързани. Съгласно направените изводи в глава първа, микроуслугите, които реализират тези модули, следва да следват принципите и практиките, разгледани в раздел 1.4. За да се разграничат модулите за поръчки от модулите за доставки, са използвани „ограничени контексти“ (от английски – bounded contexts). Те са част от DDD методологията и служат за разделяне на микроуслуги в облачната система. Всяка услуга има собствени правила и бизнес логика, а във всеки ограничен контекст се използва специфичен UL език, който подпомага за разделянето на отговорностите. По този начин бъдещи промени в контекста за поръчки не засягат пряко този за доставки.

UML диаграма на компонентите[13] визуализира двата модула, техните контексти, микроуслуги и бази от данни. Илюстрира взаимодействието им с вътрешни и външни приложения. Диаграмите на компонентите по същество са диаграми на микроуслуги, използвани за моделиране на статичната реализация и документиране на системата. Поради това обстоятелство, на фиг. 2.5 са дефинирани основните компоненти на ПОСУП и връзките между тях. Фигурата включва подсистемите за управление на поръчки и доставки в обхвата на технологии като IoT, мобилни и уеб приложения, както и

[13] Както е добре известно, компонентните диаграми са подмножество на структурните UML диаграми, които предават концепциите в системата. Компонентите, показани в тези диаграми, имат прилика със съществителните, открити в естествения език, а връзките, които ги свързват, са или структурни, или семантични по природа.

вътрешни системи.



Фиг. 2.5. Основни компоненти на облачна услуга за управление на поръчките и връзките между тях

Разработка на автора

Фиг. 2.5 представя модулната структура на системата, разграничавайки отговорностите между различните компоненти и

съчетавайки няколко комуникационни протокола и услуги за данни. При създаването е използван DDD подходът, дефинирайки ограничени контексти за поръчки и доставки. Дизайнът на системата се характеризира с три типа микроуслуги: услуги за приемане на съобщения, изпълнение на команди и услуги за извличане на информация.

В съответствие с CQRS принципите, описани в т. 1.4 на първа глава, услугите за изпълнение на команди актуализират състоянието на системата, като паралелно съхраняват транзакционни данни за синхронизация с ERP и следят последователността на събитията. В същото време приемат данни асинхронно от услугите за съобщения. От друга страна, услугите за извличане на информация извършват заявки към базата от данни, без да променят актуалното състояние на данните.

Важна част от архитектурата на системата е механизмът за синхронизация, който осигурява съгласуваност на данните между отделните хранилища за запис и четене. При този механизъм се прилагат процеси за репликация, за да се поддържа постоянна актуалност на информацията. Всяка база разполага със собствени логове, регистриращи всички промени. Синхронизационният процес започва със събиране на логовете от участващите бази и сравняване на настъпилите промени. По този начин своевременно се откриват и коригират евентуални несъответствия в състоянието.

Вътрешните системи осъществяват връзка с ПОСУП посредством услугите за приемане на съобщения. Данните от облачната система и ERP се синхронизират в реално време чрез SOAP протокол. Промените, направени от бизнес клиентите, се проверяват и валидират, след което се съхраняват в съответната база от данни и се отразяват автоматично в ERP подсистемата. ПОСУП поддържа директна TCP връзка с IoT устройства, давайки възможност за интеграция на набор от сензори (Armiyanova, 2019).

Хакерските атаки не трябва да достигат до вътрешните системи, тъй като те поддържат основата на работните процеси. При най-лош сценарий,

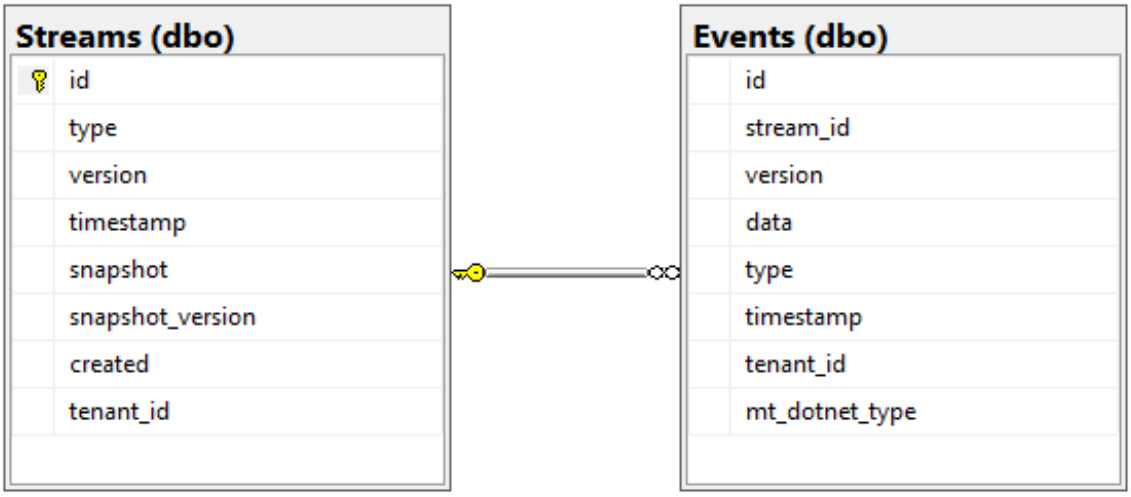
единствено облачната система следва да бъде засегната. За предотвратяване на подобни рискове, в ПОСУП се интегрира механизъм за контрол на входящия и изходящия трафик към отделните микроуслуги. Ограничава се броя на заявките, които даден потребител може да направи, с което се блокират злонамерени действия и се предотвратява прекомерната употреба на ресурси, включително спам и атаки за отказ на услуга (Denial of Service – DoS).

На фиг. 2.5 са представени четири отделни бази от данни: две, предназначени за четене и запис на информация, свързана с поръчки, и две за четене и запис на информация, свързана с доставки. Всяка база от данни е проектирана да отговаря на конкретните изисквания на съответния компонент, в който работят. По този начин, архитектурата се основава на принципите на CQRS, при които се разделят отговорностите за четене и запис. Дава се възможност за паралелна обработка на големи обеми от данни, без прекъсвания или забавяния, причинени от блокиране на ресурси (от английски – race conditions).

Според принципите на ES, изложени в т. 1.4 на първа глава, базите от данни трябва да поддържат съхранение на събития. Целта на съхраняването на събития е да се регистрират промените в състоянието на системата. Веднъж записани, събитията не подлежат на промяна. Това предоставя възможност за възстановяване на състоянието на обектите за поръчки и доставки, както и за извършване на одит на всяко действие на клиента.

Следвайки принципите на ES, предлагаме схемата на всяка една от четирите бази от данни да включва две основни таблици: „потоци“ и „събития“. Това дава възможност за сравнително лесно добавяне на нови типове потоци и събития, без необходимост от промени в структурата. Събитията се съхраняват хронологично, което дава възможност за тяхното реконструиране или анализ на по-късен етап (Young, 2011).

Фиг. 2.6 представя релационен (E-R) модел на таблиците за потоци и свързаните с тях събития.



Фиг. 2.6. Релационен (E-R) модел на таблиците за потоци и свързаните с тях събития.

Разработка на автора

Потоците служат за основа на организиране и категоризиране на събитията. Например, в базите от данни за запис и четене на информация за поръчки, таблицата „Потоци“ съдържа информация за всяка поръчка, докато таблицата „Събития“ съхранява всички свързани със съответната поръчка събития: създаване и актуализация. Тази организация осигурява поддържането на история на всяка поръчка. Всяко събитие представлява конкретно действие или промяна в състоянието на поръчката.

Табл. 2.1 описва модела и структурата на „Потоци“.

Таблица 2.1

Структура на таблица „Потоци“

Поле	Описание
Идентификатор (id)	Универсален уникален идентификатор, който представлява първичния ключ за всеки поток.
Вид (type)	Тип на потока.
Версия (version)	Номер на версията на потока.
Времеви печат (timestamp)	Датата и часът, в който записът е създаден или последно актуализиран.

Текущо състояние на потока (snapshot)	Състояние на поръчка или доставка в определената версия.

Разработка на автора

Събитията представляват основни компоненти при регистрирането на промени в състоянието и действията, извършвани в системата. Техните характеристики включват историческа неизменност и възможност за проверка. Събитията са динамични обекти, които поддържат последователност, отчетност и адаптивност. Те са част от бизнес анализа на данните, който играе основна роля в процеса на вземане на информирани решения. При създаване на поръчка чрез облачната система, в таблицата за събития се регистрира запис с информация за потребителя, дата и час за доставка и текущия статус. В таблицата 2.2 е представена структурата на таблицата за събития, както и допълнителни свойства и полета.

Таблица 2.2

Структура на таблица „Събития“

Поле	Описание
Идентификатор (id)	Универсален уникален идентификатор, който представлява първичен ключ за всяко събитие.
Идентификатор на потока (stream_id)	Свързва конкретно събитие със съответния поток чрез референция към таблицата „Потоци“.
Идентификатор на последователността (seq_id)	Идентификатор, свързан с последователността, в която се случват събитията.
Вид (type)	Тип на събитието.
Времеви печат (timestamp)	Дата и час на създаване на събитието.
Данни (data)	Данни на събитието.

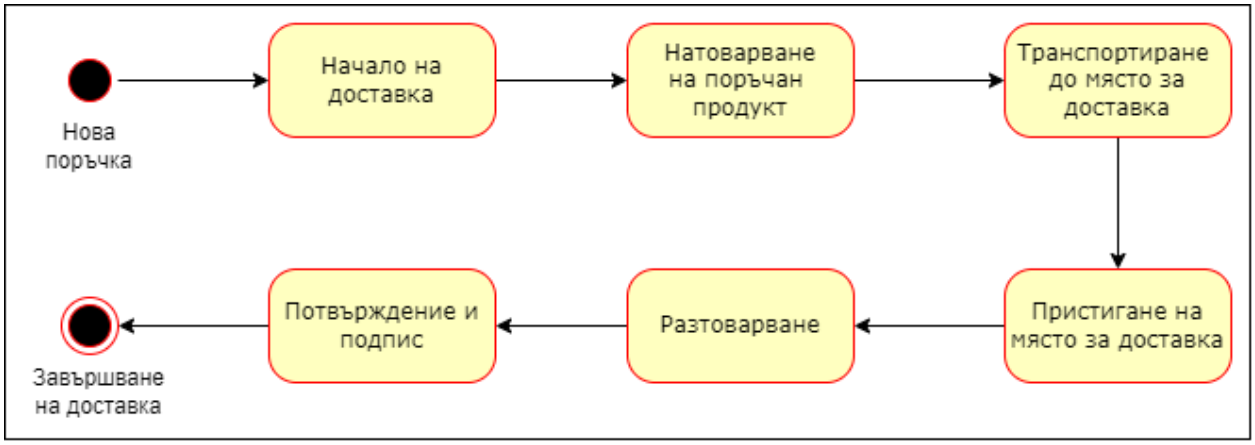
Разработка на автора

При извършване на доставка, всеки етап от работния процес се

отбелязва като отделно събитие и се съхранява в базата от данни. Това дава възможност за проследяване на статуса на доставка в реално време, анализ на логистичния процес и предоставяне на актуална информация на клиентите. Освен това се поддържа дневник на всички извършени действия.

Според някои изследователи (Hofmann et al., 2019), ефективната комуникация между технически и нетехнически лица може да бъде значително улеснена чрез употребата на UML диаграми на активността, които визуално да представят процеси и потоци от действия, което подпомага прилагането на принципите на DDD, CQRS и ES.

В тази връзка, фиг. 2.7. представя последователността на процеса по доставка, организирана под формата на UML диаграма на активността.



Фиг. 2.7. Последователност на етапите за извършване на доставка

Разработка на автора

Процесът на доставка започва с получаването на заявка за нова поръчка, което е последвано от етапа на възлагане на доставчик или „начало на доставка“. След това следва подготовката и натоварването на поръчания продукт. Продуктът се транспортира до крайната дестинация, където пристига и се разтоварва. Процесът на доставка завършва с потвърждение на получаването от клиента чрез подпис. Тази диаграма представлява модел на процеса. Някои от етапите, например управление на рекламации и комуникация с клиента, са пропуснати. От теоретична гледна точка,

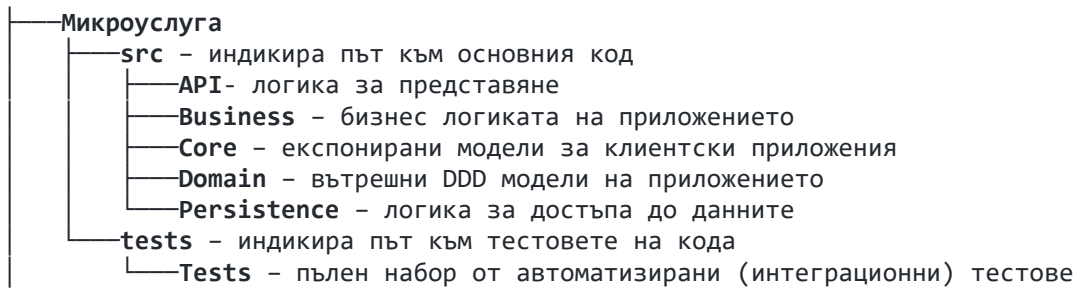
рекламациите или анулирането на доставка са част от „обратната верига на доставки“, която е разгледана в първа глава.

Всеки етап, представен на фигура 2.7, се регистрира като отделно събитие в таблицата за събития. Всяко от тях е свързано с един запис в таблицата за потоци. По този начин се проследява част от вътрешната веригата на доставки. Наблюдава движението на продукцията спрямо конкретна клиентска поръчка. Подобна архитектура на данните дава възможност за анализ на процесите в реално време, своевременно откриване на потенциални проблеми и оптимизация на отделни дейности, свързани с логистиката в предприятието.

2.2.2. Декомпозиция на модулите за поръчки и доставки на ниво микроуслуги

Логическият модел на облачната система за управление има за цел да предостави абстрактна представа за функционалните компоненти и тяхната взаимовръзка. Декомпозицията на модулите за поръчки и доставки на ниво микроуслуги представлява етап, в който логическите компоненти се разделят на малки, независими уеб услуги. Всеки от компонентите в контекстите, представени на фиг. 2.5, се намира на най-високото ниво в йерархията, предоставяйки API за връзка между данните и клиентските уеб и мобилни приложения.

Представени на фиг. 2.8, на основно ниво в структурата на изграждане на всяка услуга, стоят две основни поддиректории: „*програмен код*“ (src) и „*тестове*“ (tests), които съответно съдържат изходния код и компонентните тестове.



Фиг. 2.8. Структурата на пакетите на всяка микроуслуга

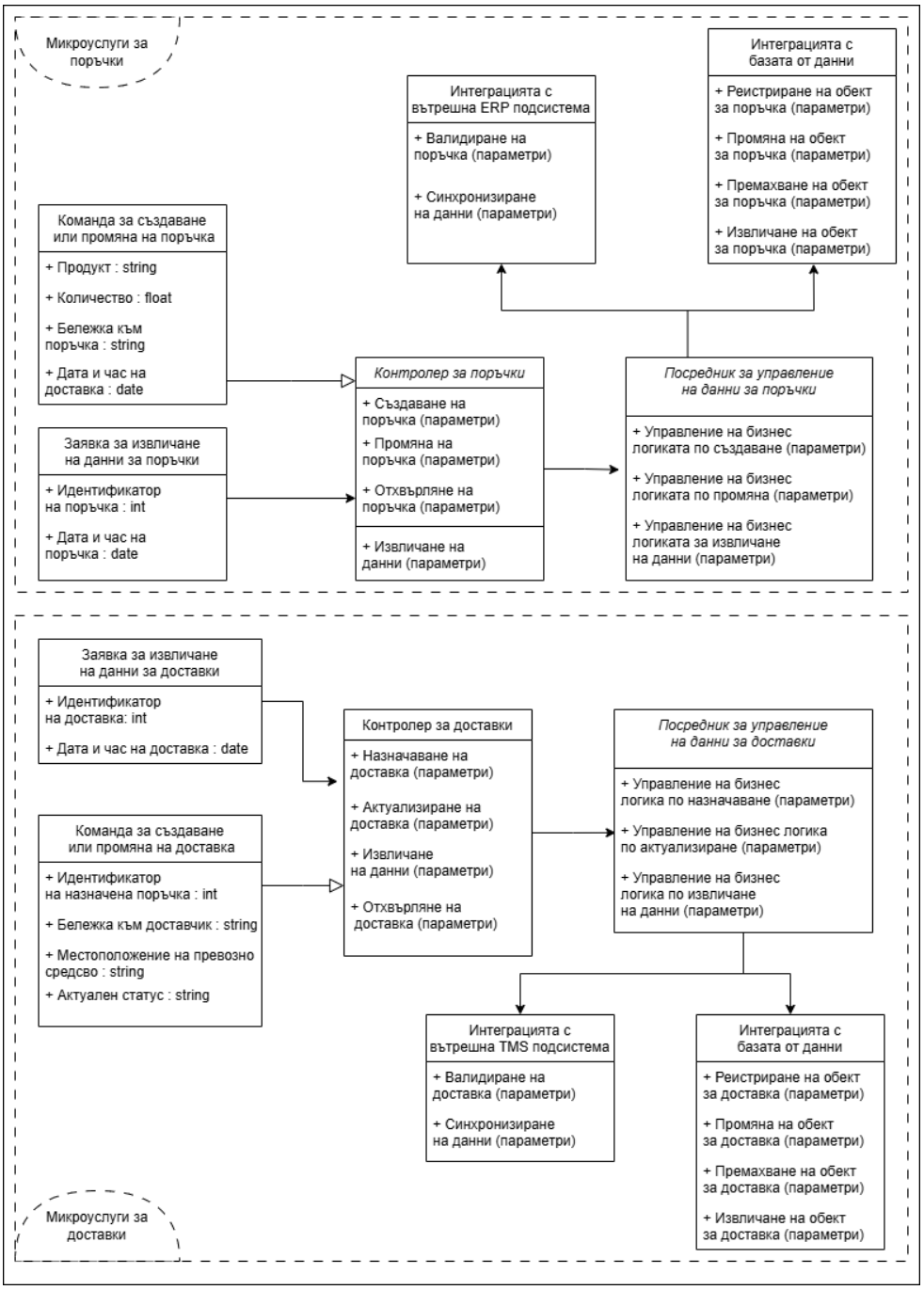
Разработка на автора

Нивото „src“ служи като централен посредник, координиращ взаимодействията между слоевете, представени от различни пакети и поддържащи принципите на DDD архитектура:

- API – входна точка за комуникация, обработвайки SOAP, HTTP, TCP заявки и отговори;
- Основен (Core) – център за команди, заявки и модели за валидиране. Той капсулира основните операции и логиката на домейна, насърчавайки възможността за повторна употреба и поддръжка;
- Бизнес (Business) – съдържа бизнес логика, организираща основните функции на приложението. Поддържат се класове „посредници“ на команди и заявки, заедно с интерфейси към външни системи. В този слой се осъществява обмяната на съобщения и изпълнението на CQRS;
- Домейн (Domain) – хранилище за агрегати, обекти и събития. Този слой съдържа основните класове на настоящата микроуслуга;
- Съхранение (Persistence) – този слой съдържа класове за интеграцията с базите от данни. Класовете изпълняват извличането и записването на информацията;
- Тестови проект – този слой бива изолиран от „src“, разположен в поддиректорията “tests”, включвайки набор от класове за интеграционни и компонентни тестове.

За да се осигури функционална съгласуваност и да се спазят основните принципи и практики на DDD, всяка услуга в системата следва да

използва сходна структура на пакетите. Всеки пакет съдържа обектно-ориентиран програмен код. За да се визуализират графично класовете, техните атрибути, методи и връзките между тях, на фиг. 2.9 е представена диаграма на класовете, която е част от логическата структура на софтуерната система.



Фиг. 2.9. Диаграма на класовете и връзките между тях в ПОСУП

Разработка на автора

Фиг. 2.9 е разделена на две основни секции: микроуслуги за поръчки и микроуслуги за доставки. И в двете секции се срещат класове за команди, заявки, мрежови контролери, посредници за управление на данни, класове за интеграция с базите от данни и с вътрешни подсистеми.

Командите за създаване и промяна на поръчки и доставки са част от основния (core) пакет. Те съдържат входни данни с атрибути като продукт за поръчка, количество, местоположение, статус, дата и час на доставка. Заявките също са част от основния пакет и обслужват извличането на данни чрез уникални идентификатори.

Контролерите, които са част от API пакета, приемат команди и заявки от клиентските приложения и връщат съответни отговори. Те са директно свързани с класовете, наречени „посредници за управление на данни“ (mediators), които съдържат основната бизнес логика. Класовете „посредници“ взаимодействат с обектно ориентираните класове за вътрешните TMS и ERP подсистеми, както и с базите от данни. Класовете за интеграция с базите от данни са част от пакета за съхранение (persistence), а класовете за интеграция с вътрешните подсистеми са част от домейна.

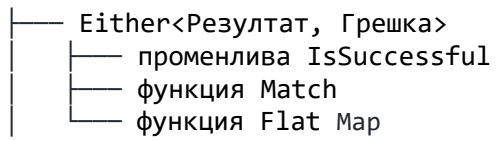
Лингвистичната рамка UL, разгледана в предходната глава, се използва в класовете за подобряване на комуникацията между членовете на разработващия екип. Тази рамка подпомага процеса по създаване на представения обектно-ориентиран програмен код. Необходимо е всички заинтересовани страни да имат цялостно разбиране на изходния код, което да им даде възможност да предлагат или одобряват подобрения, както и да откриват потенциални проблеми. Лингвистичната рамка дава възможност за плавен преход от псевдокод[14] към изпълним код, подходящ за производствена среда (Nikolov, 2019). Чрез използване на псевдокод екипите следва да създадат предварително представяне на логиката на домейна. Този

[14] Псевдокодът е неформален начин за описание на алгоритми, използващ смесица от естествен език и структурни елементи за програмиране. Той служи за нагледно представяне на логиката и последователността на действията в алгоритъма, без да се съобразява със синтаксиса на конкретен програмен език.

подход рационализира следващите фази по внедряване и поддръжка на системата.

В съответствие с принципите на DDD и UL, т.нар. Either монад се отличава като усъвършенстван инструмент за изразяване на сложна бизнес логика в програмните класове (Wlaschin, 2018). Either монадът представлява функционална програмна структура, която капсулира два възможни резултата: успешно изчисление или грешка. Бизнес логиката задава условията, при които дадена операция може да се изпълни успешно или неуспешно. Either монадът подпомага обработката на грешки, например при проблем на микроуслуга с връзката към базата от данни, като подобрява четливостта на програмния код и прави бизнес логиката по-разбираема. Прилагането на Either монада във функциите на класовете осигурява връщането на валидни резултати или дефинирани грешки.

Общата структура на Either монада е представена на фиг.2.10.



Фиг. 2.10. Структура на Either монад

Разработка на автора

Съществена част от структурата е състоянието на булевата променлива „*IsSuccessful*“ и поведението на функцията „*Flat Map*“. Когато даден монад от тип Either, обозначен като *C<T>*, представи резултат от конкретен метод, функцията FlatMap може да приложи трансформация, в зависимост от стойността на булевата променлива, по следния начин:

$$(C<T>, (T => C<T2>)) => C<T2>$$

При възникване на изключение или грешка, съответните обекти се връщат със стойност на нов, трансформиран монад. Функционалният подход на Either монада и възможностите за трансформация чрез функцията Flat Map предоставят последователна методология за изграждане на вериги от

операции, което прави кода насочен към конкретна цел. На фиг. 2.11. е представена функция, изразена чрез псевдокод, която отговаря за създаване на нова поръчка в базата от данни.

```
├──Команда за създаване на поръчка
│ Flat Map ├── валидация на входните данни
│ Flat Map ├── проверка на ресурсите на предприятието
│ Flat Map ├── запазване на промените в базата от данни
│ Flat Map ├── изпращане на данни към услугите за анализ
│ Match    ├── връщане на резултат или грешка
```

Фиг. 2.11. Примерен код за създаване на поръчка чрез използването на Either монад

Разработка на автора

Примерният псевдокод следва да се превърне в реален програмен код, който да изгради представените на фиг.2.9 класове. Използването на Either монад в кода улеснява прехода от псевдокод към програмен, адаптирайки бизнес логика в отделни, независими функции. За разлика от императивното писане на код, включващо програми, работещи с вложени кодови структури, верижният подход с Either монад предлага едно ниво на „обхват“ и ясно дефиниран ред на изпълнение (виж приложение 2).

2.2.3. Модул за управление на потребителските профили

В исторически план методът за базово удостоверяване (Basic Access Authentication) е прилаган самостоятелно във всяка услуга от информационните системи. Всяка услуга (или микроуслуга в ПОСУП) е осигурявала собствена логика за управление на потребителските профили. Макар че този подход е често срещан и сравнително лесен за реализация, той има съществени недостатъци:

- потребителското име и парола се предават кодирани във формат, който лесно може да се декодира;
- потребителското име и парола трябва да се предават с всяка заявка, което увеличава риска от прихващане на тези данни при

незащитена връзка;

- при базово удостоверяване липсва вградена поддръжка за управление на сесии.

Посочените недостатъци показват, че методът за базово удостоверяване не е подходящ за управлението на потребителските профили в ПОСУП. Основен проблем произлиза от необходимостта всяка облачна микроуслуга да обработва и защитава данни за текущия потребител и паролата му. Прехвърлянето на отговорността за управлението на потребителските профили към централизиран доставчик на идентичност значително улеснява прилагането на мерки за сигурност. Например защитено съхранение на пароли, многофакторно удостоверяване, интеграция с външни доставчици и редовно осигуряване на актуализации (Nacheva, Sulova & Penchev, 2022). Счита се, че централизираното управление помага за спазването на регулаторните изисквания. Предлага се единна рамка за прилагане на стандарти свързани с „Общия регламент относно защитата на данните“[15] (Kesan et al. 2013).

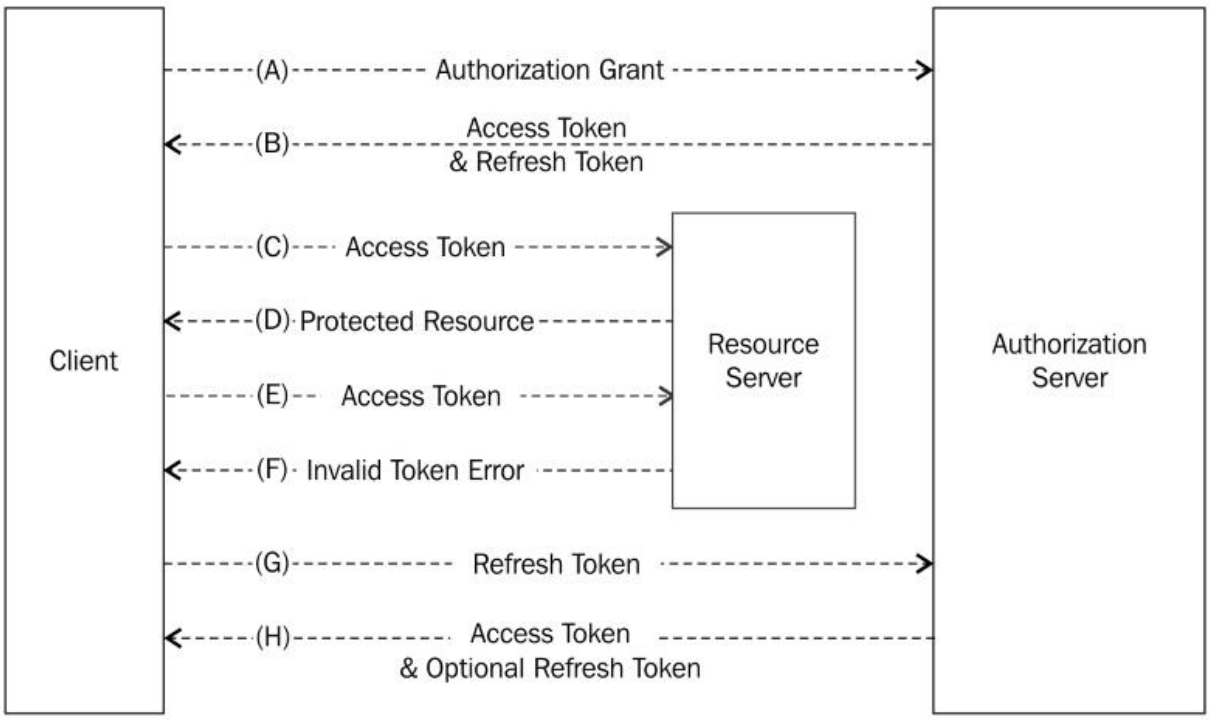
В тази насока, управлението на потребителски профили поддържа постоянна връзка с микроуслугите и осигурява защита на данните от неоторизиран достъп и киберзаплахи. Редица изследователи (Brewster, 2018; Paganini, 2019) описват конкретни примери за кибератаки, водещи до пробиви, манипулиране или загуба на данни. Описаните случаи показват съществената роля и значението на мерките за сигурност, както и необходимостта от постоянно усъвършенстване на защитните механизми. Възможно е комбинирано да се прилагат различни технологии за защита на данните: криптографски алгоритми, биометрични методи за оторизация и машинно обучение за откриване и противодействие на хакерски атаки.

[15] Общият регламент относно защитата на данните (General Data Protection Regulation - GDPR) е законодателна рамка, приета от Европейския съюз, с цел да осигури защитата на личните данни на гражданите в ЕС. Чрез въвеждане на изисквания към организациите относно прозрачността, отчетността и сигурността при обработката на лични данни, на физическите лица се дават възможности за достъп, корекция, изтриване, ограничаване на обработването и преносимост на данните. Нарушенията на GDPR могат да доведат до финансови санкции.

При тези обстоятелства, модулът за управление на потребителските профили функционира на „първа линия“ за защита, предоставяйки цялостна рамка за удостоверяване на потребителите и контрол върху техния достъп. По този начин пряко влияе върху устойчивостта на системата срещу външни заплахи за сигурността. Този модул прилага набор от практики за сигурност, включително „мрежова архитектура с нулево доверие“ (Zero Trust Network Architecture) и „достъп до гранични услуги“. Тези практики налагат идентификация, автентикация и оторизация на всеки потребител и устройство, които се опитват да получат достъп до данни за поръчки или доставки (Димитров, 2018). По подразбиране се приема, че както вътрешният, така и външният мрежови трафик може да бъде потенциална заплаха, поради това достъпът до ресурсите е контролиран.

Протоколите за удостоверяване и оторизация, прилагани при управлението на потребителски профили и контрола на достъпа, се базират на OAuth 2.0 и OpenID Connect. OAuth 2.0 е протокол, основан върху стандарта RFC 6749[16]. Приложенията изискват и получават ограничен достъп до микроуслугите за поръчки и доставки. Удостоверяването на потребителя се делегира, без да се компрометират идентификационните му данни. Тази технология се използва широко в онлайн платформи: Facebook, GitHub и DigitalOcean. OpenID Connect допълва OAuth 2.0, въвеждайки допълнителен слой за идентичност. Този слой добавя идентификационни токени, представени като JSON Web Tokens, които капсулират удостоверената потребителска информация. Така се осъществява сигурно и ефективно управление на потребителските профили и контрол на достъпа. На фиг. 2.12 е представена схема на взаимодействието между различните компоненти, свързани с OpenID и OAuth 2.0.

[16] RFC 6749 дефинира протоколът OAuth 2.0, като въвежда основни принципи, които демонстрират как клиентските приложения могат да получават ограничен достъп до уеб услуги, без да разкриват идентификационните данни на потребителя..



Фиг. 2.12. Абстрактна диаграма на удостоверяване на потребител

Източник: RFC 6749, 2012.

На база на представената фигура, в ПОСУП следва да се въведе централизиран доставчик на идентичност, който имплементира протоколите OAuth и OpenID. Централизирания доставчик на идентичност прилага стандартизиран механизъм за управление на потребителски достъп, удостоверяване, издаване и разчитане на токени за проверка на самоличността. Посочените протоколи допринасят за интеграция от високо ниво и сигурност при управлението на потребителския достъп. OAuth 2.0 предоставя набор от механизми за достъп, които отговарят на различните нужди на приложенията и нивата на сигурност.

Някои от основните типове механизми са представени в таблица 2.3.

Таблица 2.3

Видове механизми за достъп на OAuth 2.0

Вид	Случай на употреба	Описание	Съображение за сигурност
Идентификационни данни за парола	Влизане с потребителско име и парола.	Позволява на клиентското приложение директно да поиска и използва идентификационните данни на потребителя, за да получи токен за достъп.	Несигурен поради директното използване на потребителски данни.
Идентификационни данни за сървър	Удостоверяване на услуга към услуга.	Използва се за комуникация между сървъри, при която определено приложение достъпва ресурси въз основа на собствените си идентификационни данни, а не от името на конкретен потребител.	Сигурен при взаимодействия без участието на потребител, насочен към вътрешна комуникация.
Код за оторизация (Authorization Code)	Приложения, работещи на принципа клиент-сървър.	Този вид включва обмен на код за оторизация, след което пренасочва потребителя към крайна точка за токен за достъп.	Висока сигурност, минимизира риска от „фалшифициране на междусайтови заявки“ (CSRF).
Доказателствен ключ за обмен на данни (PKCE)	Публични приложения, работещи на принципа клиент – сървър.	Подобрена версия на „кода за оторизация“, при която клиентът генерира таен верификатор, използван по време на обмена на токен. Това свежда до минимум риска от прихващане на кода за оторизация.	Висока сигурност, предпазва от атаки за прихващане на код за оторизация, (man in the middle).

Разработка на автора

Код за оторизация (*Authorization Code)* е един от най-често използваните механизми за достъп в OAuth 2.0. При този механизъм клиентското приложение пренасочва потребителя към страницата за вход в системата. След успешното удостоверяване, клиентът използва т.нар. „*клиентска тайна*“ (client secret), за да подготви и изпрати HTTP POST. След като централизиран доставчик на идентичност валидира кода за оторизация, той връща на клиентското приложение т.нар. „токен за достъп“ (access token). В много случаи се генерира и „опреснителен токен“ (refresh token), който

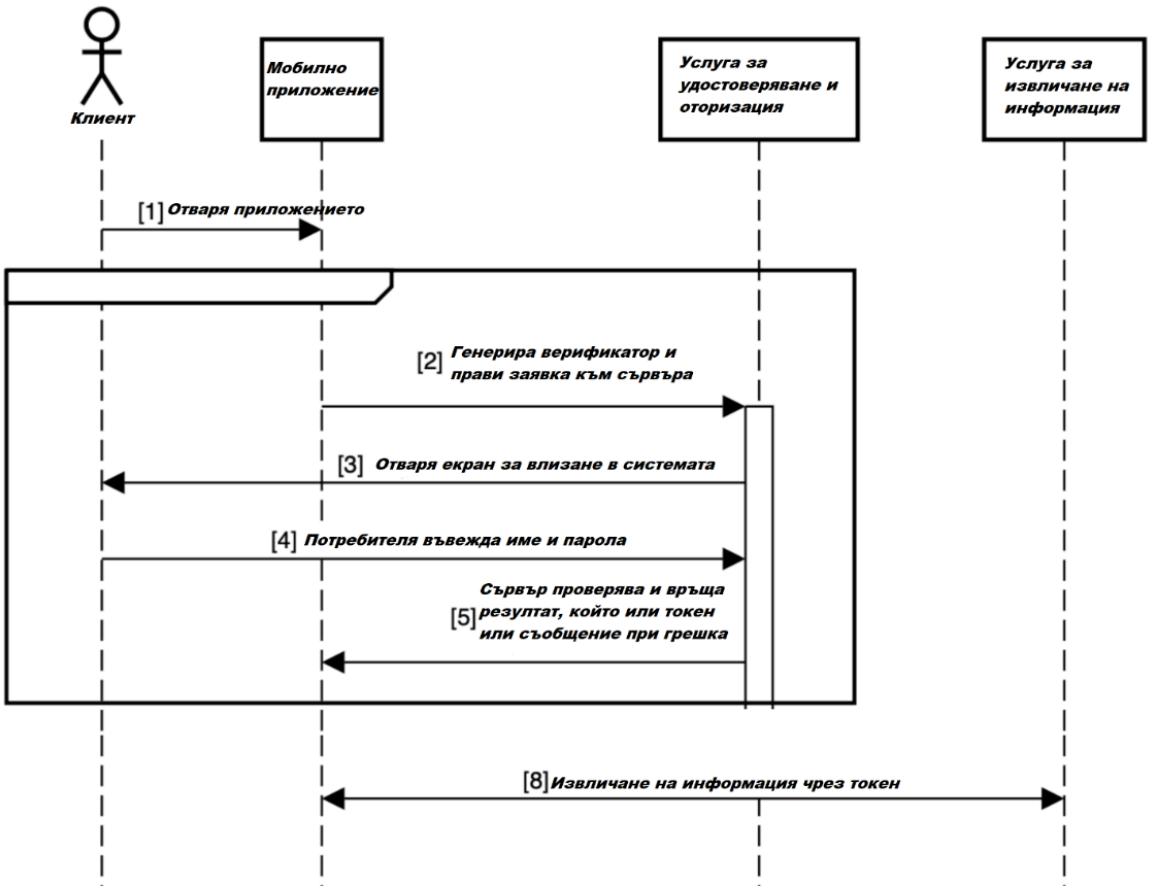
дава възможност за подновяване на достъпа без повторно удостоверяване.

Доказателственият ключ за обмен на данни (Proof Key for Code Exchange – PKCE), описан в RFC 7636[17], е механизъм за повишаване на сигурността при получаването на „код за оторизация“. При PKCE „клиентската тайна“ се заменя с динамично генериран код, наречен „верификатор“. Този верификатор представлява произволен низ от символи, който впоследствие се хешира (обикновено чрез SHA-256), за да осъществи сигурна връзка с централизирания доставчик на идентичност и да предотврати прихващане на кода за оторизация.

Процесът по удостоверяване и извличане на информация е представен чрез диаграма на последователността, в която участват бизнес клиент, мобилно приложение, централизиран доставчик на идентичност и услуга за извличане на информация. В тази диаграма, API за удостоверяване и оторизация функционира като централизиран доставчик на идентичност, отговорен за проверка на потребителските идентификационни данни.

Считаме, че PKCE е подходящ механизъм за достъп, използващ OAuth 2.0 и OpenID, при който верификаторът улеснява достигането на крайната точка за получаване на токен и осигурява сигурна потребителска сесия. Процедурните стъпки за извършване на удостоверяването в ПОСУП са илюстрирани на фиг. 2.13.

[17] RFC 7636 дава описание на „Proof Key for Code Exchange“ (PKCE), който подобрява сигурността при OAuth 2.0.

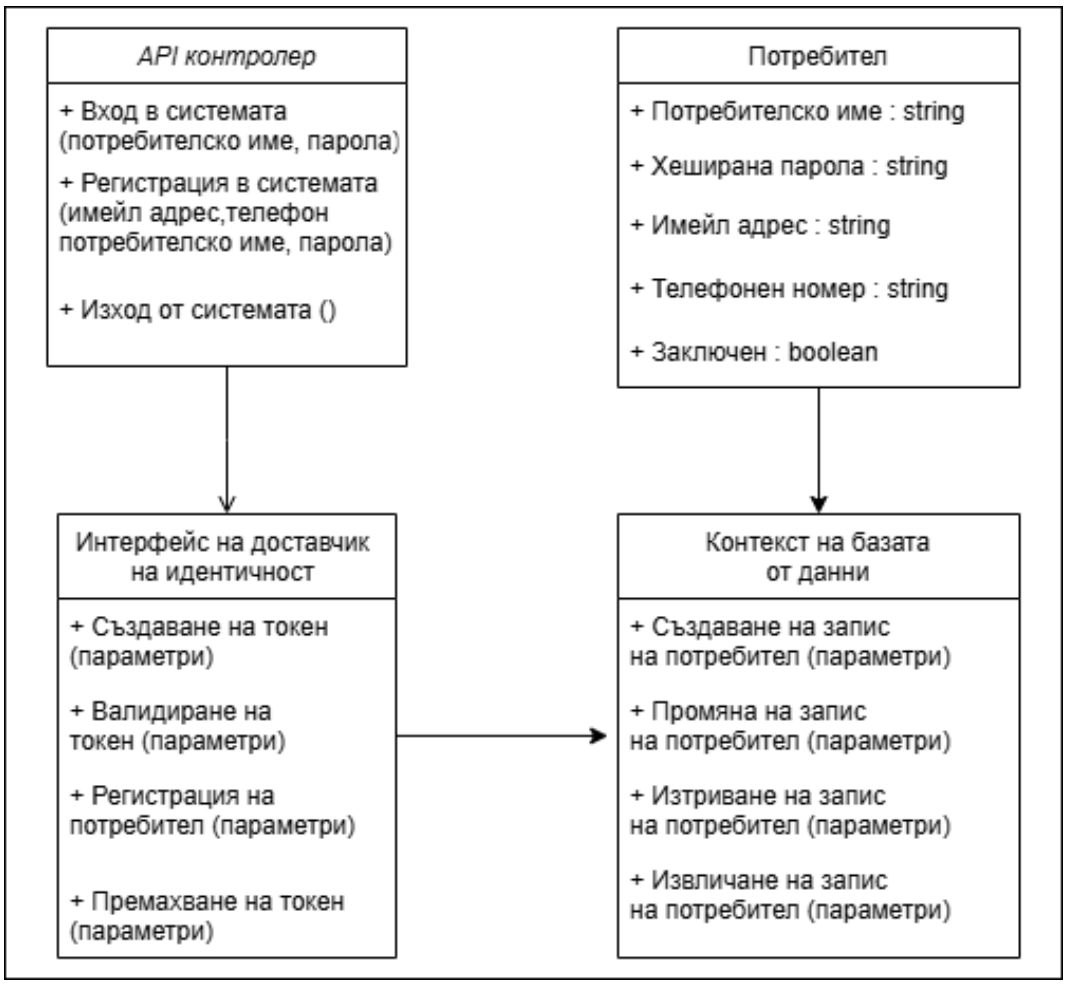


Фиг. 2.13. Диаграма на удостоверяване чрез код и ключ за обмен

Разработка на автора

Мобилното приложение инициира процеса, генерира верификатор и изпраща HTTP POST заявка към крайна точка за удостоверяване с параметри: „client_id“, „redirect_uri“, „verificator“ и „response_type“. След успешна проверка на въведените данни, сървърът издава код за оторизация, а приложението показва екран за вход в системата и банер за съгласие от страна на потребителя за използване на „бисквитки“. Потребителят въвежда име и парола, сървърът проверява данните и връща резултат: успешен токен за достъп или съобщение за грешка. След успешен вход в системата, мобилното приложение използва токена за създаване на сесия и последващо извличане на информация от съответните микроуслуги.

Диаграмата на класовете и зависимостите в централизирания сървър за самоличност, които обслужват процесите по удостоверяване и оторизация, са представени на фиг. 2.14.



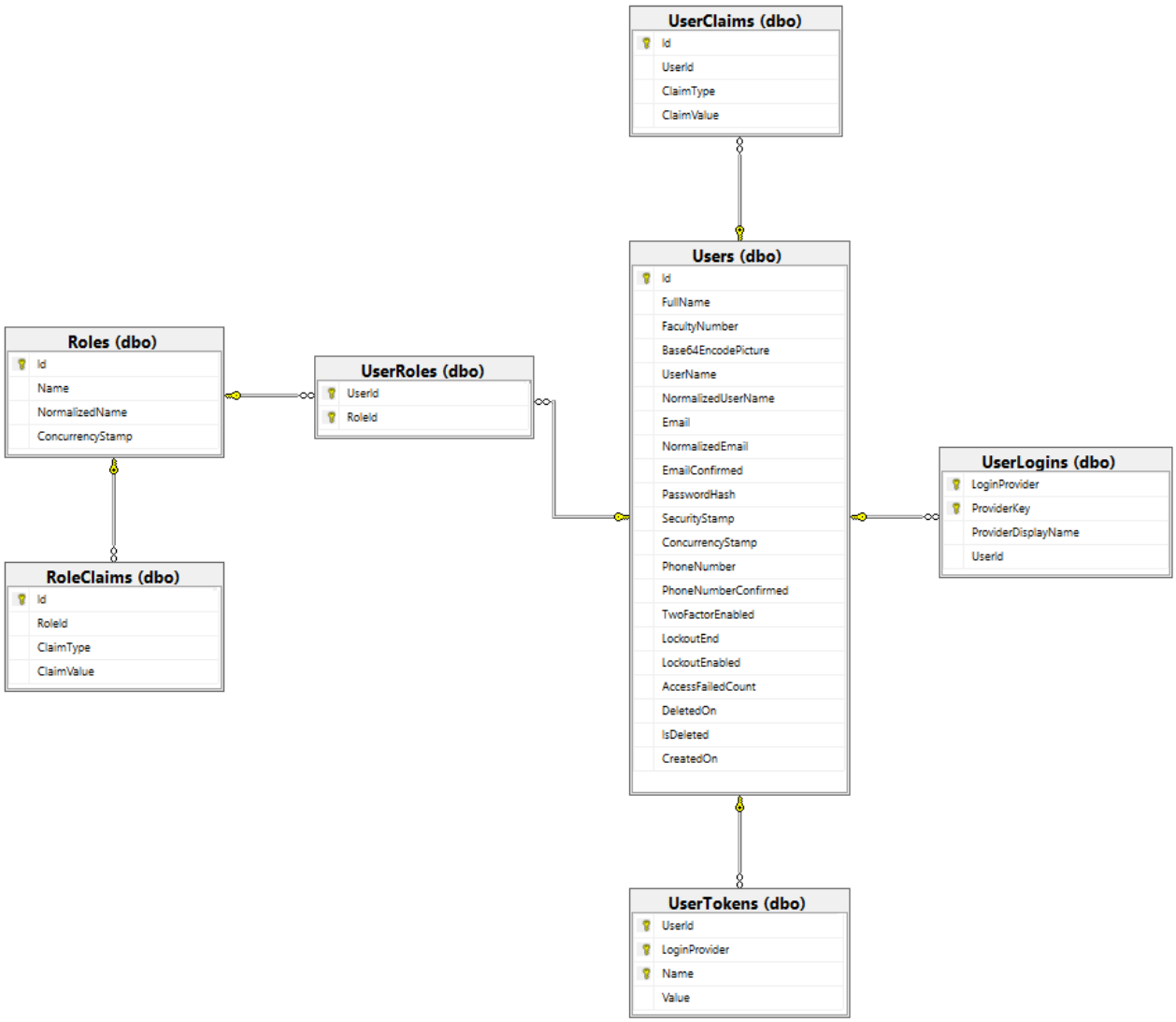
Фиг. 2.14. Диаграма на класовете, отговарящи за удостоверяване

Разработка на автора

Диаграмата на фиг. 2.14 представя структурата на основните класове в API на централния доставчик на самоличност. Контекстът на базата от данни е клас, който е част от обектно-релационния „мапър“, дава възможност на разработчиците да работят с данни чрез обектно-ориентирано програмиране, вместо SQL заявки (Kuyumdzhiev & Nacheva, 2020). Контекстът на базата от данни е свързан с класа за потребители, предоставяйки функционалност и свойства, специфични за нуждите на приложението. Обектно-ориентираните класове, свързани с интерфейса, обединяват бизнес логиката за управлението на потребителите. Те поддържат операции за създаване, актуализиране, извличане и изтриване на потребителски акаунти, както и функционалности за регистрация, вход и изход от системата. Класовете, които имплементират интерфейса, взаимодействат пряко с класовете за потребител и с контекста на

базата от данни.

Фиг. 2.15 представя релационен модел на базата от данни за потребителите.



Фиг. 2.15. Релационен модел на базата от данни за потребителите

Разработка на автора

Базата от данни, която обслужва API на централизирания доставчик на идентичност, включва следните таблици:

- Users: основна таблица, съдържаща информация за потребителите, включително имена, хеширани пароли, имейли и данни за сигурност и идентификация. Таблицата поддържа атрибути за потвърждение по имейл и по телефонен номер, двуфакторна

идентификация и управление на заключване при неуспешен достъп;

- Roles: таблица, дефинираща ролите, които могат да бъдат зададени на потребители: администратор, доставчик, диспечър или бизнес клиент;
- UserRoles: свързваща таблица между Users и Roles, което позволява на един потребител да има множество роли (най-често комбинация между администратор и диспечер);
- UserClaims и RoleClaims: таблици, които се използват да обработват „твърдения“ (claims) за потребители и роли, например потребителски права за достъп или настройки, които не са пряко включени в основните таблици за роли и потребители;
- UserLogins и UserTokens: таблици, които се използват при интеграцията на потребители от външни системи с ПОСУП. Когато бизнес клиенти използват своите акаунти от различни платформи (например: Google, Facebook и др.), за да се свържат с ПОСУП, данните за автентикация и оторизация се съхраняват в тези таблици.

2.3. Комуникационен модел между модулите

Комуникационният модел описва взаимодействието между различните модули на системата, обменяната информация и управлението на потока от данни. Основен проблем, който този модел решава, е изборът на подходящите комуникационни протоколи, като HTTP/HTTPS, SOAP, REST и gRPC (Банков & Петкова, 2024). Чрез комуникационния модел се анализират предимствата и недостатъците на протоколите, които клиентските и уеб приложения използват за свързване с микроуслугите в облачната система.

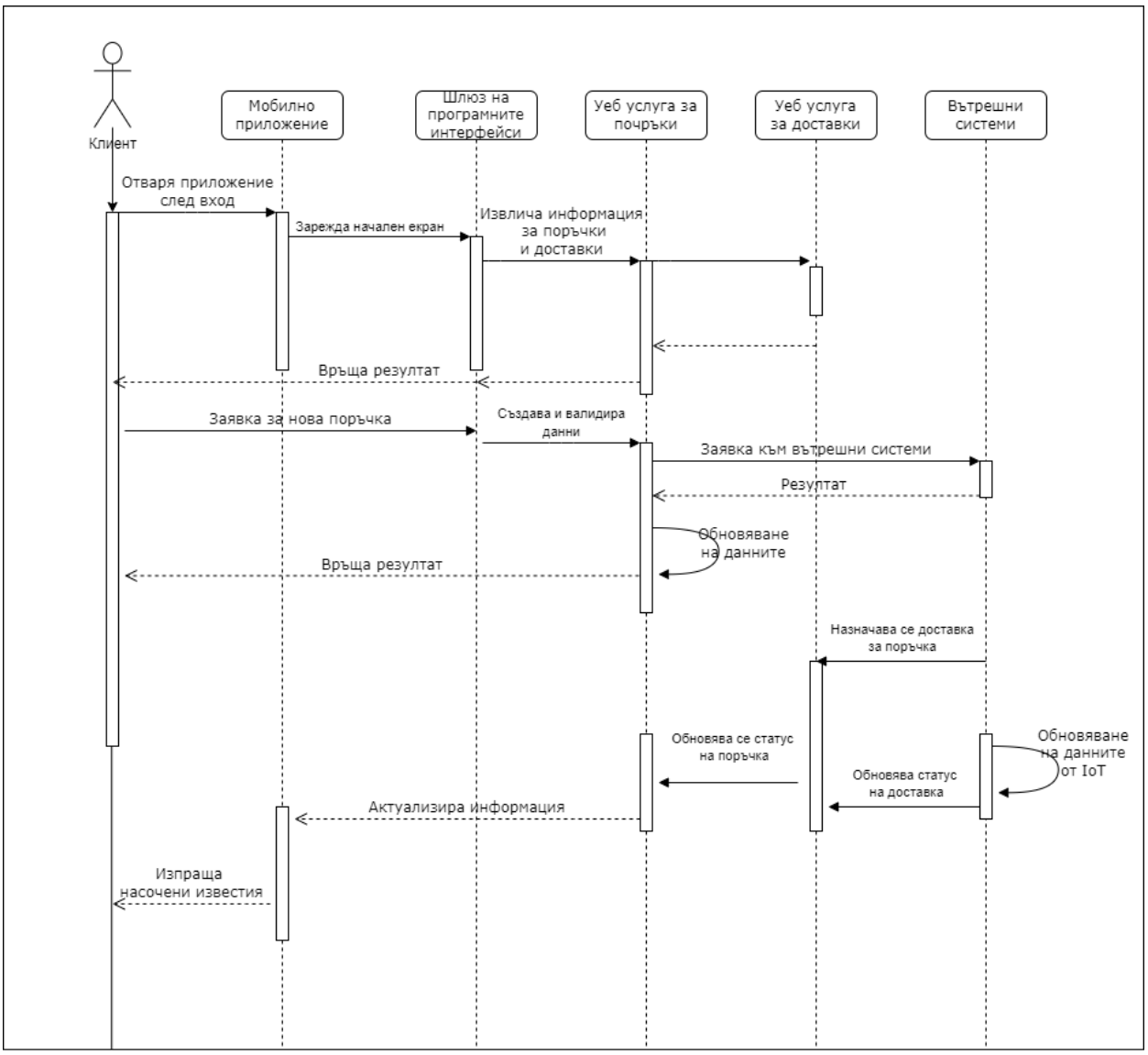
Класическите комуникационни модели включват синхронна и асинхронна комуникация. Синхронната комуникация изисква незабавен отговор на изпратеното съобщение, което изисква процесите в клиентски или

сървърни приложения да изчакват отговор преди да продължат. От друга страна, асинхронната комуникация дава възможност на приложенията да изпращат съобщения и да продължават с други задачи, като отговорите се обработват на заден план.

В диаграмите, част от концептуалния и логическия модел и представени на фигури 2.4 и 2.5, са дадени примери за използването на няколко комуникационни протокола. Протоколът SOAP се използва за предаване на структурирана информация към вътрешните системи. Протоколите HTTP, REST, gRPC и AMQP се използват за обмен на данни между мобилни и уеб клиентски приложения и микроуслугите на облачната система. В съответствие със стандартите за архитектура на облачни услуги, инфраструктурата включва IoT устройства, които използват TCP връзка за предаване на данни от сензори, прикрепени към превозните средства за доставка (Huang et al., 2013).

Чрез UML диаграма на последователностите може да се покаже как продуктите и услугите в ПОСУП си взаимодействат, за да изпълнят основната функционалност на системата. В резултат от това, фиг. 2.16 представя диаграма на последователността за основен сценарий, който се очаква да бъде изпълнен от бизнес клиенти. На диаграмата се визуализират времевата линия и редът на операциите. Въз основа на DDD концепциите, на диаграмата са показани последователността от събития в отговор на конкретна бизнес заявка, както и взаимодействието между мобилното приложение и облачните микроуслуги.

Основната цел на диаграмата е да онагледи интерактивното сътрудничество между отделните компоненти на системата и поради това е от съществено значение за подпомагане както на техническите, така и на нетехническите заинтересовани страни. Като елемент от поведенческите диаграми в UML, моделът се фокусира върху динамичните аспекти на ПОСУП.



Фиг. 2.16. Диаграма на последователността на бизнес сценарий за изпълнение на клиентски заявки в ПОСУП

Разработка на автора

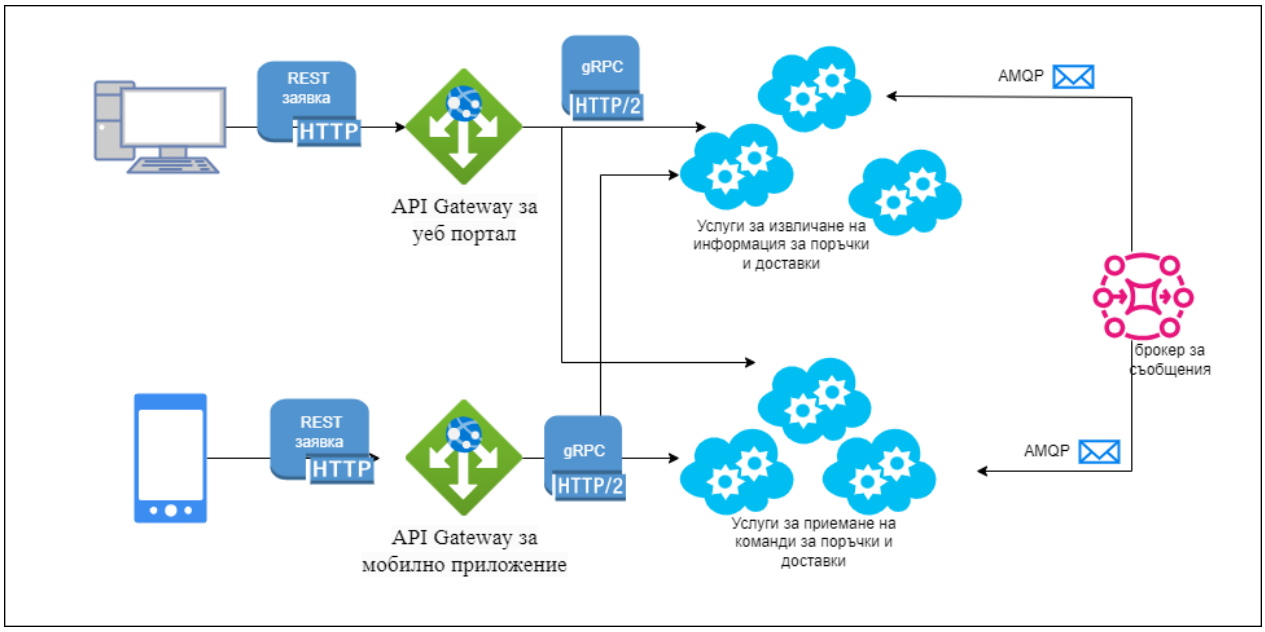
На диаграмата на последователността е представен ходът на комуникацията между бизнес клиент и мобилно приложение, както и услугите, които обработват получените заявки. Този процес е продължение на предходната процедура, започваща след стартиране на мобилното приложение и успешно удостоверяване. Бизнес клиентът има възможност да прегледа информация за текущи, предстоящи и минали поръчки, както и да създаде нова, промени съществуваща или отмени вече назначена за доставка. Системата автоматично променя статусите на поръчките и доставките и след

това предава информацията обратно на клиента.

На базата на концептуалните и логическите модели и с цел тяхното усъвършенстване, в архитектурата на облачна система за управление на поръчки се внедрява шлюз за приложни интерфейси (API Gateway). Той предоставя централна входна точка за множество микроуслуги, следвайки принципа на дизайн „*фасада*“ (Nguyen et al., 2019). Основни характеристики на API Gateway са:

- Маршрутизация на HTTP заявки от клиентски приложения: API Gateway приема входящи заявки и ги пренасочва към съответните микроуслуги, в зависимост от зададените правила за маршрутизация.
- Управление на сигурността: API Gateway осигурява удостоверяване и оторизация на заявките, валидира токени и сертификати, генерирани от централния доставчик на идентичности.
- Баланс на натоварването: API Gateway може да разпределя натоварването между различни инстанции на микроуслугите, за да се осигури висока производителност и достъпност.
- Кеширане: API Gateway кешира често използвани HTTP заявки. По този начин се намалява времето за отговор и се облекчава натоварването на микроуслугите и базите от данни.
- Трансформация на заявки: API Gateway може да преобразува формати на данни и структури на заявки, за да осигури съвместимост между различни микроуслуги. Например, HTTPS заявка от мобилно приложение, приета в API Gateway, се преобразува към gRPC заявка към микроуслуга за извличане на информация за поръчки или доставки.
- Мониторинг: API Gateway събира и анализира статистики за заявките. Агрегира информация за производителността и възможните проблеми в системата.

Интегрирането на API Gateway в облачната инфраструктура подобрява сигурността. Също така осигурява поддръжката на връзките към микроуслугите (Xu, Jin & Kim, 2019). С оглед на това, фиг. 2.17 представя API Gateway в рамките на ПОСУП.



Фиг. 2.17. API Gateway в рамките на ПОСУП

Разработка на автора

За да осигури обобщена информация за поръчки и съответните доставки, след получаване на HTTP заявка от мобилното или уеб приложение, API Gateway я разделя на отделни части и я препраща към съответните микроуслуги чрез gRPC. След това комбинира получените резултати и ги връща обратно към клиентското приложение в единен отговор.

API Gateway може да прилага правила за ограничаване на достъпа на определени IP адреси. По този начин защитава микроуслугите от претоварване или хакерски атаки. API Gateway използва механизъм за контрол на броя заявките, които могат да бъдат изпратени в определен период от време. Администраторите могат да задават конкретни лимити, например максимум 1000 заявки за минута. Когато клиентът достигне зададения лимит, API Gateway блокира следващите заявки от този IP адрес до

изтичане на времевия период. В този случай, API Gateway връща съобщение за грешка „*HTTP 429 Too Many Requests*“, указващо, че лимитът е достигнат.

За да се свържат клиентските приложения със сървърната част, API Gateway дефинира поредица от крайни точки (таблица 2.4), които отговарят на стандартите на RFC 2616[18]. Крайните точки използват Representational State Transfer (REST) архитектурен стил, за да осигурят стандартизирани интерфейси за комуникация. HTTP методите дават възможност на клиентските приложения да взаимодействат с дадена крайна точка по различни начини. Една и съща крайна точка може да поддържа различни HTTP методи, за да предостави различна функционалност.

Таблица 2.4

Крайни точки на API Gateway и техните REST ресурси

Крайна точка	**GET**	**POST**	**PUT**	**DELETE**
/orders	Извлича всички поръчки	Създава нова поръчка	Общо актуализиране на поръчките	Премахва всички поръчки
/orders/id	Извлича детайли за поръчка по идентификатор		Актуализира данните за поръчка 1, ако съществува	Премахва поръчка 1
/orders/id/ deliveries	Извлича доставките по идентификатор	Създава нова доставка за поръчка по идентификатор	Общо актуализиране на доставките за поръчка по идентификатор	Премахва всички доставки за поръчка по идентификатор
/deliveries /id	Извлича детайли за		Актуализира данните за доставка	Премахва поръчка по

[18] RFC 2616 е спецификацията за HTTP, която описва основните методи на заявки като GET, POST, PUT и DELETE. Също така и статусните кодове за отговори (например 200 OK, 404 Not Found), както и детайли за структурирането и обработката на HTTP съобщенията.

	доставка по идентификатор		по идентификатор, ако съществува	идентификатор
/users	Извлича всички потребители			
/user/id	Извлича потребител по идентификатор	Създава нов потребител	Актуализиране на потребител по идентификатор	Премахване на потребител
/auth/login		Вход в системата (създава нов токен)		
/auth/logout		Изход от системата (изтрива съществуващ токен)		

Разработка на автора

Уеб услугата на API Gateway поддържа JSON (application/json) като формат за обмен и представяне на данни. Например, заявка към посочения по-горе ресурс за детайли на поръчка връща следния отговор във формат JSON:


```
{
   "order_id": 12345,
   "customer_name": "Иван Иванов",
   "order_date": "30-10-2020",
   "total": 75.50,
   "status": "in progress",
   "items": [
      {
         "item_id": 1,
         "product_name": "Продукт А",
         "quantity": 2,
```

```
            "price": 20.00
        },
        {
            "item_id": 2,
            "product_name": "Продукт Б",
            "quantity": 1,
            "price": 35.50
        }
    ],
    "iot_data": [
        {
            "latitude":44.19499291706286,
            "longitude":-30.24615035459843,
            "altitude":100,
            "accuracy":150,
            "altitudeAccuracy":80,
            "timestamp":"30-10-2020"
        }
    ]
}
```


2.4. Функционалност и потребителски интерфейс

Потребителският интерфейс е визуалната част на системата, която се представя пред крайния потребител. Добре проектираният интерфейс допринася за положителния потребителски опит, правейки функционалностите на информационната система интуитивни и лесни за използване. Визуалните интерфейси на мобилни и уеб приложения са създадени с цел потребителите да изпълняват своите задачи с минимално усилие и време, което от своя страна увеличава ефективността и удовлетворението от използването на системата.

Както бе разгледано в предходна точка 2.3, функционалностите на

мобилното и уеб приложението започват с процес на удостоверяване и оторизация на потребителите. Първата стъпка за достъп до системата изисква въвеждане на потребителско име и парола. Началният екран в двете приложения изисква идентификация. На този етап може да се приложат допълнителни механизми за удостоверяване чрез външни доставчици, както и двуфакторна автентикация. Посочените мерки са от ключово значение за осигуряване на конфиденциалността и интегритета на потребителските данни.

На фиг. 2.18 е представена скица на екран за вход в системата.

Фиг. 2.18. Скица на екран за вход в системата

Разработка на автора

Ако след въвеждане на потребителско име и парола, доставчикът на идентичност определи текущия потребител като **бизнес клиент,** на

мобилното приложение се зарежда екран, съдържащ списък с всички негови поръчки. На фиг. 2.19 е представена скица на екрана.

Фиг. 2.19. Скица на екран на мобилно приложението за бизнес клиент

Разработка на автора

В горната част на екрана се намира информация за текущия потребител и инструмент за избор на дата. Представени са основни бутони, които включват:

- Бутон „Меню“ – предоставящ достъп до основни опции и настройки на приложението;
- Бутон „Нова поръчка“ – визуализира екран за регистриране на нова поръчка.

Под бутоните се визуализира списък с предстоящи, текущи или завършени поръчки. Всяка поръчка в списъка е представена с уникален

номер (напр. №38310376) и съответен статус, който е кодиран с различни цветове: активна поръчка е отбелязана със зелен текст, завършена поръчка със син и отказана поръчка с червен.

Освен статуса, за всяка поръчка са налични и допълнителни детайли, като:

- Очаквана дата и час на пристигане на доставчик;
- Карта с текущото местоположение на активна доставка;
- Функция за пряк контакт с доставчик или диспечер.

Всеки елемент от списъка на екрана може да бъде натиснат за допълнителни действия: преглед на детайли или редактиране на поръчката. Ако бизнес клиентите се съгласят на използване на бисквитки, всяко тяхно действие, например натискане на бутон, въвеждане на данни или отваряне на екран, се проследява и изпраща към облачните микроуслуги. Това дава възможност за събиране на данни за поведението на потребителите, които по-късно могат да бъдат анализирани за подобряване на потребителския опит (user experience). В комбинация с информацията, извлечена от облачните бази от данни (описани в раздел 2.2), става възможно откриването на тенденции или проблеми.

Използването на мобилното приложение от **доставчик** има някои специфични особености. В горната част на екрана, в менюто се включва бутон за изпращане на съобщение за повреда към диспечера, както и бутон за пряка връзка с бизнес клиента. Това улеснява навременната реакция при възникване на непредвидени инциденти.

Списъкът с предстоящите доставки, възложени на текущия доставчик, е визуализиран в основната част на мобилното приложение. На фиг. 2.20 е представена скица на екран, който се визуализира на мобилното приложение, когато доставчик приеме доставка. Екранът съдържа информация за детайлите на доставката: количество, местоположение за товарене и разтоварване, както и планирани часове. Тъй като дадена доставка може да бъде анулирана или пренасочена към друга поръчка, приложението

периодично изпраща запитвания към сървъра (препоръчително на всяка секунда) за да актуализира промените.

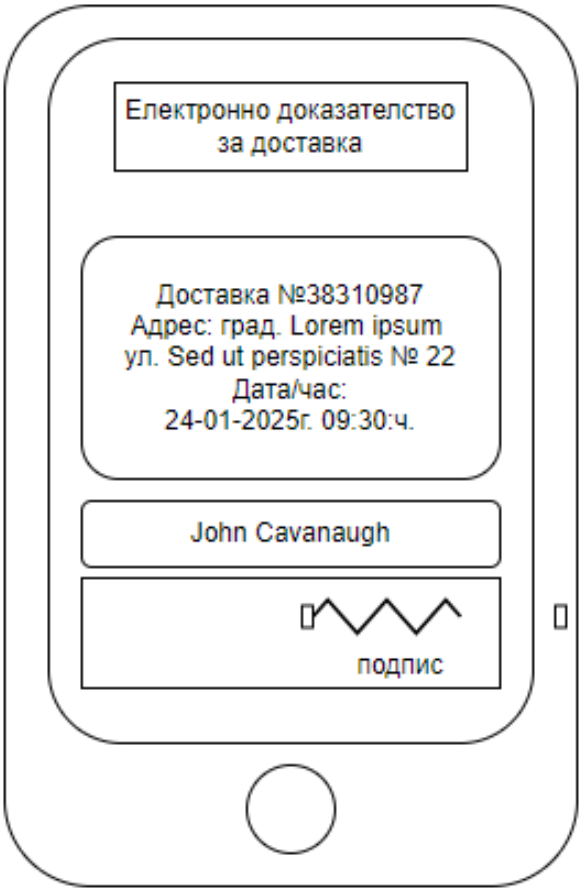
Фиг. 2.20. Скица на екран за доставка

Разработка на автора

За подобряване на процеса по документиране, приложението предлага функционалност за електронно доказателство за доставка. Тази функция генерира документ, който удостоверява получаването на стоката от бизнес клиента. Електронното доказателство за доставка осигурява прозрачност на доставките. Записват се точните данни за време, дата и местоположение на получаването. Електронните документи значително улесняват съхранението на данни, намалявайки нуждата от физически архиви. Те се прехвърлят от облачната система към ERP и SCM подсистеми, като същевременно се използват и за последващо фактуриране.

На фиг. 2.21 е представена скица на екран за електронно

доказателство за доставка.



Фиг. 2.21. Скица на екран за електронно доказателство за доставка

Разработка на автора

Електронното доказателство потвърждава получаването на стоката от бизнес клиента чрез електронен подпис. За да се гарантират автентичността и целостта на данните в електронното доказателство за доставка, се използват криптографски техники. Електронните доказателства за доставка трябва да съответстват на местните и международни стандарти и нормативни актове. Това осигурява правната сила и приемането на електронното доказателство в случаи на претенции от страна на бизнес клиентите.

Във връзка с функционалността и потребителския интерфейс на уеб приложението, предназначено за използване от **диспечерите**, фиг. 2.22 показва главния екран, след влизане в системата. То е адаптирано за използване през настолни компютри. Поради необходимостта диспечерите да

имат достъп до микроуслугите за поръчки, доставки и потребители, уеб приложението е наречено „*уеб портал*“.

Фиг. 2.22. Главен екран в уеб портала, използван от диспечерите

Разработка на автора

Уеб порталът интегрира набор от функционалности, които дават възможност на диспечерите да управляват оперативните процеси във веригата на доставки. Функционалностите включват проследяване на поръчки, управление на логистични операции и поддръжка на потребителски акаунти. Екранът, представен на фиг. 2.24, е насочен към планиране на доставките към поръчки за продажби в реално време. В заглавната част са разположени няколко основни бутона, които улесняват диспечерите при изпълнение на:

- Нова поръчка – създаване на нов запис в графика чрез въвеждане на необходимата информация за дата, час, дестинация и

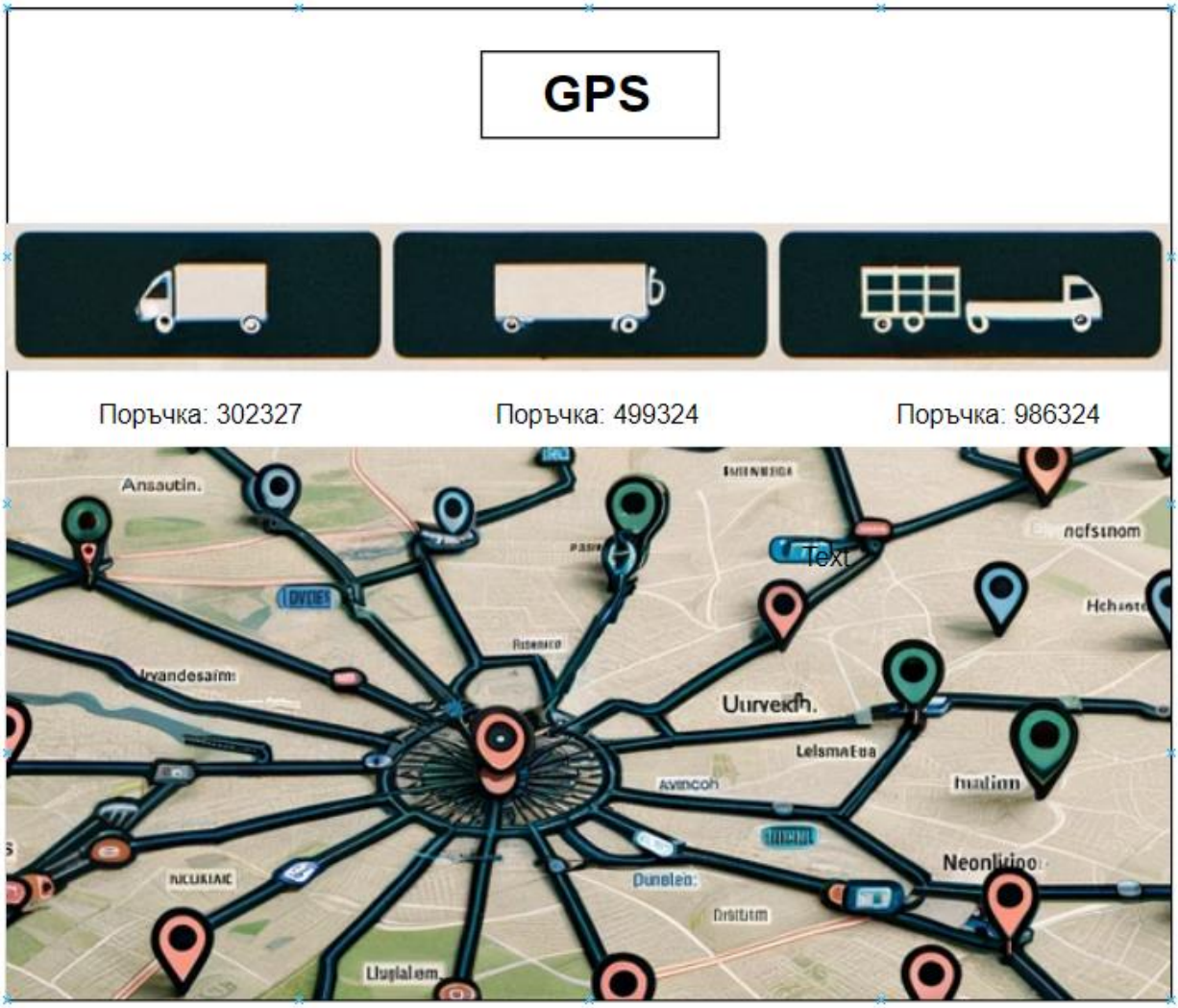
количество.

- Промяна – дава възможност за редактиране на съществуващи поръчки или доставки, например промяна на часа или пренасочване към друг доставчик.
- Уведомление – служи за изпращане на известия към доставчици или бизнес клиенти при настъпили промени в графика.

Графикът, разположен в основната част на екрана, е организиран като времева линия, оптимизирана за ежедневно планиране. Времевите слотове са разделени по часове, а всеки доставчик има собствен ред в таблицата. Полетата с поръчки са позиционирани спрямо времето и доставчика, което показва, че процесът е динамичен и дава възможност за планиране на множество задачи едновременно. В екрана са налични възможности за търсене на конкретна поръчка или прилагане на филтри по статус, доставчик и времеви интервал.

Уеб порталът осигурява съвместимост с мобилното приложение, както и с ERP и SCM подсистемите, което улеснява добавянето и актуализирането на информацията. По този начин диспечерите могат бързо да реагират на промените в оперативната среда и да оптимизират наличните ресурси. Данните се актуализират в реално време и след всяка ERP транзакция. Диспечерите имат възможност ръчно да променят статуса на дадена доставка, ако сигналът от сензорите на превозвачите бъде загубен. Чрез уеб портала се генерират различни видове отчети и анализи, подпомагайки вземането на информирани решения.

Уеб порталът поддържа функционалност за изчисляване на времето, необходимо за достигане на дадено превозно средство от точка А до точка Б. Това дава възможност приложението да се използва като инструмент за изчисляване на разстоянието и времетраенето на пътуването. На фиг. 2.25 е показан екран на уеб портала, който визуализира текущите местоположения и очакваните времена за пристигане на различни превозни средства.



Фиг. 2.23. Екран за маршрутизация на превозните средства

Разработка на автора

Уеб порталът и мобилното приложение използват GPS система, която предоставя обратна връзка на диспечерите. Уеб порталът използва събития за местоположение, за да изчисли приблизителната продължителност на пътуването въз основа на геокоординатите, изпратени от мобилното приложение и сензорите, прикачени към превозните средства. Интеграцията между уеб портала, мобилното приложение, ERP и SCM подсистемите оптимизира управлението на транспорта и подобрява ефективността на логистичните операции чрез подобряване на точността на доставките. В същото време, GPS системата осигурява прозрачност и контрол върху транспортните процеси.

Дизайнът на приложенията е разработен по начин, който улеснява добавянето на нови функционалности и модифицирането на съществуващите, съобразно нуждите на компанията. Това осигурява разширяемост и адаптивност на системата, осигурява гъвкавост и устойчивост спрямо бъдещи промени.

Изводи и обобщения към втора глава

В настоящата глава представихме концептуален, логически и комуникационен модел на облачната информационна система. Основната цел на моделите е да представят взаимодействието между диспечери, доставчици и бизнес клиенти на производствено предприятие. Процесите, които са представени чрез модели, са свързани със създаване, промяна или отхвърляне на поръчки за продажби, координацията по време на доставка и комуникацията между отделните страни. Предложените модели изпълняват втора и трета задача, свързани с архитектурата на системата, и са основни приноси на настоящия дисертационен труд.

Реализирането на концептуалния модел преминава през итеративен процес, включващ няколко основни етапа: прогноза за растежа на системата, дефиниране на основните бизнес сценарии и преглед от високо ниво на концепцията. Логическият модел има за цел да представи рамка на взаимодействие между модули и компоненти. Служи като план за проектиране на софтуера в съответствие с теоретичната основа, разгледана в първа глава. В него се описват обекти, техните свойства, взаимовръзки и процеси в системата. Това се постига чрез създаване на диаграми от тип „обект-връзка“, диаграми на последователности и диаграми на дейности.

В резултат на нашето изследване дефинирахме три основни модула: поръчки, доставки и управление на потребителски профили. За да се придобие цялостна представа за архитектурата, зависимостите и функционалните възможности на модулите, всеки от тях бе разгледан поотделно. Представен е комуникационен модел между основните модули и

протоколите за връзка между тях: HTTP/HTTPS, SOAP, REST и gRPC.

Проведена е оценка на предимствата и недостатъците на различните методи за комуникация, прилагани в клиентски и уеб приложения. Представени са синхронни и асинхронни технологии за обмен на данни между клиентските приложения, микроуслугите, вътрешните и външни подсистеми на предприятието. Спецификите и взаимовръзките на тези модели са илюстрирани чрез фигури и диаграми.

Използвана е технологията за API Gateway като централна точка за управление на маршрутизацията, натоварването, кеширането и мониторинга на микроуслугите. Заедно с това се обосновава и използването на централизиран доставчик на идентичност, който да осигурява удостоверяване и оторизация на потребители.

Описани са основни екрани и крайни точки на мобилните и уеб приложения, включително потребителските интерфейси и API. Клиентските и сървърните приложения използват стандартни HTTP методи и различни формати за пренос на данни.

На база теоретичните постановки, развити във втора глава, в трета глава се разглеждат въпроси, свързани с практическото приложение на системата в конкретно производствено предприятие.

Глава 3. Изграждане и използване на персонализирана облачна система за производствено предприятие

В трета глава се разглеждат практико-приложни въпроси, свързани с внедряването на ПОСУП в конкретно производствено предприятие – „Хейделберг Цимент Девня“ АД. Разгледани са основни характеристики на дейността на компанията и са избрани подходящи технологични средства за физическа реализация на системата, което дава възможност системата да бъде апробирана в реална работна среда. Оценяват се възможностите и функционалностите, методите за мониторинг, както и прогнозните разходи за внедряване на облачната информационна система.

3.1. Обща характеристика на дейността на „Хейделберг Цимент Девня“ АД

„Хейделберг Цимент Девня“ АД е водещ производител на цимент в България, разположен в град Девня, област Варна. Освен цимент, предприятието предлага различни бетонови смеси и специализирани строителни материали. Компанията е дъщерно дружество на немската мултинационална корпорация Heidelberg Materials.

Според бизнес доклад (Георгиева, 2025) „Хейделберг Цимент Девня“ е класирана като най-рентабилната компания за строителни материали за 2023 г. В същия доклад се посочват проекти, върху които Heidelberg Materials се фокусира – както на глобално, така и на национално ниво. Сред технологичните нововъведения, планирани за интегриране в над 50 държави, е внедряването на облачни информационни системи в търговските организации на компанията.

Основната дейност на „Хейделберг Цимент Девня“ АД включва производство и дистрибуция на цимент, инертни материали, готови бетонови смеси и асфалт. Компанията произвежда и доставя бетонова смес, приготвена

в централизирани съоръжения за дозиране. Смесите се създават съгласно спецификациите на бизнес клиентите и отговарят на изискванията за здравина, обработваемост и издръжливост. Обикновено готовите смеси се транспортират с камиони, оборудвани с миксери, и трябва да бъдат използвани незабавно след пристигането им. Поради това проследяването на точното местоположение на камионите е от съществено значение.

Монтирането на сензори върху превозните средства, които в реално време изпращат данни за нивото на водата, температурата и основните характеристики на сместа, осигурява високо качество на продукта, тъй като свойствата му могат да се променят по време на транспортиране. Счита се, че централизираното смесване е по-екологично от смесването на строителната площадка. Генерират се по-малко отпадъци и се дава възможност за по-голям контрол върху използваните материали (Delnavaz et al. 2022). Продуктите на компанията намират приложение в строителството на жилищни сгради, инфраструктура, търговски и промишлени обекти.

„Хейделберг Цимент Девня“ АД използва вертикално интегриран бизнес модел, който обхваща всички дейности на производствената верига – от добива на суровини до доставката на готовия продукт. Тази вертикална интеграция дава възможност на компанията да осигури контрол върху качеството на продуктите, да намали разходите от външни партньори и да осигурява надеждност на доставките.

Въз основа на проучванията, представени в предходните две глави, внедряването на ПОСУП в компания „Хайделберг Цимент Девня“ има за цел да оптимизира бизнес процесите във вътрешната веригата на доставки, управлението на ресурсите и повишаването на конкурентоспособността. За разлика от други снабдителни вериги, при които участват множество различни фирми и подизпълнители, „Хайделберг Цимент Девня“ прилага интегриран модел, обхващащ производството, транспортирането и доставката на готовите продукти. Прилагането на ПОСУП би улеснило събирането и обработката на данни от различните отдели в организацията.

ПОСУП е адаптирана към специфичните нужди на компанията и цели да рационализира управлението на поръчки и доставки, което е от първостепенно значение за предприятие с висока степен на сложност на логистичните и производствените процеси.

В настоящия раздел са разгледани текущите оперативни процедури в производствено предприятие, обхващащи обработката на клиентски заявки и последващите логистични дейности по доставки на бетон, пясък, чакъл, цимент или асфалт. При приемането на поръчка от клиент, диспечер въвежда заявката в SAP ERP системата с начален статус „непотвърдена“. След това системата изпраща известие до отговорния отдел за проверка на наличността на материалите и ресурсите. След като бъде потвърдена наличността, заявката получава статус „потвърдена от предприятието“ и се прехвърля към производствения екип за изпълнение. Едновременно с това, логистичният отдел започва планирането на доставка въз основа на предпочитаната дата и час за получаване от клиента. Следващ етап в този процес е действието на диспечера, ден преди планираната дата, който се свързва с клиента за потвърждаване на поръчката, променяйки статуса ѝ на „потвърдена от клиента“ и по този начин включва продукта в плана за доставка.

В случай че клиентът откаже доставката, поръчката преминава в статус „отхвърлена“. Това автоматично задейства процедура по коригиране на плана за доставки, при което следват промени в звената по логистичната верига, съобразени с нововъзникналите обстоятелства. Подобна ситуация може да доведе до редица проблеми:

- Допълнителни разходи: Пренасочването на доставка към друг клиент може да генерира допълнителни транспортни разходи. Ако отказаната поръчка не може да бъде пренасочена към друг клиент, това може да затрудни дейността на логистичния отдел, както и да доведе до загуби за предприятието.
- Управление на запаси: Върнатите материали или стоки трябва да бъдат правилно обработени, проверени за качество и повторно

въведени в складовите запаси. Например, ако камион, натоварен с бетонова смес, е върнат, сместа от камиона се излива в специален контейнер за повторна обработка. След това технолог проверява и анализира състава, за да се увери в неговото качество.

- Логистични затруднения: Промяната на маршрутите и разписанията на доставките, вследствие на отказани поръчки, може да създаде предпоставки за дезорганизация и закъснения на други доставки.

След потвърждаване на поръчката и възлагане на доставка, диспечерът поддържа постоянен контакт с шофьорите и следи за възможни проблеми: интензивен трафик, задръствания или внезапни промени, поискани от клиента. Подобни обстоятелства изискват незабавна реакция от негова страна. Редовното актуализиране на статуса на поръчките и предоставянето на актуална информация относно очакваното време на доставка са ключови за ефикасното управление на ресурсите.

Последният етап на логистичния процес включва доставката на бетоновата смес до указаното от клиента място и получаването на потвърждение за приемането. Необходимо е подписване на документи на хартиен носител като доказателство, че стоките са доставени в оптимално състояние и в уговорения часови диапазон. При успешна доставка, поръчката трябва ръчно да се отбележи като „изпълнена“ в ERP системата, след което се издава фактура за извършване на плащането.

Настоящият анализ разкрива сложния характер на логистичния процес, свързан с обработката и изпълнението на клиентските заявки. Особено важна е ролята на диспечерите във всеки етап от процеса по доставка и навременната комуникация. Основен недостатък в обслужването е, че диспечерите ръчно отбелязват текущото състояние на поръчка и доставка. Нивото на автоматизация остава ниско. В текущата дейност на фирмата се забелязват някои проблемни области и направления, които могат да бъдат подобрени, което би довело до положителен ефект върху цялостната верига на доставки.

Предложената от нас облачна информационна система за управление на поръчките от клиенти може да бъде внедрена в дейността на „Хейделберг Цимент Девня“ АД, като по този начин да спомогне за решаване на изложените проблеми. Прилагането на ПОСУП, може да доведе до значителни подобрения на дейността на предприятието в следните направления:

- **Приемане на поръчка** – към момента поръчките се получават чрез имейл или телефонно обаждане и се обработват ръчно от диспечери. Това включва приемане на поръчка от клиент за типа бетонова смес, необходимия обем, дата и час за доставка. ПОСУП следва да адаптира тези процеси чрез функции за онлайн регистрация, съгласяване с общите правила и одобрение от диспечера. След получаване на одобрение, потребителите могат да регистрират нови поръчки, както и да променят или отхвърлят съществуващи.
- **График –** след като поръчка е приета, тя трябва да бъде планирана за производство и доставка. Изготвя се график, който определя необходимите ресурси и времеви прозорци за изпълнение на доставка. Според нас, онлайн порталът представлява инструмент за подпомагане на управлението и автоматизацията на посочените задачи, интегрирайки вътрешните подсистеми с облачните услуги. В случай на промяна или отхвърляне на поръчка, системата автоматично пренарежда графика. Чрез автоматизирани известия, всички заинтересовани страни биват информирани.
- **Товарене** – процесът на товарене на бетонова смес и инертни материали започва с подготовка и подбор на подходящо оборудване. Следва зареждането на пясък, чакъл и цимент, както и добавянето на вода в товарен камион или от топ „миксер“. Материалите се смесват до получаване на хомогенна смес, която впоследствие се транспортира до строителната площадка. През цялото време се следи за запазването на качеството ѝ. Подобно на други решения, ПОСУП приема данни за нивото на водата и температурата от IoT сензори, които се изпращат в реално време към облачна

платформа за съхранение и анализ (Oh, Koo, & Kim, 2022). По този начин бизнес клиентите и диспечерите имат възможност да наблюдават показателите непрекъснато, осигурявайки ефективен мониторинг и контрол на процеса.

- **Доставка** – процесът по доставка на бетонова смес и инертни материали чрез облачна информационна система включва проследяване в реално време на камионите. Шофьорите имат възможността да избират оптимални маршрути, използвайки предоставената система, докато клиентите могат да получават достъп до текущото местоположение на превозното средство. В случай на непредвиден инцидент (катастрофа или спукана гума), шофьорите могат да се свържат с отговорните лица на работната площадка, както и да сигнализират диспечерите. След извършване на доставката, клиентите подписват цифрово документите.
- **Фактуриране** – След извършване на доставката, на клиента се издава фактура, в която е отбелязана крайната стойност заедно с начислените данъци и такси. Фактурирането и плащането представляват допълнителни функционалности, които могат да бъдат интегрирани в системата.

Според проучване, „42% от компаниите, използващи облачни услуги“, преместват своите данни обратно на локални сървъри (Shaikh, 2024). Възможно „Хайделберг Цимент Девня“ да изгради частен облак, който да отговаря на специфичните изисквания и нужди на производственото предприятие. Въпреки това, поддръжката на частен облак изисква значителни инвестиции в хардуер и софтуер. Необходими са висококвалифицирани IT специалисти, които да осигуряват непрекъсната работа и сигурност на инфраструктурата. Това повишава оперативните разходи на компанията.

Публичните облачни услуги предлагат ценови планове, базирани на реалното потребление. Това означава, че предприятието плаща само за използваните ресурси и може да увеличава или намалява обема на изчислителните ресурси според променящите се бизнес изисквания. Именно

поради тази причина, публичните облачни услуги се считат за подходящи. Както вече бе отбелязано, те дават възможност за сравнително бързо адаптиране към новите пазарни условия и минимизират финансовия риск, свързан с придобиването на скъпоструваща компютърна техника.

Процесът по внедряване на ПОСУП посредством публични услуги може да се структурира в няколко етапа. Първоначално се извършва анализ на съществуващите подсистеми, отбелязват се необходимите изисквания за облачна интеграция и се оценява доколко настоящата инфраструктура от подсистеми е подготвена за промени. След това се избират подходящите облачни услуги от тип IaaS, PaaS и SaaS, които да удовлетворят специфичните потребности на предприятието. При този подбор се вземат предвид фактори като мащабируемост, сигурност и възможности за интеграция със съществуващите подсистеми (ERP, TMS и CRM).

След избора на подходящи облачни услуги, се пристъпва към изграждане на архитектурата от микроуслуги, създаване на API интерфейси, конфигуриране на базите от данни, инсталиране на уеб портала и публикуване на мобилното приложение в Google Play и Apple Store. Разработването на приложенията следва методологиите и моделите на софтуерна архитектура, описани в предходната глава.

След изграждане на системата се преминава към етап на тестване – откриване на дефекти и възможни проблеми, което трябва да доведе до подобряване на качеството на програмния код и осигуряване надеждността на крайния продукт. На този етап се изпълняват множество видове тестове:

- Функционални тестове – проверяват дали всички функции на приложението работят според спецификациите;
- Тестове на производителността – анализират поведението на системата под различни натоварвания и оценяват способността ѝ да се справя с голям брой потребители или операции;
- Интеграционни тестове – изследват взаимодействието между различните подсистеми и микроуслуги;

- Тестове на сигурността – проверяват за уязвимости.

След успешното преминаване на тестовете, новата система може да се интегрира поетапно в различните отдели на предприятието. Необходимо е да се създадат механизми за поддръжка и редовна актуализация на софтуерните приложения, мониторинг и анализ на производителността на системата, както и автоматизирано архивиране на данни за възстановяване след аварии (Куюмджиев, 2019).

Съпротивата срещу промяна от страна на персонала може да затрудни внедряването на облачната система, особено когато липсва адекватно обучение и подкрепа при използването на софтуера. Поради тази причина е от съществено значение редовно да се провеждат обучения, за да се гарантира, че служителите са запознати с новите функционалности и подобрения (Тодоранова, 2024). Заедно с това, е необходимо да се разработи и поддържа документация.

Концептуалният и логическият модел на ПОСУП, разработени във втора глава, са изградени въз основа на проучване на организацията на работните процеси в „Хейделберг Цимент Девня“ АД. Облачната система дава възможност за интеграция със съществуващите ERP, CRM и TMS подсистеми. При практическа реализация е от съществено значение да се направи обоснована прогноза за необходимата производителност на ПОСУП, да се вземат предвид фактори като брой потребители, брой извършвани от тях действия и брой HTTP заявки, генерирани от клиентските приложения към облачните микроуслуги. Изборът на технологични средства и доставчик на публични облачни услуги трябва да бъде съобразен с тази прогноза.

Ако приемем, че в системата са регистрирани 5000 активни потребители, всеки от които създава или променя средно по 10 поръчки дневно, всяка от които генерира поне по една HTTP заявка към сървърите, то системата би трябвало да има възможности да обработва няколко хиляди заявки на час.

Извличането на данни за доставките в реално време значително

увеличава броя на HTTP заявките, тъй като се генерира поне по една заявка на всяка секунда за всеки потребител. Както бе посочено във втора глава, IoT устройствата, които са свързани директно към системата чрез TCP протокол, натоварват микроуслугите и забавят времето за отговор.

Необходимо е да се предвиди бъдещо нарастване на броя на потребителите и действия, свързани с въвеждането на нови функционалности. Изброените спецификации следва да се вземат под внимание при избора на технологични средства, за да се осигури устойчивост и ефективност при увеличаване на натоварването на системата и разрастване на бизнеса.

3.2. Избор на технологични средства за реализация на системата

За апробирането на облачна информационна система за управление в „Хайделберг Цимент Девня“ АД е необходимо да се подберат подходящи технологични средства за разработка на софтуерните компоненти. Изборът на тези средства следва да бъде резултат от проучване и оценка на различни технологични аспекти, сред които са програмни езици, работни рамки, доставчици на публични облачни услуги и бази от данни. Сравнителен анализ на уеб-базирани работни рамки, направен от софтуерната компания „*TechEmpower*“ през 2023 г. може да се използва при вземането на решение. Резултатите от този анализ са представени в таблица 3.1.

Таблица 3.1

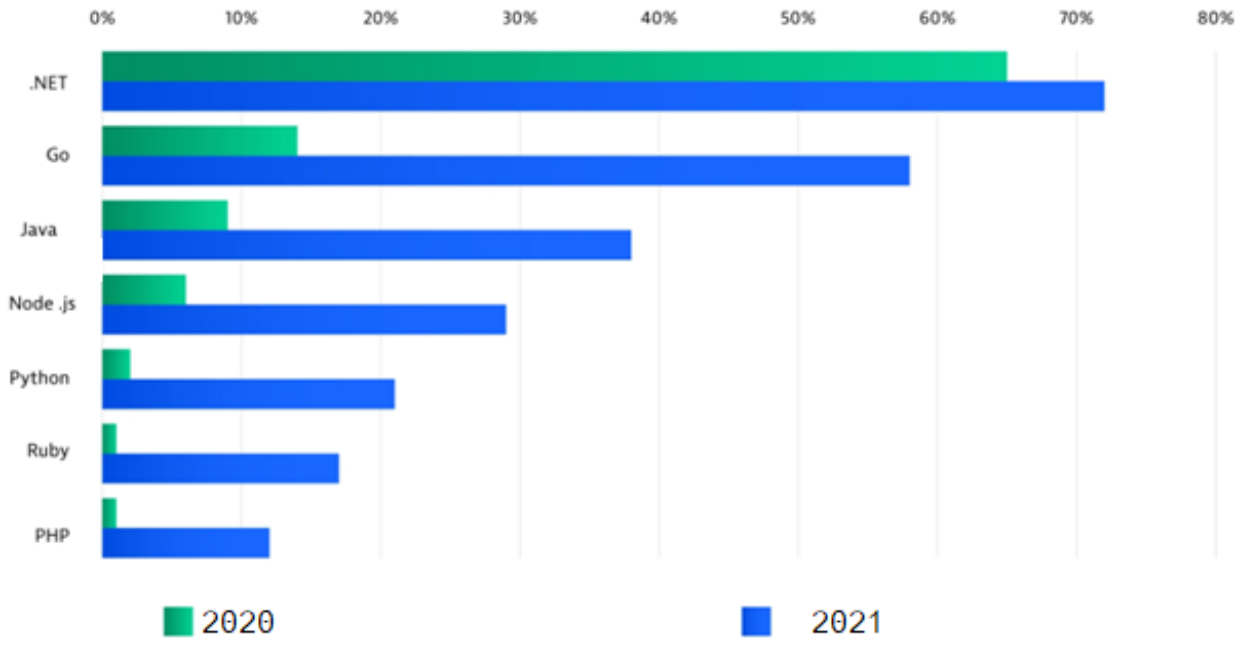
Сравнение на работни рамки за разработка

Сървърна технология	Програмен език	Брой HTTP отговори за секунда
ASP .NET Core	C# / .NET	~ 300 613
NodeJS	JavaScript / C++	~ 125 743
Gin	Go	~ 102 559
Symphony	PHP	~ 70 382
Spring	Java	~ 30 891

Източник: Techempower, 29.09.2023.

TechEmpower дефинират понятието „работна рамка“ като комбинация от сървърна технология и програмен език. В техния сравнителен анализ са разгледани няколко работни рамки, като производителността им се оценява по броя на HTTP заявки, които могат да бъдат обработени за секунда. Според получените данни, ASP.NET показва по-висока производителност в сравнение с останалите.

GitHub предоставя информация за над 5,7 милиона активни разработчици, работещи по проекти с отворен код, които използват езика за програмиране C#, част от технологичната екосистема .NET на Microsoft (Gouigoux, 2024). В доклад на Stack Overflow се отбелязва, че .NET Core е водещата работна рамка за годините 2020 и 2021, както е показано на фиг. 3.1.



Фиг. 3.1. Сравнение на работни рамки във връзка с използването им от проекти и предприятия

Източник: Stack Overflow, 08.05.2023.

Ефективността на ASP.NET Core не се ограничава само до броя обработени HTTP отговори за секунда, но включва и способност за оптимално управление на памет и процесорното време. Чрез .NET Aspire,

Microsoft разкрива набор от инструменти и библиотеки, които включват кеширане, асинхронно програмиране, компилиране на програмния код от тип „just-in-time“ и „ahead of time“. ASP.NET се поддържа на различни операционни системи: Windows, Linux и macOS. Това дава възможност за бързо стартиране и изпълнение на приложенията.

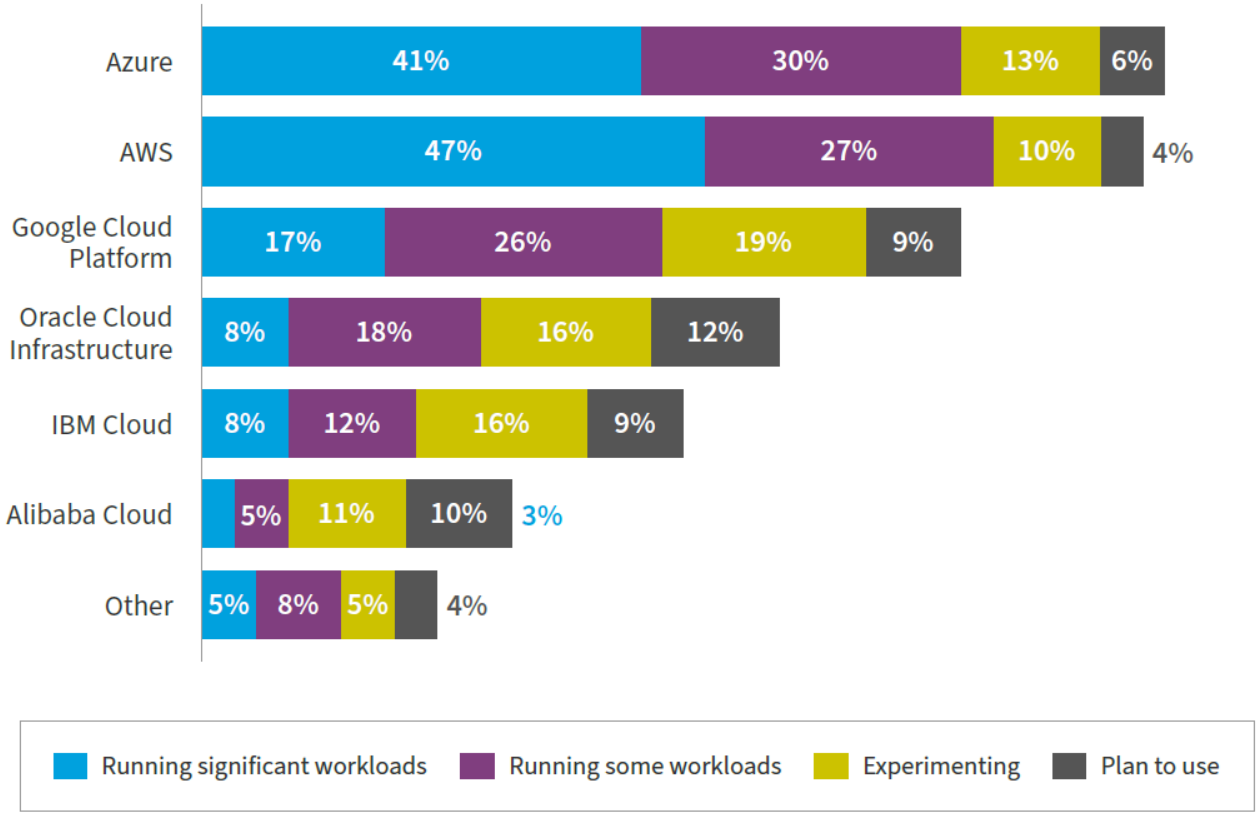
Изборът на ASP.NET Core дава възможност за интеграция с други съществуващи подсистеми и платформи в компанията. .NET Core поддържа широк набор от API интерфейси и пакети за разработка на софтуер (software development kit), позволяващи връзка с корпоративни подсистеми: ERP, CRM и TMS. Това интеграционно предимство следва да доведе до създаването на единна и свързана информационна среда.

.NET предлага възможност за разработка на модели за машинно обучение чрез работната рамка ML .NET. Тази рамка включва множество вградени алгоритми и поддържа интеграция с популярни библиотеки – TensorFlow и ONNX (Mahasivabhattu & Bandi, 2024). С помощта на ML.NET се осигуряват и API-та за интеграция с други продукти на Microsoft, например Outlook, което може да спомогне за реализирането на омниканалния подход, разгледан в т. 1.2 от първа глава.

Използвайки ML .NET, специално разработена микроуслуга може да прихваща и анализира имейли, постъпващи в центъра за диспечери. След като основната информация (като клиентски идентификатор, поръчка, продукт) бъде извлечена, разработени модели за машинно обучение могат да класифицират имейлите в различни категории и да определят ключови действия: създаване, промяна или анулиране на поръчка. Тази информация се препраща към облачната система. Диспечерите имат възможност да прегледат предстоящите промени в уеб портала, както и да потвърдят или коригират информацията преди нейното финализиране. Накрая се генерира автоматичен имейл като потвърждение за направените промени, който съдържа и линк за изтегляне на мобилното приложение.

Microsoft Azure, водещ доставчик на облачни услуги, осигурявщ

обширна поддръжка за всякакъв вид .NET приложения (ASP .NET, Aspire и ML .NET) чрез интегрираната среда за разработка Visual Studio. Според практици в областта (Palermo, 2019; Rendle, 2024), високата степен на интеграция между Azure и .NET значително подобрява процеса на разработка на софтуер и осигурява оперативна съвместимост в рамките на технологичната екосистема, поддържана от Microsoft, което е от съществено значение за успешната реализация на съвременни софтуерни проекти. Фиг. 3.2 представя тенденциите за използване на доставчици на публични облачни услуги в различни предприятия.



Фиг. 3.2. Доставчици на облачни услуги, използвани от предприятия

Източник: Flexera State of the Cloud Report, 03.11.2023.

Констатациите, базирани на извадка от 750 участници, показват, че над 40% от фирмите използват Azure като основна облачна платформа. Около 30% използват Azure частично, 13% експериментират с услугите му, а 6% планират да започнат да го използват в бъдеще. AWS също показва

сходни резултати, докато публичните облачни услуги на Google, Oracle, IBM и Alibaba се намират след водещите в класацията.

Данни от Gartner за 2023 г., представени на фиг. 3.3, показват 47% ръст в облачната инфраструктура и платформени услуги, което затвърждава позицията на Azure на водеща облачна платформа за публични уеб услуги.

Фиг. 3.3. Пазарното позициониране на доставчиците на облачни услуги

Източник: Gartner Magic Quadrant for Cloud Platforms, 01.12.2023.

Както бе отбелязано в началото на трета глава, „Хейделберг Цимент Девня“ АД е дъщерна фирма на компанията Heidelberg Materials, която

оперира в над 50 държави. Ако холдингът вземе решение да използва облачната информационна система във всички (или в повечето) си търговски организации, това би осигурило централизиран достъп до информация, анализ на данни от глобалната верига на доставки и изготвяне на отчети в реално време към борда на директорите. Използването на общ софтуер би уеднаквило вътрешните процеси, което би улеснило сътрудничеството и комуникацията между различните дъщерни фирми (Vasilev & Kehayova-Stoycheva, 2019).

Azure разполага с 64 центъра за данни по целия свят (фиг. 3.4). Те са разположени в различни региони, за да осигурят на микроуслугите максимална производителност и надеждност. Центровете за данни са свързани чрез високоскоростни оптични връзки, което дава възможност за репликация на данни и ресурси между различните локации. Azure използва технологии за мрежова сигурност, за да гарантира целостта и конфиденциалността на информацията, съхранявана и обработвана в тяхната инфраструктура.

Фиг. 3.4. Географски региони на разполагане на центровете на Azure

Източник: De La Torre, 2024.

Въз основа на проведените анализи, може да се заключи, че изборът

на .NET и Azure за изграждане на ПОСУП е рационално решение, което осигурява висока производителност и модулност. С оглед на този избор е необходимо да се отбележи, че различните микроуслуги, разработени за информационната система, имат специфични изисквания за съхранение на данните. В Azure има няколко различни вида хранилища за данни. В таблица 3.2 са представени съответствията между различните облачни услуги и характеристиките на съхранените данни. Хранилищата за данни включват релационни и NoSQL бази от данни, файлови системи и оптимизирани за „големи данни“ услуги.

Таблица 3.2

Сравнителен анализ на услуги за данни с оглед на тяхната структура и характеристики

	Database	Cosmos DB	Blob	Table	File	PostgreSQL, MySQL	SQL Data Warehouse	Data Lake Store
Relational data	✓					✓	✓	✓
Unstructured data		✓	✓					✓
Semistructured data		✓		✓				✓
Files on disk					✓			
Store large data		✓	✓		✓	✓	✓	✓
Store small data	✓	✓	✓	✓	✓	✓		
Geographic data replication	✓	✓	✓	✓	✓	✓		
Tunable data consistency		✓						

Източник: De La Torre, 2024.

Във връзка с проучването на посочените хранилища за данни, е

необходимо да се анализират техните характеристики и да се направи оценка на предимствата и недостатъците им. По този начин можем да определим кои от тях са подходящи бази от данни, които да поддържат разнообразните нужди на микроуслугите в ПОСУП:

- Azure SQL Database – облачна услуга за релационни бази от данни, която предлага вградени механизми за висока производителност, автоматизирани актуализации и архивиране, както и защита на данните чрез криптиране. Тя поддържа пълна съвместимост с Microsoft SQL Server, което улеснява локалната разработка, давайки възможност на програмистите да използват SQL Server на личните си компютри за тестване на приложения. Услугата предоставя и автоматично настройване на обема използвана памет, репликация на данните в различни географски региони, маскиране на данни за определени потребители, както и одит на всички дейности, извършени върху данните.
- Azure Cosmos DB – NoSQL база от данни, предназначена за разработка на приложения, изискващи висока производителност. Тя поддържа множество модели на данни: граф, документ, ключ-стойност. Основни характеристики на Cosmos DB включват ниска латентност, по-малко от 10 милисекунди за четене и запис, както и гарантира 99.999% SLA. Базата работи на високо ниво на надеждност чрез автоматично репликиране на данни между различни региони. Cosmos DB разполага с функции за автоматично индексиране, което се адаптира динамично спрямо структурата на съхранените данни.
- Azure Blob Storage – хранилище за данни, оптимизирано за съхранение и извличане на огромни обеми от неструктурирани данни, които могат да бъдат текстови или двоични данни. Тази услуга е предназначена за различни сценарии, включително архивиране и възстановяване на данни от други бази, например Azure SQL Database и Cosmos DB, както и за файлове като документи, снимки или видео клипове. Azure Blob Storage е организиран в три отделни нива за съхранение, всяко от които подпомага оптимизацията

на разходите според необходимостта от достъп. Услугата поддържа RESTful API интерфейси, които улесняват интеграцията с микроуслуги.

- Azure SQL Data Warehouse (преименувана на Azure Synapse Analytics) е облачна услуга за съхранение и обработка на големи данни, оптимизирана за изпълнение на аналитични работни натоварвания. Тази услуга дава възможност на бизнеса да анализира големи масиви от данни и да извлича ценна информация от тях. Azure Data Lake Storage представлява подобна облачна услуга, насочена към съхранение на неструктурирани данни в тяхната първоначална форма. Тя поддържа висока степен на интеграция с разнообразни аналитични услуги и инструменти, включително и Azure Synapse Analytics.

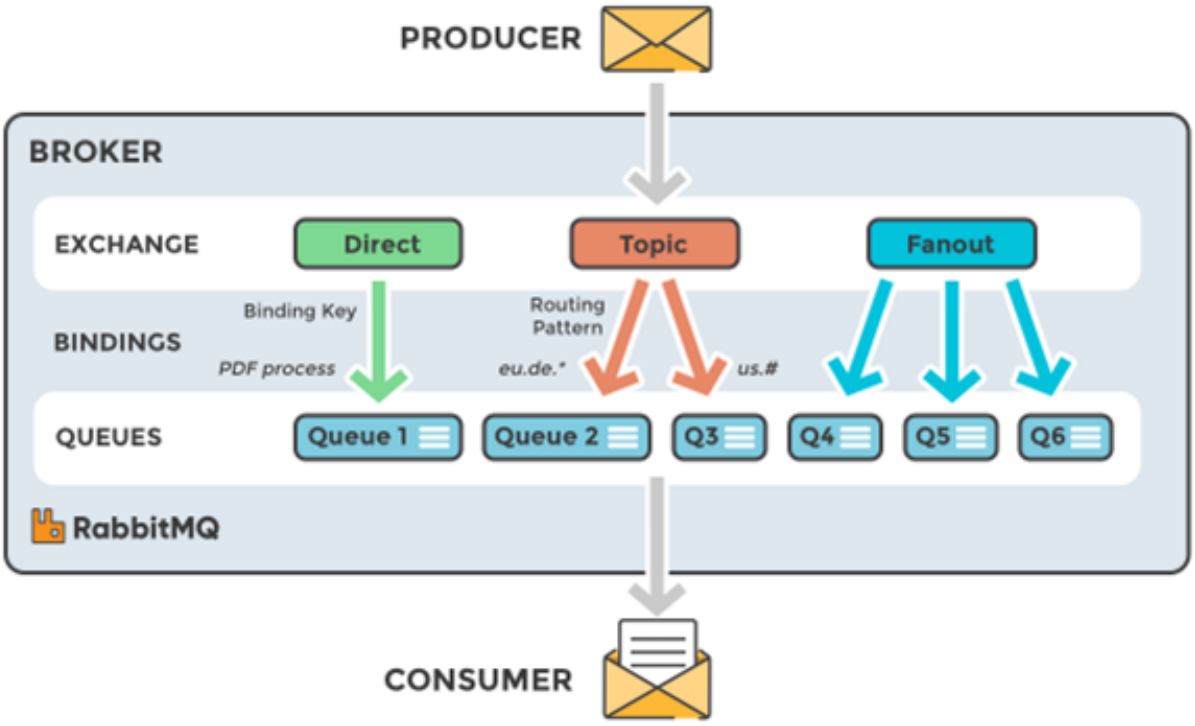
За нуждите на микроуслугите в ПОСУП могат да се използват Azure SQL Database, Azure Cosmos DB и Azure Blob Storage. Микроуслугите за приемане на съобщения, команди и извличане на информация, описани в т. 2.1 на втора глава, могат да разчитат на Azure Cosmos DB, осигурявайки съвместимост между четирите бази от данни за четене и запис на обекти, свързани с поръчки и доставки. Тази комбинация дава възможност за създаването на т.нар. „материализирани изгледи“, които обединяват данни за поръчки и доставки в една заявка, базирана на две бази за четене.

Микроуслугата, която отговаря за управлението на потребителските профили, представена в т. 2.3, може да използва Azure SQL Database заедно със софтуерните пакети ASP.NET Core Identity. Данните за потребители, роли и токени се съхраняват в релационна база от данни, а Identity пакетите предоставят алгоритми за хеширане на пароли, управление на роли и свързване с външни доставчици на идентичности.

Услугата Azure Blob Storage може да служи като място за съхранение на архивирани данни от Azure SQL Database и Cosmos DB. При интеграция с платежната система се дава възможност за съхранение на фактури.

Съгласно проучването във втора глава, следва да се направи избор за технология, работеща като брокер на съобщения. Това следва да е софтуер от

тип SaaS, който дава възможност за асинхронна комуникация между различни микроуслуги чрез прилагане на протокола AMQP. RabbitMQ е софтуерно приложение, което изпълнява ролята на брокер на съобщения. Представлява проект с отворен код, който е съвместим с различни операционни системи и сървърни технологии. Приложенията могат да установяват асинхронна комуникация чрез обмен на съобщения чрез т.нар. „опашки“. Фигура 3.5 представя вътрешните компоненти, които поддържат работата на брокера, включително опашка, модел на маршрутизиране и ключове за обвързване.



Фиг. 3.5. Схема на RabbitMQ

Източник: Ćatović et al., 2022.

Изборът на технологични средства за реализация на мобилните приложения и уеб портала е предизвикателство, тъй като мобилните и уеб технологии се развиват постоянно (Сълова, Банков & Стоянова, 2024). Тази динамика обуславя наличието на разнообразни програмни езици за създаване и поддържане на подобни приложения. Например, Objective-C и Swift са

специално предназначени за устройства с iOS операционна система, докато Java и Kotlin са основните езици за разработка на Android приложения. Съществуват и множество други технологии, които работят на различни платформи: iOS, Android, HarmonyOS и Tizen. При отчитане на тези обстоятелства, табл. 3.3 представя сравнение между различните технологии.

Таблица 3.3

Сравнение между технологии за мобилна и уеб разработка

	Нейтив приложения (native)	**Хибридни приложения**	**Прогресивни уеб приложения (PWA)**	**Междуплатформени приложения (cross platform)**
Програмни езици и инструменти за разработка	**iOS** – Objective-C или Swift чрез X-Code & iOS SDK **Android** – Java или Kotlin чрез Android Studio	HTML, JavaScript, CSS, Cordova, Onsen	Работни рамки като Ionic с Angular или Vue и Blazor със C#	NET MAUI, Kotlin Multiplatform, React Native
Достъп до функциите на смартфон	Пълен контрол, без ограничения	Слаб контрол, известни ограничения	Слаб контрол, доста ограничения	Характеризира се с ограничена, но развиваща се поддръжка от страна на софтуерните компании
Потребителско изживяване (UX)	Отлично	Добро, но ограничено	Добро, но ограничено	Добро

Разработка на автора

Нейтив приложения (native applications) се разработват специално за конкретни операционни системи с помощта на определени програмни езици и интеграционни инструменти: X-Code и Android Studio. Приложенията имат пълен достъп до функциите на устройството – камера, GPS и различни сензори. ***Хибридните приложения***, от друга страна, използват т.нар. уеб изглед (от английски – web view), в който се вгражда уеб съдържание, разработено чрез HTML, JavaScript/TypeScript и CSS, с помощта на

платформи Cordova и Onsen. Целта е приложенията да изглеждат като „нейтив“ за крайните потребители, но на практика те представляват браузър с допълнителни възможности и взаимодействия. Поради това, хибридните приложения често имат ограничения по отношение на достъпа до функциите на устройството.

Прогресивните уеб приложения (от английски – progressive web apps), подобно на хибридните приложения, използват работни рамки: Ionic с Angular или Vue и Blazor със C# (Sulov, 2024). Приложенията функционират като уеб сайтове, но включват допълнителни функционалности, които им дават възможност да работят офлайн и да се инсталират на мобилни или десктоп устройства. Достъпът им до функциите на смартфона или компютъра е ограничен, но имат потенциал за усъвършенстване в бъдеще. Според нас, прогресивните уеб приложения са подходящи за реализацията на уеб портала.

Междуплатформените приложения (от английски – cross-platform), разработени с технологии .NET MAUI, Kotlin Multiplatform и React Native, могат да функционират на различни операционни системи (Stonis, 2024). Чрез тях разработчиците създават единична база от програмен код, която се адаптира към множество платформи, минимизирайки необходимостта от поддържане на отделни версии на приложението за различни операционни системи: Android, iOS, HarmonyOS и Tizen. Тези технологии имат възможност за интеграция с „нейтив“ елементи и функционалности, което подобрява потребителското изживяване. Въпреки че имат известни ограничения по отношение на достъпа до хардуера на мобилните устройства, с нарастващото им развитие и общността около тях, ограниченията постепенно намаляват. Това прави междуплатформените приложения подходящ избор за реализацията на мобилно приложение.

В заключение, за реализацията на облачната система са избрани следните технологични средства:

- ASP .NET Core за сървърната част и микроуслугите;

- Azure като доставчик на публични облачни услуги;
- Azure SQL Database, Cosmos DB и Blob Storage за бази от данни;
- RabbitMQ като брокер на съобщения;
- Microsoft Blazor за технология за уеб портала;
- .NET MAUI за разработка на мобилното приложение.

Интеграцията на различни технологии осигурява надеждност и мащабируемост в ПОСУП. Благодарение на координацията в технологичната екосистема на Microsoft, процесите по разработка и поддръжка се опростяват, което води до съвместимост и взаимовръзка между отделните компоненти на системата (back-end и front-end).

3.3. Физическа реализация на системата

След направения избор за Microsoft Azure като доставчик на облачни услуги и на технологиите на .NET за реализация на клиентските приложения и микроуслугите, следва да се направи избор за хостинг услуги. Хостингът е от съществено значение за физическата реализация и интеграция на информационната система в инфраструктурата на Azure, тъй като хостингът определя както началните, така и дългосрочните параметри за развитие.

При избора на подходящи хостинг услуги е необходимо да се вземат предвид следните фактори:

- Възможност за динамично мащабиране: хостинг услугите трябва да дават възможност за увеличаване или намаляване на изчислителни ресурси в зависимост от текущите нужди;
- Контейнеризация и оркестрация: хостинг услугите следва да поддържат разгръщане на нови версии на микроуслугите чрез Docker контейнери и управление на внедряването чрез инструмент за оркестрация;
- Практики за непрекъсната интеграция и доставка: използването на подобни практики осигурява автоматизирано компилиране на

програмния код, тестване и създаване на контейнер, който да се внедри в хостинг услугата.

Таблица 3.4 предоставя информация за основните хостинг услуги на Azure и съответстващите им случаи на употреба.

Таблица 3.4

Услуги на Azure за хостинг и съответстващи случаи на употреба

	App Service Web Apps	App Service Mobile Apps	Azure Functions	Logic Apps	Virtual Machines	Azure Kubernetes Service (AKS)	Container Instances
Monolithic and N-Tier applications	✓				✓*		✓
Mobile app back end		✓			✓*		
Microservice architecture-based applications			✓			✓	
Business process orchestrations and workflows			✓	✓			

Източник: Moniz et al., 2021.

Azure App Services е един от вариантите за хостване на приложения, който е подходящ за използване при монолитна архитектура. Хостингът е сравнително надежден – според SLA работи в 99,95% от времето. Тази PaaS услуга предоставя възможности за автоматично мащабиране, актуализиране на нови версии, без прекъсване на работещи услуги и удостоверяване чрез централен доставчик на идентичност (т.2.3). Услугата осигурява механизъм за диагностика и отстраняване на грешки в приложението в реална производствена среда с помощта на инструмента Snapshot Debugger. По подразбиране, хостваното приложение е достъпно в Интернет, без

необходимост от настройка на домейн или конфигуриране на DNS.

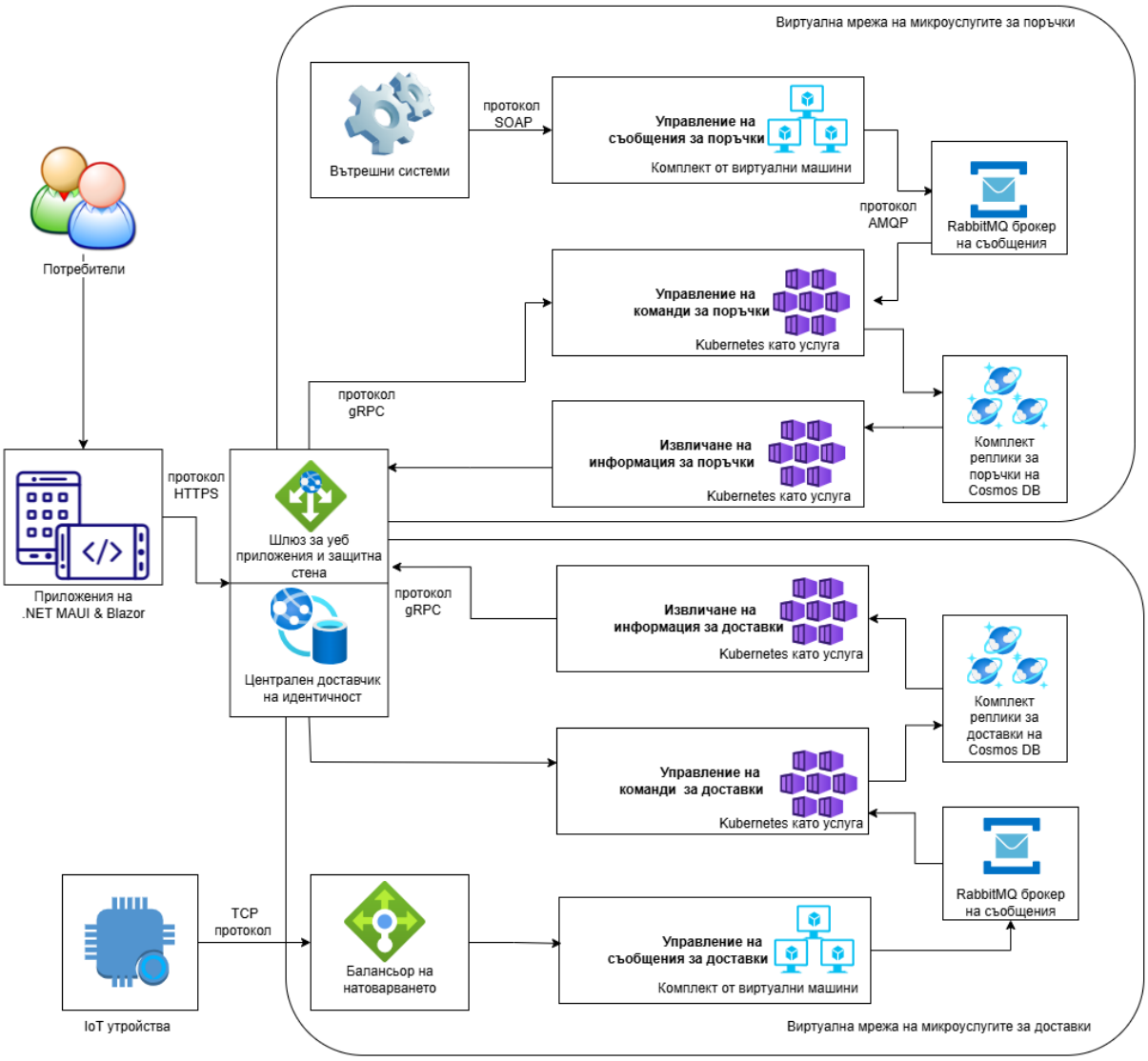
Друг вариант за избор, с подобни на App Services функции, но от тип IaaS, е услугата ***Azure Virtual Machines***. Тя поддържа набор от протоколи за обмен на данни, включително TCP и SOAP, и дава възможност за преместване на съществуващи приложения от други виртуални машини. Услугите поддържат ISO файлове с изображения, например Windows Server, който работи с IIS и има инсталиран и предварително конфигуриран ASP.NET, както и собствени софтуерни лицензи (като например за SQL Server). Azure Virtual Machines е подходяща услуга за приложения, които се свързват с множество IoT устройства в реално време, чрез TCP протокол.

Azure Kubernetes Service (AKS) е облачна услуга от тип PaaS, предназначена за автоматизиране на внедряването и управлението на микроуслуги. Проектирана с висока степен на модулност, тя намалява необходимостта от участие на разработчици и ИТ специалисти в стандартните процеси за хостинг на приложения. Сигурността в AKS се разглежда чрез многопластов подход, който защитава както инфраструктурата, така и самите приложения. Микроуслугите са организирани във вътрешни обекти (от английски – pods), които представляват най-малките изпълними единици. Те са групирани в клъстери от изчислителни машини. Основни компоненти на AKS включват: интерфейс (kube-apiserver) за приемане на команди за внедряване на микроуслуги, хранилище от тип „ключ-стойност“ (etcd) за съхранение на конфигурационни данни, както и компоненти, които следят мрежовия трафик към кластера и автоматично коригират броя на работещите микроуслуги. Компонентите работят съвместно, за да предоставят надеждност, устойчивост и висока производителност. По този начин се оптимизират разходите, следят се внезапни пикове в трафика, необичаен растеж или спад в използването.

За разлика от Azure App Services и Azure Virtual Machines, AKS се отличава с висока степен на автоматизация, баланс на натоварването и възможности за самовъзстановяване след непредвидени инциденти. Според

проучване на CNCF (2023) сред 725 софтуерни компании, 64% от тях използват Kubernetes в производствени среди, докато 25% експериментират с тази технология. На базата на изнесените данни, считаме, че AKS е подходяща услуга за хостинг на микроуслугите.

Въз основа на разгледаните софтуерни технологии и инструменти, на фиг. 3.6 е представена архитектурна диаграма, която съответства на концептуалния, логически и комуникационен модел, както и на основните компоненти на облачна услуга за управление, описани в т. 2.1.



Фиг. 3.6. Архитектурна диаграма на софтуерните технологии, изграждащи облачната система

Разработка на автора

Съхранението и поддръжката на програмния код представляват следващ етап от физическата реализация на информационната система. За съвместна работа по изпълнение на задачи, екипите от програмисти използват платформи за софтуерно разработване и контрол на версиите на приложенията, най-често базирани на Git[19].

Подобни платформи работят с уеб-базиран интерфейс, чрез който се поддържа създаването и управлението на т.нар. „хранилища за програмен код“. Управлението на промените в кода е от съществено значение за информационните системи поради необходимостта от постоянни актуализации, подобрения във функционалностите и интеграция на нови търговски организации. За тази цел разработчиците могат да използват различни платформи – GitHub, GitLab, Bitbucket и Azure DevOps. Въпреки че основната цел на всички изброени платформи е поддръжката на програмния код, техните характеристики се различават. За сравнение, в табл. 3.5 са описани някои от техните предимства и недостатъци.

Таблица 3.5

Сравнение на уеб базирани платформи за софтуерно разработване и контрол на версиите

	GitHub	**GitLab**	**Bitbucket**	**Azure DevOps**
Система за контрол	Git	Git	Git/Mercurial	Git/CVS/ Perforce
Непрекъсната интеграция и доставка	GitHub Actions	GitLab CI/CD, Auto DevOps	Bitbucket Pipelines	Azure Pipelines
Управление на проекти	Вградени проекти Kanban табла	Вградени проекти Kanban табла	Интеграция с Jira, интеграция с Trello, Bitbucket Issues	Azure Boards
Потребителски интерфейс	Интуитивен и лесен за използване	Обширен, но сложен за използване	Функционален, но не интуитивен	Обширен, но сложен за използване
Документация	GitHub Pages	GitLab Pages	Markdown	Azure DevOps Docs

[19] Git е система за проследяване на промените в изходния код на софтуерни проекти, създадена през 2005 година. Всеки разработчик разполага с копие на проекта и историята на промените.

	GitHub	GitLab	Bitbucket	Azure DevOps
Общност	Най-голяма общност	Голяма, но предназначена за корпорации	Средна, поддържана от Atlassian	Средна, поддържана от Microsoft

Разработка на автора

От изброените по-горе, GitHub се отличава с поддръжката на различни инструменти за непрекъсната интеграция и доставка на софтуер: GitHub Actions, Jenkins, CircleCI и Travis CI. GitHub поддържа редица механизми за подобряване на сигурността на кода, включително проверки за уязвимости (dependency vulnerabilities), инструменти за сканиране на пароли (secret scanning), автоматизирани процеси за обновяване на софтуерните пакети (dependabot) и интеграция с външни приложения (GitHub Apps), които да следят и докладват за грешки или възможни подобрения в кода.

Интуитивният потребителски интерфейс улеснява сътрудничеството между разработчиците чрез функционалности за известия за промени в кода (pull requests) и управление на проблеми (issues). GitHub поддържа и среда за разработка (codespaces), която има предварително конфигурирани виртуални машини с всички необходими инструменти и настройки за започване на работа по проект. По този начин разработчиците могат да използват облачни услуги, без да е необходимо да конфигурират локално своите компютри.

През 2022 г. GitHub и OpenAI представят Copilot – инструмент, който използва изкуствен интелект за подпомагане на разработчиците при писане на програмен код. Интегриран директно в редактори Visual Studio и Codespaces, Copilot осигурява възможност за автоматично генериране на цели класове и функции, въз основа на коментарите в текущия файл. Copilot може да предложи фрагменти от програмен код, които са в съответствие с актуалните стандарти и добри практики, което да ускори процеса на разработка.

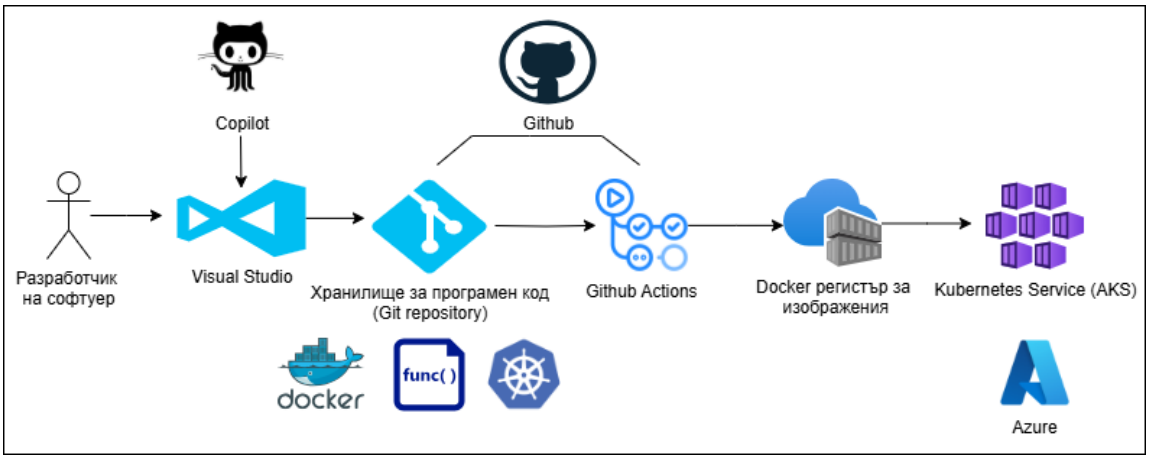
Това дава основа ***GitHub*** да бъде използван като платформа за съхранение и поддържане на кода на информационната система. Публичното хранилище с програмния код на облачната система, е достъпно на следния

адрес: https://github.com/profjordanov/cloud-based-management-information-system.

При практическата реализация и поддръжка на програмния код е необходимо да се дефинира ролята на виртуализацията в експлоатацията на информационната система, особено по отношение на контейнеризацията (Иванов, 2022). Изследването в т. 1.3 на първа глава показва, че виртуализацията на хардуера, софтуерните услуги и контейнеризацията на микроуслуги са основни елементи на съвременните облачни системи. Виртуализацията на хардуера дава възможност множество „виртуални машини“ да работят на един физически сървър, което увеличава използваемостта на наличните ресурси. Софтуерната виртуализация създава изолирани среди за приложения, повишавайки тяхната сигурност. Контейнеризацията на микроуслуги е метод, който улеснява тяхното внедряване и управление. Всяка микроуслуга е „капсулирана“ в свой собствен контейнер, който бива интегриран в AKS. AKS работи съвместно с разнообразни услуги и технологии на Azure и Microsoft, включително Azure SQL Server, Cosmos DB, Blob Storage, ASP.NET Core, Blazor, .NET MAUI и RabbitMQ.

Виртуализацията и контейнеризацията подпомагат практиките по „развитие и операции” (DevOps), които автоматизират разработката и внедряването на облачни услуги чрез технологии, като Docker, GitHub Actions и AKS. Както е известно, основната цел на DevOps е да се изгради интегрирана среда, в която рутинните дейности по изграждане, тестване и актуализиране на информационната система се осъществяват по предварително изготвен план, при което се дава възможност за непрекъсната доставка и внедряване. Допълнително се осигурява високо качество на софтуера и се намалява времето за реализация на нови функционалности, тъй като всяка микроуслуга, мобилно и уеб приложение, преминават през изолиран работен поток (фиг. 3.7), който включва етапи по компилиране и компонентно тестване на програмния код, изграждане и внедряване на

Docker контейнер в AKS.



Фиг. 3.7. Работен поток, съчетаващ процесите на разработване и актуализиране на облачни услуги

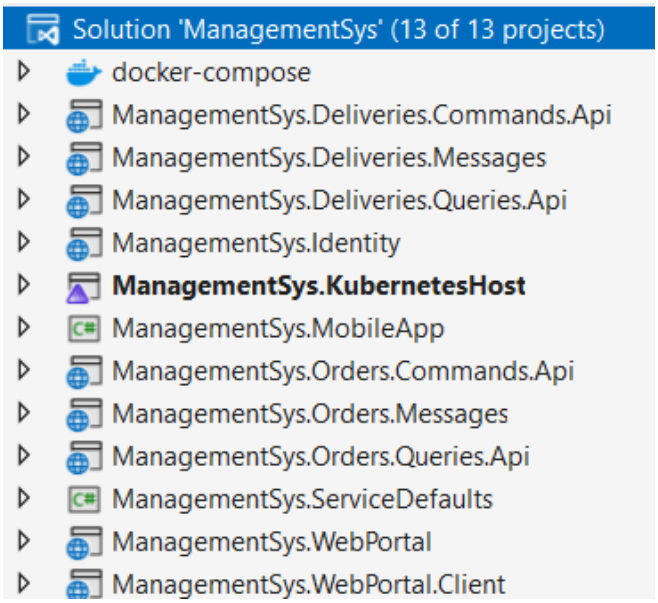
Разработка на автора

Работният поток, представен на фиг. 3.7, е разделен на няколко етапа. В първия етап разработчик създава или модифицира програмен код в среда за разработка Visual Studio или GitHub Codespaces, с помощта на Copilot. След това, програмният код, заедно с инструкциите за Docker и Kubernetes, се качват в GitHub. Третият етап е насочен към непрекъсната интеграция, в който автоматизирани процеси на GitHub Actions анализират кода за уязвимости или допуснати грешки при компилиране (Oukes et al., 2021). След това се изпълняват компонентните тестове и се създава изображение на контейнер, което се съхранява в регистър, функциониращ като централизирано хранилище за съхранение и администриране на Docker изображения.

Ако някой от процесите на GitHub Actions завърши неуспешно, например някои от компонентните тестове дадат невалиден резултат, работният поток прекъсва в този момент. Ако всичко мине успешно, в последния етап, свързан с „доставка на софтуера“, GitHub Actions автоматизира внедряването на контейнерното изображение в AKS.

Програмните класове в ПОСУП, интегрирани в средата за разработка

Visual Studio, са представени на фиг. 3.8. То е организирано във вид на подпроекти и следва диаграма на класовете и техните връзки от раздел 2.2.



Фиг. 3.8. Структура на подпроектите във Visual Studio

Разработка на автора

3.4. Приложение на системата чрез технологичните средства за реализация

След началния анализ на дейността на „Хейделберг Цимент Девня“ АД и настоящите бизнес процеси по приемане на поръчки и доставка на готова продукция, бяха установени възможности за рационализация, автоматизация и оптимизация. В следващ етап бе извършен избор на подходящи технологични средства, които да позволят интеграция на облачната система със съществуващите подсистеми на предприятието. Като финален етап следва да се проведат тестове на функционалностите, микроуслугите и клиентските приложения на облачната система.

3.4.1. Тестване на облачната система

След изграждането на системата е необходимо да се проведе

изпитателен период с продължителност 1-2 месеца. Целта на този период е да оцени приложимостта на предложената концепция за персонализирана система чрез тестове на нейната функционалност. В хода на този период могат да се установят и коригират потенциални проблеми или пропуски, което да осигури надеждното функциониране на системата.

При тестването на облачната система, базирана на микроуслуги, могат да се използват стратегии, които да определят ползите от разработката и внедряването чрез оценка дали системата функционира според очакванията на потребителите. По този начин може да се подпомогнат мерките за предотвратяване на софтуерни дефекти и подобряване на устойчивостта на системата. Описание на някои от тези стратегии е представено в таблица 3.6.

Таблица 3.6

Стратегии за тестване на облачна система

Стратегия за внедряване	Описание
Синьо-зелено внедряване (blue-green deployment)	Синьо-зеленото внедряване позволява едновременното изпълнение на две идентични производствени среди. „Синята“ представлява активната, докато „зелената“ обозначава новата версия. Идеята на този подход е да се изпробва нова версия в среда, подобна на производствена, без да се прекъсва активната услуга.
Постепенно внедряване (rolling deployment)	При „постепенното внедряване“ на новата версия, приложението се актуализира поетапно, като се разгръщат няколко броя Docker контейнери. Докато определен брой услуги продължават да работят със старата версия, при възникване на проблем процесът на внедряване може да бъде спрян. По този начин се подпомага локализирането и отстраняването на проблема.
Внедряване на Canary (Canary release)	При „внедряване на Canary“, малка част от потребителите или сървърите се пренасочват към нова версия на приложението, докато останалите продължават да работят със старата версия. Това позволява евентуални проблеми да бъдат отстранени, без прекъсване на цялата система. Ако възникнат непредвидени грешки, процесът може да бъде временно спрян или да се върне към предишната версия.

A/B тестване	A/B тестването представлява двувариантен експериментален метод за оценка на различни подобрения или промени. В този процес потребителският трафик се разделя между две версии – версия „A“ (контролна) и версия „B“ (тестова), като се измерват предварително дефинирани параметри. Това позволява вземане на обосновани решения за оптимизации, намалява риска от неуспешни промени и допринася за по-добра производителност.
Chaos Engineering	Chaos Engineering е методологичен подход за изследване на устойчивостта на системата чрез умишлено предизвикване на грешки, в контролирана среда. Неговата основна цел е своевременно да се отстранят проблеми, които биха останали скрити при нормална работа. Този подход обхваща симулиране на неочаквани повреди в микроуслуги или ограничаване на мрежови ресурси, за да се прецени как цялостната системата реагира. По този начин екипите от разработчици могат да тестват аварийни сценарии преди реалното им възникване и да предприемат мерки за предотвратяване на сривове.

Разработка на автора

За целите на апробирането на облачната информационна система за управление на поръчки, приемаме стратегията за ***A/B тестване*** като подходяща. Тази стратегия предоставя възможност за въвеждане на нови функционалности и наблюдение на резултатите в реално време.

За реализиране на стратегията се използват ръчни и автоматизирани тестови процедури, които симулират потребителското поведение във временно създадена облачна среда (виж приложение 3). Ръчните тестове дават възможност за проверка на специфични функционалности и оценка на потребителския интерфейс, докато автоматизираните тестове регистрират и модифицират данни чрез крайните точки на микроуслугите. Тестовите процедури са разделени на две групи:

- **Група A** – симулира използването на системата от страна на бизнес клиенти;
- **Група B** – симулира използването от страна на доставчик.

Основната цел на A/B тестването е да се оцени приложимостта на системата. За да постигнем тази цел, е необходимо да се създадат, променят и отхвърлят множество поръчки за продажба, които да бъдат отразени в SAP ERP подсистемата на предприятието. Допълнителни задачи включват събиране на входни данни за местоположението (GPS координати) на превозните средства за доставка и извличане на данни от IoT устройства.

Както бе отбелязано, провеждането на A/B тестовете се осъществява в симулационна среда, която имитира реалните условия на работа на системата. Тестовите процедури обхващат различни сценарии, за да се осигури цялостна оценка на функционалността и сигурността в ПОСУП. Тестовете от група А са насочени към управлението на информацията за поръчки, докато тези от група В – към информацията за доставки.

След провеждане на необходимите тестове и анализ на резултатите, получените данни са представени в табл. 3.7 и табл. 3.8, където са описани:

- ID на поръчка – автоматично генериран уникален идентификатор за всяка поръчка;
- Дата на поръчката – датата, на която е направена поръчката чрез мобилното приложение;
- Дата на предпочитана доставка;
- Състояние на доставката – текущо (актуално) състояние на изпълнението на доставката (напр. доставена, предстояща);
- GPS координати – данни за текущото местоположение;
- IoT данни – данни от сензорите, свързани с температурата и нивото на водата.

Таблица 3.7

Резултати от A/B тестове при създаване на поръчки и доставки

ID	Дата на поръчка	Дата на доставка	Състояние	Координати	IoT
12314	1-Mar-24	3-Mar-24	Delivered	40.7128° N, 74.0060° W	Temp: 25°C, Level: 60%
32262	4-Mar-24	6-Mar-24	In Transit	34.0522° N, 118.2437° W	Temp: 22°C, Level: 55%
23123	7-Mar-24	9-Mar-24	Delivered	41.8781° N, 87.6298° W	Temp: 20°C, Level: 65%
33434	10-Mar-24	12-Mar-24	Pending	29.7604° N, 95.3698° W	Temp: 28°C, Level: 50%
90905	13-Mar-24	15-Mar-24	Delivered	33.4484° N, 112.0740° W	Temp: 30°C, Level: 45%
66786	16-Mar-24	18-Mar-24	In Transit	39.7392° N, 104.9903° W	Temp: 18°C, Level: 55%
90867	19-Mar-24	21-Mar-24	Delivered	32.7767° N, 96.7970° W	Temp: 25°C, Level: 60%
34248	22-Mar-24	24-Mar-24	Pending	37.7749° N, 122.4194° W	Temp: 20°C, Level: 70%
23129	25-Mar-24	27-Mar-24	Delivered	47.6062° N, 122.3321° W	Temp: 15°C, Level: 75%
31210	28-Mar-24	30-Mar-24	In Transit	25.7617° N, 80.1918° W	Temp: 27°C, Level: 65%

Всички данни, посочени в табл. 3.7, са налични в SAP ERP подсистемата на предприятието. Това представя взаимодействието между корпоративните подсистеми и ПОСУП (виж приложение 4).

В допълнение, табл. 3.8 илюстрира промените в някои от поръчките и доставките, които също така са отразени във вътрешните подсистеми.

Таблица 3.8

Резултати от A/B тестове при промяна на поръчки и доставки

ID	Дата на поръчка	Дата на доставка	Състояние	Координати	IoT	Направени промени
12314	1-Mar-24	~~3-Mar-24~~ 5-Mar-24	Delivered	40.7128° N, 74.0060° W	Temp: 25°C, Level: 60%	Промяна в дата на доставка
32262	4-Mar-24	6-Mar-24	In Transit	34.0522° N, 118.243° W	Temp: ~~22°C~~, 25°C Level: 55%	Промяна в температурата чрез IoT съобщение
23123	7-Mar-24	9-Mar-24	Delivered	~~41.8781° N, 87.6298° W~~ 56.7532° N, 96.4615° W	Temp: 20°C, Level: 65%	Промяна в координатите
33434	10-Mar-24	12-Mar-24	~~Pending~~ Cancelled	29.7604° N, 95.3698° W	Temp: 28°C, Level: 50%	Промяна в състоянието

3.4.2. Системен мониторинг

След внедряването на облачната система е необходимо непрекъснато наблюдение на клиентските и сървърни приложения. Някои стратегии за наблюдение включват:

- Мониторинг в реално време – интегриране на системи за наблюдение, които дават възможност за проследяване на системните ресурси, времето за отговор на микроуслугите и натоварването;
- Създаване на индикатори за производителност – определени на база броя приети поръчки през системата. Осигуряват на

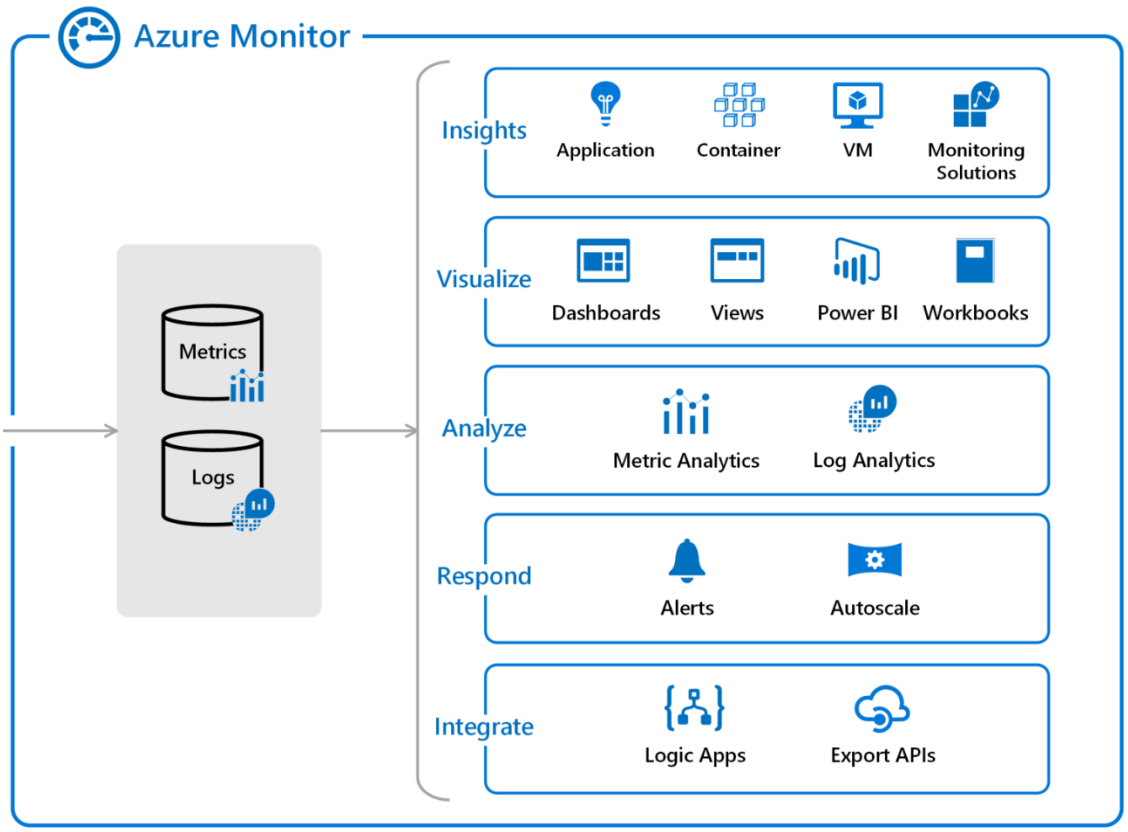
ръководството информация за капацитета на ПОСУП и улесняват процеса по усъвършенстване;

- Поддържане на системен дневник – насочен към ИТ специалисти, този подход подпомага проследяването на грешки и разбирането на последователността от събития, довели до неизправност, както и улеснява системния анализ и одита.

Мониторингът, интегриран в облачната платформа Azure, предоставя инструменти за наблюдение, водене на системен дневник и проследяване на индикатори за производителност, които могат да бъдат използвани от разработчици, бизнес специалисти и мениджъри. Мониторингът се разделя на две основни категории: мониторинг на инфраструктурата и мониторинг на приложенията. В първата категория се включват оценка и контрол на системните ресурси (процесор, памет, дисково пространство и мрежов трафик), а във втората – наблюдение на функционалността и ефективността на отделните услуги, обхващайки параметри: време за реакция, честота на грешки и проследяване на транзакции между микроуслугите и ERP.

Обобщен модел на Azure Monitor, предлагащ услуги за диагностика в реално време, е представен на фиг. 3.9.

Една от основните дейности на мониторинга се изпълнява от системата за предупреждения, която се активира при отклонение на специфични показатели и индикатори от нормалния им диапазон. Например, такива отклонения могат да включват ниво на използване на процесора над 90% или липса на създадена поръчка в рамките на последните 24 часа. При откриване на отклонение, системата уведомява съответна група специалисти (ИТ или бизнес експерти), които трябва да предприемат действия за отстраняване на проблема.

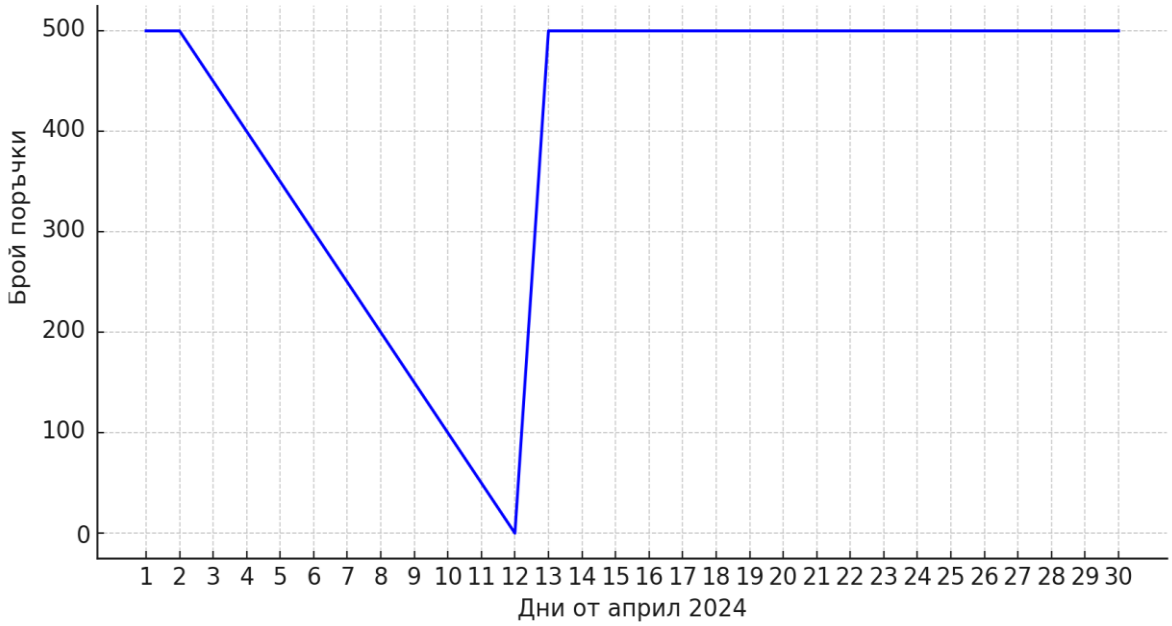


Фиг. 3.9. Обобщен модел на Azure Monitor

Източник: Vettor, 2023.

Значението на непрекъснатото наблюдение е свързано с очакванията на бизнес клиентите за постоянната достъпност на системата. Както бе споменато в глава първа, мониторингът определя и изпълнението на SLA.

Възможни са случайни колебания, предизвикани от непредвидени събития – пазарни смущения, резки промени в потребителските предпочитания или геополитическа нестабилност (Tunç & Büyükkeklik 2017). Наличието на непредвидими колебания в търсенето налага разработване на адаптивни стратегии, предназначени да минимизират риска от недостиг или излишък на запаси (Sulova et al., 2022). В тази връзка, пример за колебание в търсенето, осъществен в контролирана среда, е представен чрез Azure Monitor на фиг. 3.10. Ранното откриване на подобни проблеми, от страна на диспечерите, дава възможност за навременна реакция, способна да предотврати допълнителни проблеми.



Фиг. 3.10. Пример за колебание при търсенето с Azure Monitor

Разработка на автора

3.4.3. Изчисляване на разходите за използване на облачна услуга

На базата на проведените тестове може да се определи приблизителната цена за облачни услуги. За тази цел се използва ценовият калкулатор на Azure (Pricing Calculator), който представлява онлайн инструмент, предназначен за оценка на разходите при използване на различни облачни услуги. С помощта на този инструмент ИТ специалистите могат да определят ориентировъчно колко струват различните ресурси.

Ценовият калкулатор разполага с възможност за сравнение на различни ценови планове, включително Free, Shared, Basic, Standard, Premium и Isolated. Всеки от плановете предлага за използване различни ресурси по отношение на брой процесори, памет, съхранение и допълнителни функции, като автоматично мащабиране и интеграция с виртуални мрежи. Направата на обоснован избор между различните планове дава възможност на компанията да планира по-ефективно своите инвестиции в облачни ресурси.

Таблица 3.9 представя разпределение на очакваните разходи за: изчислителни ресурси, балансиране на натоварването, шлюзове на

приложения, механизми за наблюдение, бази от данни и инструменти за управление на контейнери.

Таблица 3.9

Прогнозни месечни разходи за инфраструктура на облачни услуги

Тип услуга	Описание	Очаквана месечна цена
Виртуални машини	2 D4s v4 (4 vCPU, 16 GB RAM) (запазени за 3 години), Linux	$127.60
Балансьор на натоварването	Стандартно ниво: 5 правила, 1000 GB обработени данни	$23.25
Шлюз за приложения (Application Gateway)	Ниво на защитна стена за уеб приложения V2, 730 фиксирани часа, 5 GB трансфер на данни	$352.15
Azure Monitor	Ежедневни регистрационни файлове, 1 показател за всяка виртуална машина, допълнителни събития и насочени известия	$46.45
Azure Cosmos DBs	Стандартно осигурена пропускателна способност, запис в няколко региона; 400 RU/s x 730 часа; 4000 GB трансакционно хранилище, периодично архивиране	$1,023.36
Брокер на съобщения	24 часа пропускателна способност, приблизително 10 милиона входни събития	$153.58
Инструмент за оркестрация (Kubernetes)	Стандартно ниво; Premium v2 P3V2 (16 ядра, 1,75 GB RAM, 50 GB място за съхранение); 24 часа uptime; Linux OS	$273.00
	Общо	**$1999.40**

Забележка: Потенциалните месечни разходи са обект на промени в резултат от динамиката в ценовата политика на услугите. Посочените стойности са валидни към началото на 2025 г.

Разработка на автора

В прогнозата за месечните разходи са използвани Standard и Premium ценови планове, които осигуряват мащабиране на системата. По наше мнение, ценовите планове са подходящи за компании с голям обем от данни, като „Хайделберг Цимент Девня“.

Изводи и обобщения към трета глава

Производственото предприятие „Хайделберг Цимент Девня“ АД е

водещ производител на цимент, готови бетонови смеси, инертни материали и асфалт. Компанията прилага вертикално интегриран бизнес модел, обхващащ всички етапи от производството до доставката. Тя е част от глобалната компания Heidelberg Materials, която активно инвестира в технологични иновации с цел оптимизиране на операциите в своите търговски организации.

Въвеждането на облачна система в предприятието се разглежда като решение на специфични логистични проблеми:

- Предоставяне и приемане на информация директно от бизнес клиентите чрез възможности за извличане, създаване и промяна на данни за поръчки чрез мобилно приложение.
- Проследяване на доставките чрез системата за местоположението в реално време на превозните средства и текущото състояние на превозваните продукти.
- Автоматизацията на графиците за доставка, насочена към подобряване на процесите по планиране, вземайки предвид наличните ресурси, поръчки и приоритети.

Внедряването на облачната система в „Хайделберг Цимент Девня“ АД представлява сложен и многоетапен процес, изискващ планиране и координация между различни звена в компанията. Процесът започва с анализ на дейността на компанията, избор на подходящ доставчик на облачни услуги и необходими технологични средства за физическа реализация на системата. Използването на ASP.NET Core и Azure осигурява оперативна съвместимост с работещите ERP, CRM и TMS подсистеми на предприятието, което дава възможност за централизирано управление на данните. Функционирането на ПОСУП се осъществява чрез интегрирането на редица услуги, сред които Azure Kubernetes, SQL Database, Cosmos DB и RabbitMQ, както и технологиите Blazor и .NET MAUI.

Изброените технологични средства дават възможност за преминаване към етап на тестване, като се използва стратегия за A/B тестване. На базата на резултатите от тестовете се оценяват ползите от разработката и

внедряването, както и потенциалните възможности за бъдещо развитие. Мониторингът и оценка на очакваните разходи, помагат за определяне и следене на различни бизнес и ИТ показатели.

Интегрирането на ПОСУП към корпоративните подсистеми на производственото предприятие създава основа за обработка и управление на данни, обединяваща оперативна съвместимост и облачни услуги.

Заключение

През последните години производствените предприятия инвестират в усъвършенстване на информационните си системи чрез облачни технологии, с цел да повишат оперативната ефективност, да подобрят точността на данните и да оптимизират управлението на ресурсите. Тези инвестиции целят автоматизация на процесите във вътрешната веригата на доставки, повишаване на нивото на дигитализация и намаляване на необходимостта от ръчен труд. В настоящия дисертационен труд се разглеждат проблеми, свързани с управлението на поръчки за продажба от бизнес клиенти, логистичните процеси по доставки на продукция, както и координацията и комуникацията между заинтересовани страни. Рационализирането на процесите по управление на поръчките се осъществява посредством ПОСУП.

В първа глава е представена теоретична рамка, която включва анализ на съвременни ERP и SCM системи, тяхното приложение във веригата на доставки, различни облачни архитектури и технологии. Първа глава обхваща виртуализация, контейнеризация, инструменти за оркестрация, както и различни модели на облачни услуги: IaaS, PaaS и SaaS. Разгледани са възможностите за интеграция и надграждане на съществуващи подсистеми чрез облачни услуги, както и управлението на бизнес процесите чрез ориентиран към домейн дизайн.

Изхождайки от принципите и практиките за управление на поръчки, втора глава представя концептуален, логически и комуникационен модел, както функционалност и потребителски интерфейс на облачната система. Разработването на моделите се основава на прилагането на стандарти за визуално моделиране. Архитектурата на системата и предложените модели са част от втората изследователска задача и представляват основен принос на настоящото изследване.

Въз основа на създадените модели, в трета глава от изследването се разглежда общата характеристика на дейността на „Хейделберг Цимент

Девня“ АД. След това се извършва подбор на технологични средства за физическа реализация на системата. Практическата приложимост на изследването се демонстрира чрез прилагане на A/B тестване. Въз основа на резултатите от тестовете може да се заключи, че разработената ПОСУП подпомага изпълнението на поръчките, подобрява управлението на ресурсите, осигурява автоматизация и мащабируемост.

Системата е проектирана с възможности за интеграция на платежна система, управляваща плащанията по доставките, както и с механизъм за мониторинг на въглеродните емисии.

С оглед на нарастващата сложност на производствените процеси и необходимостта от надеждни прогнози за пазарното търсене, считаме, че в бъдеще внедряването на изкуствен интелект би допринесло за повишаване на гъвкавостта на предприятието при динамични пазарни условия. Чрез анализ на данни от базите на микроуслугите и мониторинга на системата, е възможно да се предвидят потенциални проблеми и да се осигури проактивна поддръжка. Вместо бизнес клиентите да се обръщат към диспечерите, те могат да използват чатбот, който има вътрешен достъп до комбинирана информация от потребителската сесия и агрегирани данни. Azure осигурява езикови модели на OpenAI и предоставя достъп до когнитивни услуги и инструменти като OpenAI Studio.

Използвана литература

1. Александрова, Я. (2020). *Архитектура на аналитична система за управление на взаимоотношенията с клиентите*. Варна: Знание и бизнес.
2. Армянова, М. (2018). *Осигуряване на сигурността в облачна система чрез шаблони за проектиране*. Известия на Съюза на учените – Варна. Серия „Икономически науки”, 7 (1), с. 252 – 261.
3. Банков, Б. & Петкова, Д. (2024). *Деконструиране на мрежата от данни: иновации в използването на API [Deconstructing the Web of Data: Innovations in the Usage of API]*. Информационни и комуникационни технологии в бизнеса и образованието : Сборник с доклади от Международна научна конференция по случай 55 години от създаването на катедра "Информатика" при Икономически университет – Варна = Information and Communication Technologies in Business and Education : Proceedings of the International Conference Dedicated to the 55th Anniversary of the Department of Informatics, Варна : Наука и икономика, 2024, 122-128.
4. Василев, Ю. (2015). *За един от аспектите на е-логистиката – споделяне на информация във веригите за доставка на строителни предприятия*. Строително предприемачество и недвижима собственост : Сб. с докл. от 30-та юбил. междунар. науч.-практ. конф., ноем. 2015 = Construction Entrepreneurship and Real Property : Proc. of the 30th Anniversary Inter. Sci. a. Practical Conf. in Nov. 2015. с. 403 - 409.
5. Василев, Ю. (2015). *Предоставяне на логистична информация чрез УЕБ услуги*. Икономика и компютърни науки, 2015, № 2, с. 7 - 16.
6. Василев, Ю. (2017*). Електронната логистика в условията на глобализация*. Варна: Наука и икономика, библиотека "Проф. Цани Каляджиев", 2017, 193.
7. Василев, Ю. (2018). *Системи за управление на бази от данни. MySQL*. Варна: ЦПО Знание и бизнес, 2018, 15.
8. Георгиева, М. (2025). *Топ 10 Компании за Строителни материали*. Капитал. < https://www.capital.bg/biznes/promishlenost/2025/03/13/4737829_top_10_proizvoditeli_na_stroitelni_materiali_na/ > [06.04.2025]
9. Димитров, П. (2018). *Алгоритмични проблеми при внедряване на*

двуфакторна автентикация в уеб приложения. Сборник с доклади от научна конференция на младите научни работници. Варна: Стено, с. 126 – 131.

10. Димитров, И. (2020). *Проблеми на управленското и финансовото счетоводство при приложението на ERP системи*. Варна: Наука и икономика.
11. Димитрова, В. (2023). *Взаимоотношенията във веригата за доставка в омниканалния ритейлинг*. Научни трудове. Университет за национално и световно стопанство, София : Изд. комплекс УНСС, Год. 63, 2023, 1, с. 23-33.
12. Куюмджиев, И. (2019). *Методологически и технологични аспекти при архивирането на бази от данни*. Варна: Наука и икономика, Библ. Проф. Цани Каляджиев.
13. Маринова, О. (2015). *Нови принципи в управлението на ИТ проекти чрез използването на Agile методологии*. Изв. на Съюза на учените – Варна. Сер. Икономически науки , 2015, с. 117 - 124..
14. Милушева, П. (2023). *Проблеми при управление на логистиката в строителството*. Варна : Ико-консулт, 2023, 152. - (Библ. PhD Защитени докторски дисертации).
15. Моллов, Д. (2017*). Глобални вериги за доставка – концепции и стратегии*. Второ преработено и допълнено издание. София, Издателски комплекс – УНСС.
16. Парушева, С. & Александрова, Я. (2022). *Дигитализация в строителството в контекста на въздействащи движещи сили и фактори*. Списание на Българската академия на науките: Общоакадемично списание на БАН, 135 (2), с. 65 – 70.
17. Петров, П., Сълова, С., Радев, М., Александрова, Я., Стоянова, М., Милева, Л., Янков, П. (2020). *Дигитализация на бизнес процеси в строителството и логистиката*. Варна : Знание и бизнес, 2020, 251 с. - (Моногр. библ. Знание и бизнес ; Кн. 8)
18. Радев, М. (2015). *Инвентаризация на ИТ инфраструктурата.* Икономиката в променящия се свят: национални, регионални и глобални измерения : Сб. докл. от междунар. науч. конф. : Т. 3. - Варна : Унив. изд. Наука и икономика, 2015, с. 173 - 178.
19. Раковска, М. (2021). *Иновации и добри практики в логистиката и управлението на веригата на доставките*. Университет за национално и

световно стопанство (УНСС). Научен сборник от Втората научно-бизнес конференция по логистика и управление на веригата на доставките, посветена на 30-ата годишнина от създаването на специалност „Бизнес логистика“ в Университета за национално и световно стопанство.

20. Сълов, В. (2022). *Приложение на проектите на платформата .NET при разработка на уеб приложения*. Известия на Икономически университет – Варна, 66 (4), с. 362 – 375.
21. Сълова, С. (2019). *Дигитализацията в строителството и необходимостта от иновативни модели за управление на данните* . Строително предприемачество и недвижима собственост : Сборник с доклади от 34-та международна научно-практическа конференция - ноември 2019 г., посветена на 100 г. от създаването на ИУ - Варна, Варна : Наука и икономика, 2019, с. 78 - 85.
22. Сълова, С., Банков, Б. & Стоянова, М. (2024) *Уеб технологии*. Варна : Наука и икономика.
23. Тодоранова, Л. (2024). *Дигиталната трансформация на обучението и приобщаването. Дигитална трансформация на образованието - проблеми и решения* : Втора национална научно-практическа конференция : Сборник с доклади, 25-26 април 2024 г, Русе : Унив. изд. на РУ А. Кънчев, 2024, 233-237.
24. Шишманов, К. & Маринова-Костова, К. (2024). *Осъществяване на дигитална трансформация на предприятия, базирана на интеграция на приложения*. Международна научна конференция "Информационни и комуникационни технологии в бизнеса и образованието" (стр. 14-19). „Наука и икономика“, Икономически университет – Варна.
25. Agarwal, C. (2021). *Implementing order to cash process in SAP: An end-to-end guide to understanding the OTC process and its integration with SAP CRM, SAP APO, SAP TMS, and SAP LES*. Packt Publishing.
26. Aleksandrova, Y. (2021). *Predictive Analytics Implementation in the Logistic Industry*. Economics and Computer, Varna : Knowledge and Business, 7, 2021, 2, 6-22.
27. Althabatah, A., Yaqot, M., Menezes, B. C., Kerbache, L. (2023). *Transformative Procurement Trends: Integrating Industry 4.0 technologies for enhanced procurement processes*. Logistics, *7* (3), p. 63.
28. Alzoubi, H. M., Ahmed, G., Al-Gasaymeh, A., Kurdi, B. A. (2020). *Empirical study on sustainable supply chain strategies and its impact on*

competitive priorities: The mediating role of supply chain collaboration. Management Science Letters, pp. 703 – 708.

29. Armiyanova, M. (2017). *Approach for design pattern' application in the development of information systems*. Eastern Academic Journal, Burgas : Miracle A Ltd., Vol. 4, December, 2017, pp. 62 - 75.
30. Armiyanova, M. (2019). *IoT Problems and Design Patterns which are Appropriate to Solve them*. Information and Communication Technologies in Business and Education : Proceedings of the International Conference Dedicated to the 50th Anniversary of the Department of Informatics, Varna : Science a. Economic Publ. House , 2019, 291 - 305.
31. Atchison, L. (2020). *Architecting for scale: How to Maintain High Availability and Manage Risk in the Cloud*. O'Reilly Media.
32. Barata, F. A., Febrianto, G. N., Yasin, M. (2022). *Supply chain Management strategy in building a competitive advantage through the implementation of Logistic 4.0*. Advances in economics, business and management , pp. 369–377
33. Batista, F. (2022). *Developing the ubiquitous language* - The Domain Driven Design. <https://thedomaindrivendesign.io/developing-the-ubiquitous-language/> [10.12.2023]
34. Becker, U., Herhuth, W., Hirn, M. (2016). *Pricing and the condition technique in SAP ERP*. SAP Press.
35. Betts, D., Dominguez, J., Melnik, G., Simonazzi, F., Subramanian, M. (2013). *Exploring CQRS and Event Sourcing: A journey into high scalability, availability, and maintainability with Windows Azure*. Microsoft patterns.
36. Bier, T., Lange, A., Glock, C. H. (2019). *Methods for mitigating disruptions in complex supply chain structures: a systematic literature review*. International Journal of Production Research, 58 (6), pp. 1835 – 1856.
37. Bilovodska, O., Syhyda, L., & Saher, L. (2018). *Supply chain management: world's companies experience*. MIND Journal, (5), 1-17.
38. Bisogni, P. G., Brdulak, H., Cantoni, F., Niine, T., Zsifkovits, H. (2021*). The role of European Logistics Association 2020 Standards in facing modern industry expectations and logistics managers' competencies*. International Journal of Value Chain Management, 12 (2), p. 171.
39. Bönnen, C., Drees, V., Fischer, A., Heinz, L., Strothmann, K. (2018). *SAP Gateway and OData*. SAP Press.
40. Braun, S., Bieniusa, A., Elberzhager, F. (2021). *Advanced Domain-Driven*

Design for Consistency in Distributed Data-Intensive Systems. European Conference on Computer Systems.

41. Brewer, E. (2012). *Pushing the CAP: Strategies for Consistency and Availability*. IEEE Computer, 45 (2), pp. 23 – 29.
42. Brewster, T. (2018). *Marriott hackers stole data on 500 million guests passports and credit card info included*. Forbes. <https://www.forbes.com/sites/thomasbrewster/2018/11/30/marriott-admits-hackers-stole-data-on-500-million-guests/?sh=50f10ba46492#786737086492> [20.12.2021].
43. Calabrò, G., Torrisi, V., Inturri, G., Ignaccolo, M. (2020). *Improving inbound logistic planning for large-scale real-world routing problems: a novel ant-colony simulation-based optimization*. European Transport Research Review, 12 (1).
44. Caserio, C., Trucco, S. (2018). *Enterprise Resource Planning and Business intelligence systems for information quality: An Empirical Analysis in the Italian Setting*. Springer.
45. Cataldo, I., Banaitis, A., Samadhiya, A., Banaitienė, N., Kumar, A., Luthra, S. (2022). *Sustainable Supply Chain Management in Construction: an Exploratory Review for Future Research*. Journal of Civil Engineering and Management, 28 (7), pp. 536–553.
46. Ćatović, A., Buzađija, N., & Lemes, S. (2022). *Microservice development using RabbitMQ message broker*. Science Engineering and Technology, 2(1), 30–37.
47. Chen, Y. (2020). *Intelligent algorithms for cold chain logistics distribution optimization based on big data cloud computing analysis*. Journal of Cloud Computing, 9.
48. Cichosz, M., Wallenburg, C. M., Knemeyer, A. M. (2020). *Digital transformation at logistics service providers: barriers, success factors and leading practices*. The International Journal of Logistics Management, 31(2), pp. 209 – 238.
49. Cloud Native Computing Foundation (2022) *What is cloud native and why does it exist?*. CNCF <https://www.cncf.io/online-programs/what-is-cloud-native-and-why-does-it-exist/>[25.01.2024]
50. Cloud Native Computing Foundation. (2023). *CNCF Annual Survey 2023*. CNCF. <https://www.cncf.io/reports/cncf-annual-survey-2023/> [17.05.2024]
51. De La Torre, C. (2017). *Domain Events vs. Integration Events in Domain-*

Driven Design and microservices architectures. <https://devblogs.microsoft.com/cesardelatorre/domain-events-vs-integration-events-in-domain-driven-design-and-microservices-architectures/> [01.03.2024]

52. De la Torre, C. (2023). *Containerized Docker application lifecycle with Microsoft platform and tools*. Microsoft Corporation.
53. De la Torre, C., Wagner, B., Rousos, M. (2024). *.NET microservices: Architecture for containerized .NET applications*. Microsoft Corporation
54. Debski, A., Szczepanik, B., Malawski, M., Spahr, S., Muthig, D. (2018). *A Scalable, Reactive Architecture for Cloud Applications*. IEEE Software, 35 (2), pp. 62 – 71.
55. Delnavaz, M., Sahraei, A., Delnavaz, A., Farokhzad, R., Amiri, S., Bozorgmehrnia, S. (2022). *Production of concrete using reclaimed water from a ready-mix concrete batching plant: Life cycle assessment (LCA), mechanical and durability properties*. Journal of Building Engineering, 45, 103560.
56. Dotson, C. (2019). *Practical Cloud Security: A Guide for Secure Design and Deployment*. O'Reilly Media.
57. Duff, J. (2024). *Fortune 500 companies that use SAP*. Thomson Data <https://www.thomsondata.com/blog/fortune-500-companies-that-use-sap/> [21.09.2024]
58. Elgheriani, N. S., Ahme, N. D. (2022). *Microservices vs. Monolithic Architectures The Differential Structure between Two Architectures*. MINAR International Journal of Applied Sciences and Technology, 4 (3), pp. 500 – 514 .
59. Endo, P. T., Rodrigues, M., Gonçalves, G. E., Kelner, J., Sadok, D., Curescu, C. (2016). *High availability in clouds: systematic review and research challenges*. Journal of Cloud Computing, 5.
60. Erl, T. (2007). *SOA Principles of Service Design*. Prentice Hall PTR eBooks.
61. Esposito, D. (2016). *Modern Web Development: Understanding Domains, Technologies, and User Experience*. Microsoft Press.
62. Evans, E. (2014). *Domain-Driven Design Reference: Definitions and Pattern Summaries*. Dog Ear Publishing.
63. Evans, E., Evans, E. J. (2004). *Domain-driven Design: Tackling Complexity in the Heart of Software*. Addison-Wesley Professional.

64. Fields, J., Harvie, S., Fowler, M., Beck, K. (2009). *Refactoring: Ruby Edition*. Pearson Education.
65. Fowler, M. (2010). *Domain-Specific Languages*. Pearson Education.
66. Fowler, M. (2012). *Patterns of Enterprise Application Architecture*. Addison-Wesley.
67. Fowler, M. (2019). *Agile Software Guide*. <https://martinfowler.com/agile.html> [10.12.2021]
68. Frey, D. (2023). *Automating load selection, truck dispatch, and backhaul activation in outbound logistics operations* <https://dspace.mit.edu/handle/1721.1/151587?show=full> [10.10.2024]
69. Flexera. (2023). *Flexera 2023 State of the Cloud Report* <https://info.flexera.com/CM-REPORT-State-of-the-Cloud > [14.05.2024]
70. Garg, S. (2019). *Automated Cloud Infrastructure, Continuous Integration and Continuous Delivery using Docker with Robust Container Security*. 2019 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR).
71. Gargeya, V. B., Brady, C. (2005). *Success and failure factors of adopting SAP in ERP system implementation*. Business Process Management Journal, 11 (5), pp. 501 – 516.
72. Garrett, R. D., Lambin, E. F., & Naylor, R. L. (2013). *The new economic geography of land use change: Supply chain configurations and land use in the Brazilian Amazon*. Land use policy, 34, 265-275.
73. Garverick, J., McIver, O. D. (2023). *Implementing Event-Driven Microservices Architecture in .NET 7: Develop event-based distributed apps that can scale with ever-changing business demands using C# 11 and .NET 7*. Packt Publishing.
74. Gaur, M. (2020). *ERP Migration Challenges and Solution Approach for Digital Transformation to SAP S/4HANA for SAP customers*. Social Science Research Network.
75. Gartner. (2023). *Magic Quadrant for Strategic Cloud Platform Services*. < https://www.gartner.com> [30.03.2024]
76. González, R., Gascó, J. L., Llopis, J. (2024). *Towards organisation 4.0. An empirical study*. International Journal of Information Management, 75, 102746.
77. Gouigoux, J. (2024). *Enterprise Architecture with .NET: Expert-backed*

advice for information system design, down to .NET and C# implementation. Packt Publishing Ltd.

78. Grafiati. (2022). *Academic literature on the topic "Twelve-factor app"* O'Reilly Media.
79. Guo, Z., Guo, C. (2013). *A cloud-based decision support system framework for order planning and tracking*. Springer eBooks, pp. 85 – 98
80. Gupta, S. M. (2016). *Reverse supply chains: Issues and Analysis*. CRC Press.
81. Hahn, G. J. (2019). *Industry 4.0: a supply chain innovation perspective*. International Journal of Production Research, 58 (5), pp. 1425– 1441.
82. Hartley, J. L., Sawaya, W. J. (2019). *Digital transformation of supply chain business processes*. Business Horizons, 62 (6), pp. 707 – 715.
83. Hasim, S., Fauzi, M. A., Yusof, Z., Endut, I. R., Ridzuan, A. R. M. (2018). *The material supply chain management in a construction project: A current scenario in the procurement process*. AIP Conference Proceedings.
84. Henning, S., Hasselbring, W. (2022). *A configurable method for benchmarking scalability of cloud-native applications*. Empirical Software Engineering, 27 (6).
85. Heusser, M. (2019). *How to achieve speedy application response times. Software Quality* <https://www.techtarget.com/searchsoftwarequality/tip/Acceptable-application-response-times-vs-industry-standard > [12.02.2021]
86. Hildebrand, K. (2018). *Master Data Life Cycle – Management der Materialstammdaten in SAP*. Springer eBooks, pp. 299 – 310.
87. Hippchen, B., Giessler, P., Steinegger, R. H., Schneider, M., Abeck, S. (2017). *Designing Microservice-Based Applications by Using a Domain-Driven Design Approach*. International Journal on Advances in Software, 10, pp. 432 – 445.
88. Hoffman, K. (2016). *Beyond the twelve-factor app: Exploring the DNA of Highly Scalable*. Resilient Cloud Applications.
89. Hofmann, E., Sternberg, H., Chen, H., Pflaum, A., Prockl, G. (2019). *Supply chain management and Industry 4.0: conducting research in the digital age*. International Journal of Physical Distribution & Logistics Management, 49 (10), pp. 945 – 955.
90. Huang, D., Xing, T., Wu, H. (2013). *Mobile cloud computing service models: a user-centric approach*. IEEE Network, 27 (5), pp. 6 – 11.

91. Indrasiri, K., Suhothayan, S. (2021). *Design patterns for cloud native applications*. O'Reilly Media.
92. Ingeno, J. (2018). *Software Architect's Handbook: Become a successful software architect by implementing effective architecture concepts*. Packt Publishing.
93. Kaspersky. (2024). *Security Bulletin Statistics*. Securelist <https://securelist.com/ksb-2024-statistics/114795/> [07.01.2025]
94. Karthikeyan, S. A. (2021). *Demystifying the Azure Well-Architected framework: Guiding Principles and Design Best Practices for Azure Workloads*. Apress.
95. Kakhki, M. D., Gargeya, V. B. (2019). *Information systems for supply chain management: a systematic literature analysis*. International Journal of Production Research, 57 (15–16), pp. 5318 – 5339.
96. Katsaliaki, K., Galetsi, P., Kumar, S. (2021). *Supply chain disruptions and resilience: a major review and future research agenda*. Annals of Operations Research, 319 (1), pp. 965 – 1002.
97. Kesan, J. P., Hayes, C., Bashir, M. (2013). *Information privacy and data control in cloud computing: consumers, privacy preferences, and market efficiency*. Washington and Lee Law Review, 70 (1), pp. 341 – 472.
98. Khan, S. A. R., & Yu, Z. (2019). *Introduction to supply chain management*. EAI/Springer Innovations in Communication and Computing, pp. 1 – 22.
99. Khononov, V. (2021). *Learning Domain-Driven Design*. O'Reilly Media.
100. Knolmayer, G. F., Mertens, P., Zeier, A. (2012*). Supply chain management based on SAP systems: Order Management in Manufacturing Companies*. Springer Science & Business Media.
101. Kumar, V., Agnihotri, K. (2021). *Serverless Computing Using Azure Functions: Build, Deploy, Automate, and Secure Serverless Application Development with Azure Functions*. BPB Publications.
102. Kuyumdzhiev, I., Nacheva, R. (2020). *Correlation between storage device and backup and restore efficiency in MS SQL server*. Serdica Journal of Computing, 13 (3 – 4), pp. 139 – 154
103. Lano, K., Tehrani, S. Y. (2023). *Introduction to software Architecture: Innovative Design using Clean Architecture and Model-Driven Engineering*. Springer Nature.
104. Laszewski, T., Arora, K., Farr, E., Zonooz, P. (2018). *Cloud Native*

Architectures: Design high-availability and cost-effective applications for the cloud. Packt Publishing.

105. Le, T. T. (2020). *Performance measures and metrics in a supply chain environment*. Uncertain Supply Chain Management, pp. 93 – 104.
106. Li, S., Zhang, H., Jia, Z., Zhong, C., Zhang, C., Shan, Z., Shen, J., Babar, M. A. (2021). *Understanding and addressing quality attributes of microservices architecture: A Systematic literature review*. Information & Software Technology, 131, 106449.
107. Likness, J., & Phillip, C. (2024). *Serverless apps: Architecture, patterns, and Azure implementation*. Microsoft < https://learn.microsoft.com/en-us/dotnet/architecture> [14.12.2024]
108. Magal, S., Word, J. (2013). *Business Process Integration with SAP ERP*. Packt Publishing.
109. Martin, R. C. (2008). *Clean Code: A Handbook of Agile Software Craftsmanship*. Pearson Education.
110. Martin, R. C. (2017). *Clean Architecture: A Craftsman's Guide to Software Structure and Design*. Prentice Hall.
111. Matinheikki, J., Kauppi, K., Brandon, Jones, A., Van Raaij, E. (2022). *Making agency theory work for supply chain relationships: a systematic review across four disciplines*. International Journal of Operations & Production Management, 42 (13), pp. 299 – 334.
112. Mahasivabhattu, K., & Bandi, D. (2024). *Ultimate Machine Learning with ML.NET: Build, Optimize, and Deploy Powerful Machine Learning Models for Data-Driven Insights with ML.NET, Azure Functions, and Web API.* Orange Education Pvt Ltd.
113. Meyer B. (2009). *Touch of class: Learning to Program Well with Objects and Contracts.* Springer Science & Business Media.
114. Millett, S., Tune, N. (2015). *Patterns, Principles, and Practices of Domain-Driven Design*. John Wiley & Sons.
115. Mohammed, C. M., Zeebaree, S. R. M. (2021). *Sufficient comparison among cloud computing services: IAAS, PAAS, and SAAS: A review*. International Journal of Science and Business, 5 (2), pp. 17 – 30.
116. Molamohamadi, Z., Tirkolaee, E. B., Mirzazadeh, A., & Weber, G. (2021). *Logistics and supply chain management*. Communications in computer and information science.

117. Moniz, A., Gordon, M., Bergum, I., Chang, M., & Grant, G. (2021). *Beginning Azure Cognitive services*. Apress eBooks.
118. Nacheva, R. (2020). *Standardization Issues of Mobile Usability*. International Journal of Interactive Mobile Technologies, 14, 2020, 7, 149 – 157.
119. Nacheva, R., Sulova, S. & Penchev, B. (2022). *Where security meets accessibility: Mobile Research Ecosystem*. Springer eBooks, pp. 216 – 231.
120. Nguyen, P., Song, H., Chauvel, F., Muller, R., Boyar, S., Levin, E. (2019). *Using microservices for non-intrusive customization of multi-tenant SaaS*. 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering
121. Novais, L. R., Marín, J. M. M., Ortíz, Á. (2019). *A systematic literature review of cloud computing use in supply chain integration*. Computers & Industrial Engineering, 129, pp. 296 – 314.
122. Nikolov.D. (2019). *Shipping pseudocode to production*. DotNetCurry. <https://www.dotnetcurry.com/patterns-practices/1497/deploy-pseudocode-production>[13.01.2025]
123. Nivala, M., Seredko, A., Osborne, T., & Hillman, T. (2023). *Stack Overflow Annual Developer Survey*. < https://survey.stackoverflow.co/> [08.05.2023]
124. Oh, S., Koo, J., & Kim, Y. (2022). *Security interoperability in heterogeneous IoT platforms*. Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing.
125. Ojra, J., Opute, A. P., Alsolmi, M. M. (2021). *Strategic management accounting and performance implications: a literature review and research agenda*. Future Business Journal, 7 (1)
126. Oukes, P., Van Andel, M., Folmer, E., Bennett, R., Lemmen, C. (2021). *Domain-Driven Design applied to land administration system development: Lessons from the Netherlands*. Land Use Policy, 104, 105379
127. Paganini, P. (2019). *Hacker deleted all data from VFEmail Servers, including backups*. Security Affairs <https://securityaffairs.com/81030/hacking/vfemail-destructive-cyberattack.html> [01.10.2024]
128. Palermo, J. (2018). *The Onion Architecture*. Programming with Palermo < https://jeffreypalermo.com/2008/07/the-onion-architecture-part-1 > [15.03.2024]
129. Palermo, J. (2019). *.NET DevOps for Azure: A Developer's Guide to DevOps Architecture the Right Way*. Apress.

130. Partida, D. (2023). *Cloud Supply Chain Management: Benefits and use cases*. Enterprise Networking Planet <https://www.enterprisenetworkingplanet.com/data-center/cloud-computing-supply-chain/> [24.06.2024]
131. Parusheva, S. (2019) *Digitalization and Digital Transformation in Construction - Benefits and Challenges*. Information and Communication Technologies in Business and Education : Proceedings of the International Conference Dedicated to the 50th Anniversary of the Department of Informatics, Varna : Science a. Economic Publ. House, 2019, 126-134.
132. Parusheva, S. & Pencheva, D. (2021). *Modeling a Business Intelligent System for Managing Orders to Supplier in the Retail Chain with Unified Model Language*. Lecture Notes in networks and systems, pp. 375 – 393.
133. Penchev, B. (2016). *Effectiveness of a Conceptual Model for Increased Mobile Banking Security*. Serdica Journal of Computing, 10, 2016, 1, p. 49–62.
134. Pjp, I. (2023). *Mastering SAP Master Data Governance (MDG): A Guide to Data Management and Governance*. Independently Published.
135. Rademacher, F., Sachweh, S., Zündorf, A. (2017). *Towards a UML Profile for Domain-Driven Design of Microservice Architectures*. Lecture Notes in Computer Science. Springer Science Businessy Media, pp. 230 – 245.
136. Rademacher, F., Sorgalla, J., Sachweh, S. (2018). *Challenges of Domain-Driven Microservice Design: A Model-Driven Perspective*. IEEE Software, 35 (3), pp. 36 – 43.
137. Radev, M. (2023). *Onboarding Process Automation.* Izvestia. Journal of the Union of Scientists-Varna. Economic Sciences Series, Varna : Union of Scientists - Varna, 12, 2023, 2, 147-154.
138. Rajapakse, Duminda. (2023). *Integration Between ERP Systems And Supply Chain Management*. Studies in Communication and Media.
139. Ramakrishna, Y. (2022). *Handbook of Research on Supply Chain Resiliency, Efficiency, and Visibility in the Post-Pandemic Era*. IGI Global.
140. Ren, S., Zhang, Y., Liu, Y., Sakao, T., Huisingh, D., Almeida, C. (2019). *A comprehensive review of big data analytics throughout product lifecycle to support sustainable smart manufacturing: A framework, challenges and future research directions*. Journal of Cleaner Production, 210, pp. 1343 – 1365.
141. Roy, R. (2023). *Cloud Supply Chain Software. Revolutionizing SCM*.

Selecthub <https://www.selecthub.com/supply-chain-management/10-ways-cloud-computing-revolutionizing-supply-chain-management/> [03.05.2024].

142. Runkler, T. A., Chen, C., Coupland, S., & John, R. (2019). *Just–In–Time Supply chain Management using interval Type–2 Fuzzy decision making*. IEEE International Conference on Fuzzy Systems , 8, 1–6.
143. Rendle, M., Steiner, M. (2024). gRPC for WCF developers. Microsoft <https://learn.microsoft.com/en-us/dotnet/architecture > [30.03.2024]
144. Roth, D., Fritz, J., & Southwick, T. (2024). Blazor for ASP.NET Web Forms developers. Microsoft (https://learn.microsoft.com/en-us/dotnet/architecture, 20 февруари 2024).
145. Sandberg, E. (2025). *Strategic Logistics Management: Contemporary Principles and Practice*. Kogan Page Publishers.
146. Sazanavets, F. (2022). *Secure SignalR endpoints for both web UI clients and IoT devices*. In Apress eBooks.
147. Schachenhofer, L., Kummer, Y., Hirsch, P. (2023). *An Analysis of Underused Urban Infrastructures: Usage opportunities and implementation Barriers for sustainable logistics*. Applied Sciences, 13 (13), p. 7557.
148. Schneider, R. (2020). *Practical Guide to SAP Business Partner Functions and Integration with SAP S/4HANA*. Espresso Tutorials
149. Schniederjans, D. G., Curado, C., Khalajhedayati, M. (2020). *Supply chain digitisation trends: An integration of knowledge management*. International Journal of Production Economics, 220, 107439.
150. Steinegger, R. H., Giessler, P., Hippchen, B., Abeck, S. (2017). *Overview of a Domain-Driven Design Approach to Build Microservice-Based Applications*. The Third International Conference on Advances and Trends in Software Engineering, pp. 79 – 87.
151. Stuckenberg, S., Kude, T., Heinzl, A. (2014). *Understanding the role of organizational integration in developing and operating Software-as-a-Service*. Journal of Business Economics, 84 (8), pp. 1019 – 1050.
152. Sullivan, M., Kern, J. (2021). *The digital transformation of logistics: Demystifying Impacts of the Fourth Industrial Revolution*. John Wiley & Sons.
153. Sulov, V. (2024). *Render modes of .NET Blazor technology*. Journal of Informatics and Innovative Technologies, 1(6).
154. Sulova, S., Aleksandrova, Y., Stoyanova, M., Radev, M. (2022). *A Predictive*

Analytics Framework Using Machine Learning for the Logistics Industry. CompSysTech'22 : 23rd International Conference on Computer Systems and Technologies, 17-18 June 2022, University of Ruse, Bulgaria, Association for Computing Machinery, 2022, 39-44.

155. Shaikh, K. (2024). Cloud Exit: 42% of Companies Move Data Back On-Premises. <https://www.techopedia.com/cloud-exit-as-companies-move-data-on-premises> [26.11.2024]
156. Sheldon, R., Lutkevich, B., & Gillis, A. S. (2024). *What is high availability. Definition and guide*. Search Data Center. < https://www.techtarget.com/searchdatacenter/definition/high-availability > [20.09.2024]
157. Smith, S. (2025). *Architecting modern web applications with ASP.NET Core and Microsoft Azure*. Microsoft Corporation
158. Soper, C., Addie, S., Dembovsky, C. (2024). *DevOps for ASP.NET Core developers*. Microsoft Corporation
159. Stonis, M. (2024). *Enterprise application patterns using .NET MAUI*. Microsoft Corporation
160. Tang, C., Xia, H. (2023). *Risk analysis and research of construction supply chain*. Highlights in Business Economics and Management, 11, pp. 155 – 160.
161. Templar, S., Hofmann, E., Findlay, C. (2020). *Financing the End-to-End supply chain: A Reference Guide to Supply Chain Finance*. Kogan Page Publishers.
162. Thomas, R., Tokar, T., & Van Hoek, R. (2024). *The strategic advantage omnichannel retailers have over Amazon*. Harvard Business Review. < https://hbr.org/2024/05/the-strategic-advantage-omnichannel-retailers-have-over-amazon > [18.01.2024]
163. Tukamuhabwa, B., Mutebi, H., Kyomuhendo, R. (2021). *Competitive advantage in SMEs: effect of supply chain management practices, logistics capabilities and logistics integration in a developing country*. Journal of Business and Socio-economic Development, 3 (4), pp. 353 – 371.
164. Tunç, T., Büyükkeklik, A. (2017). *Reducing the Negative Effects of Seasonal Demand Fluctuations: A Proposal Based On Cost-Benefit Analysis*. International Journal of Engineering Research and Applications, 07 (03), pp. 38 – 46.
165. Türkay, M., Saraçoğlu, Ö., Arslan, M. C. (2016). *Sustainability in Supply*

Chain Management: Aggregate Planning from Sustainability Perspective. PloS One, 11 (1)

166. Toub, S. (2024*). Performance improvements in .NET 7*. Microsoft Corporation
167. Uludağ, Ö., Hauder, M., Kleehaus, M., Schimpfle, C., & Matthes, F. (2018). *Supporting Large-Scale Agile Development with Domain-Driven Design*. Lecture Notes in Business Information Processing, pp. 232 – 247.
168. Văcar, A. (2019). *Logistics and Supply Chain Management: An Overview*. Studies in Business and Economics, 14 (2), pp. 209 – 215.
169. Vasilev, J. (2015). *Providing Logistics Information by Web Services*. Directory of Open Access Journals (DOAJ)
170. Vasilev, J., Cristescu, M. P. (2019). *Approaches for information sharing from manufacturing logistics with downstream supply chain partners*. Conferences of the Department Informatics, 1, pp. 24 – 29.
171. Vasilev, J., Stoyanova, M. (2019). *Information sharing with upstream partners of supply chains*. Proceedings of the 19th International Multidisciplinary Scientific GeoConference SGEM 2019, Geoinformatics and Remote Sensing, Vol. 19, Informatics, № 2.1. Sofia: STEF92 Technology, pp. 329 – 336.
172. Vasilev, J. & Kehayova-Stoycheva, M. (2019). *Sales Management by Providing Mobile Access to a Desktop Enterprise Resource Planning System*. TEM Journal, *8*(4) Serbia : UIKTEN-Assoc. for Inform. Communication Technology, pp.1107-1112.
173. Vasilev, J., Nikolaev, R., Milkova, T. (2023). *Transport Task Models with Variable Supplier Availabilities*. Logistics, 7 (3), p. 45.
174. Verdouw, C., Beulens, A., Trienekens, J. H., Wolfert, J. (2010). *Process modelling in demand-driven supply chains: A reference model for the fruit industry*. Computers and Electronics in Agriculture, 73 (2), pp. 174 – 187.
175. Vernon, V. (2013). *Implementing Domain-Driven Design*. Addison-Wesley.
176. Vernon, V. (2016). *Domain-Driven Design Distilled*. Addison-Wesley Professional.
177. Verwijmeren, M. (2004). *Software component architecture in supply chain management*. Computers in Industry, 53 (2), pp. 165 – 178.
178. Vettor, R. (2023). *Architecting Cloud Native .NET Applications for Azure*. Microsoft Learn

179. Vettor, R., Molenkamp, S., & van Wijk, E. (2024). *Dapr for .NET developers*. Microsoft Corporation
180. Vieira, D. (2023). *Designing Hexagonal Architecture with Java: Build maintainable and long-lasting applications with Java and Quarkus*. Packt Publishing.
181. Villaça, L. A., Azevedo, L. G., Baião, F. A. (2018). *Query strategies on polyglot persistence in microservices*. ACM Symposium on Applied Computing.
182. Von Aspen, J. (2020*). First steps in SAP S/4HANA Sales and Distribution (SD)*. Espresso Tutorials.
183. Vettor, R., Smith, S. (2024). *Architecting cloud-native .NET apps for Azure*. Microsoft Corporation
184. Wlaschin, S. (2018). *Domain Modeling Made Functional: Tackle Software Complexity with Domain-Driven Design and F#.* Pragmatic Bookshelf.
185. Xu, R., Jin, W., & Kim, D. (2019). *Microservice Security Agent based on API gateway in edge computing*. Sensors, 19(22), 4905
186. Young, G. (2011). *Event Centric: Finding Simplicity in Complex Systems*. Addison-Wesley Professional.
187. Zając, M., & Świeboda, J. (2023). *Method of assessing the logistics process as regards information flow unreliability on the example of a container terminal*. Applied Sciences, 13 (2), p. 962.
188. Zimarev, A. (2019). *Hands-On Domain-Driven Design with .NET Core: Tackling complexity in the heart of software by putting DDD principles into practice*. Packt Publishing.

Приложения

Приложение 1. Контейнеризация чрез Docker и Kubernetes

Docker, инструментът за контейнеризация, споменат в основния текст, изгражда облачните услуги под формата на виртуални изображения, използвайки т.нар. Docker файлове (Dockerfile). Тези файлове са част от програмния код на всяка микроуслуга и клиентско приложение. Пример за такъв файл:

```
FROM mcr.microsoft.com/dotnet/aspnet:8.0 AS base
EXPOSE 80
FROM mcr.microsoft.com/dotnet/sdk:8.0 AS build
WORKDIR /src
COPY ["Manager.Api/Manager. Api.csproj", "Manager. Api /"]
RUN dotnet build "Manager. Api.csproj" -c Release -o /app/build
ENTRYPOINT ["dotnet", "Manager.Api.dll"]
```

Kubernetes използва Dockerfile и YAML файлове, за да автоматизира процеса по внедряване на приложенията в облачната инфраструктура на Azure. YAML файлове в Kubernetes включват:

1) Deployment.yaml

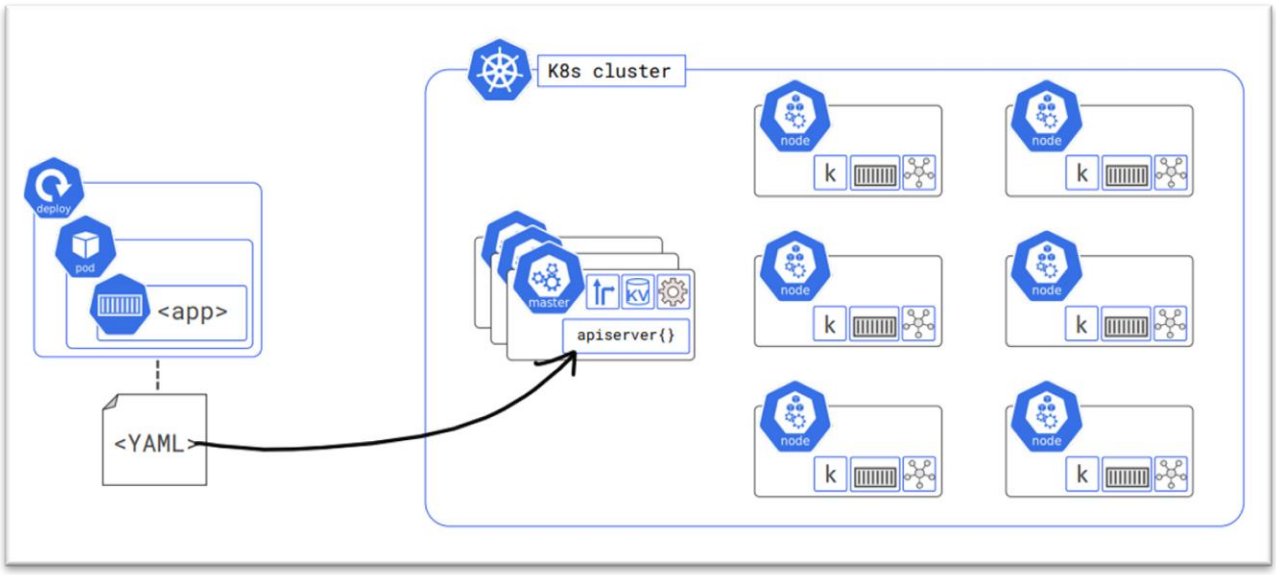
```
apiVersion: apps/v1
kind: Deployment
metadata: name: manager-api
spec:
    replicas: 2
```

2) Service.yaml

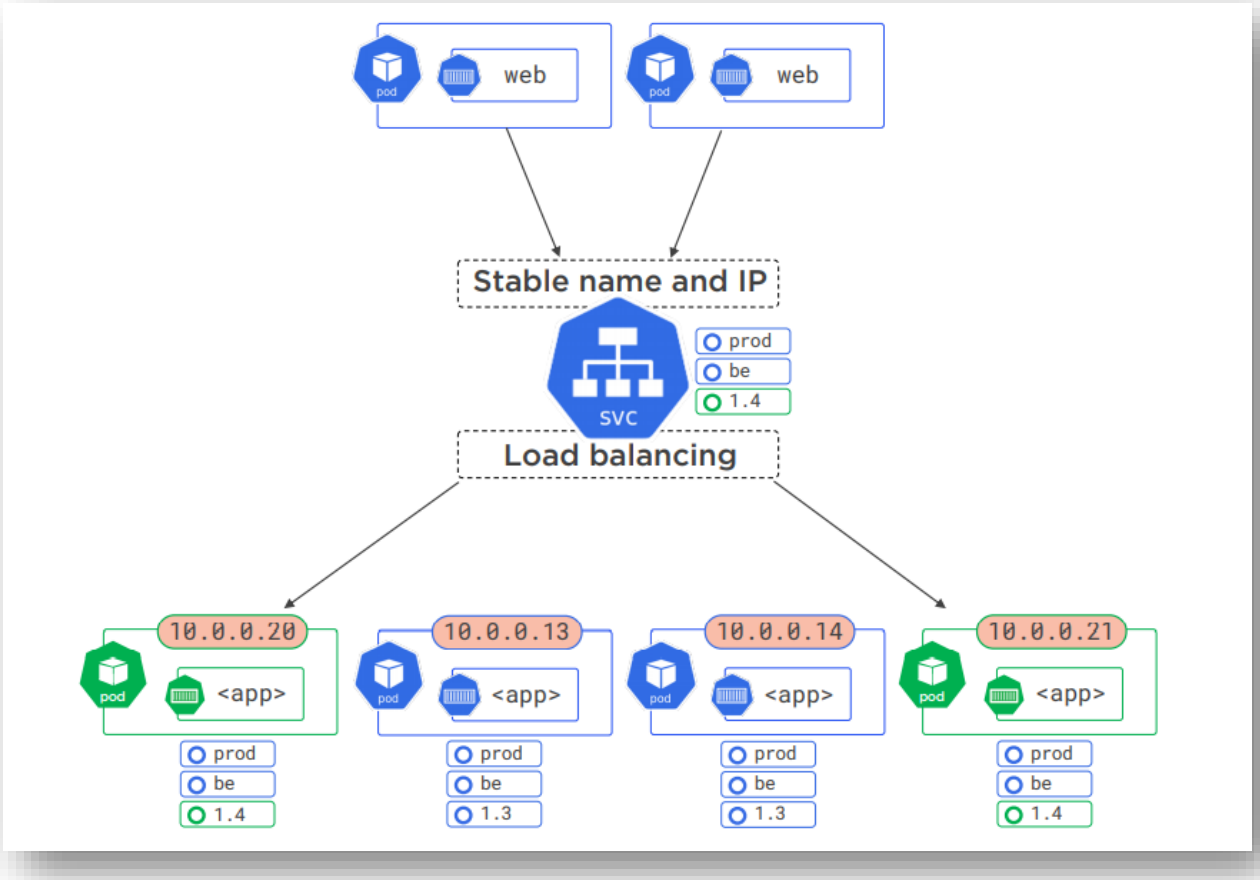
```
apiVersion: apps/v1
kind: Service
spec:
    selector: app: manager-api
    type: LoadBalancer
```

Процесът на внедряване започва с командата *“docker build”*, използвайки Dockerfile. След това Deployment и Service YAML файлове се прилагат с командите *“kubectl apply -f deployment.yaml”* и “*kubectl apply -f service.yaml*”,

което стартира микроуслугите в Kubernetes клъстера.



Deployment и Service ресурсите осигуряват IP адрес и DNS за достъп до микроуслугите, като същевременно балансират трафика между подовете и разпределят натоварването.



Приложение 2. Част от програмния код на C# и .NET при изграждане на системата

1) API контролер за поръчки

```
public class OrderController : ApiController
{
    public OrderController(IResourceMapper resourceMapper, IMediator mediator)
        : base(resourceMapper, mediator)
    {
    }

    /// <summary>
    /// Retrieves a to-go order by id.
    /// </summary>
    [HttpGet("{id}", Name = nameof(GetOrder))]
    [Authorize]
    public async Task<IActionResult> GetOrder([FromRoute] Guid id) =>
        (await Mediator.Send(new GetToGoOrder { Id = id })
            .MapAsync(ToResourceAsync<ToGoOrderView, ToGoOrderResource>))
            .Match(Ok, NotFound);
```

2) Връзка с базата от данни

```
public class OrderRepository : IOrderRepository
{
    public async Task<Either<Unit, Error>> Add(Order order)
    {
        _dbContext.ToGoOrders.Add(order);
        await _dbContext.SaveChangesAsync();
        return Unit.Value;
    }
    public Task<Either<Unit, Error>> Get(Guid id) =>
        _dbContext
            .ToGoOrders
            .Include(o => o.OrderedItems)
                .ThenInclude(i => i.MenuItem)
            .FirstOrDefaultAsync(o => o.Id == id)
            .SomeNotNull();
```

3) Посредник за управление на данни

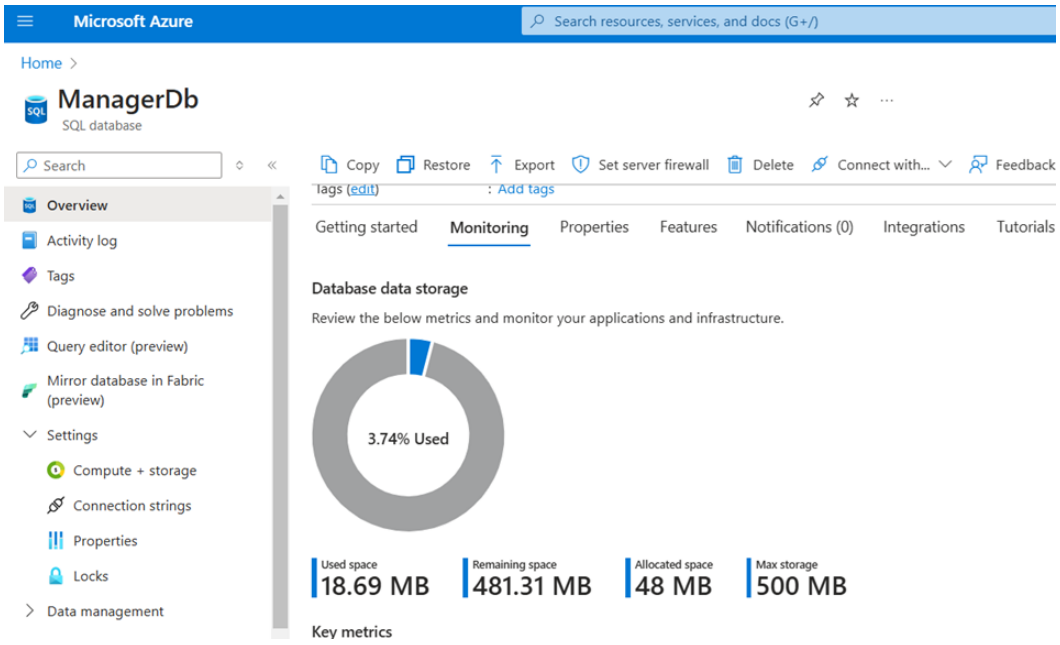
```
namespace Business.OrderContext.CommandHandlers
{
    public class OrderHandler : BaseHandler<Order>
    {
        private readonly IOrderRepository _orderRepository;

        public OrderHandler(
            IValidator<Order> validator,
            IEventBus eventBus,
            IMapper mapper,
            IOrderRepository orderRepository)
            : base(validator, eventBus, mapper)
        {
            _orderRepository = orderRepository;
        }

        public override Task<Either<Unit, Error>> Handle(NewOrder command) =>
            CheckIfOrderIsNotExisting(command).MapAsync(order =>
            MenuItemsShouldExist(command.ItemNumbers).MapAsync(items =>
            PersistOrder(order, items)));

        private async Task<Either<Order, Error>> CheckIfOrderIsNotExisting(NewOrder command)
```

4) Синхронизация с ERP

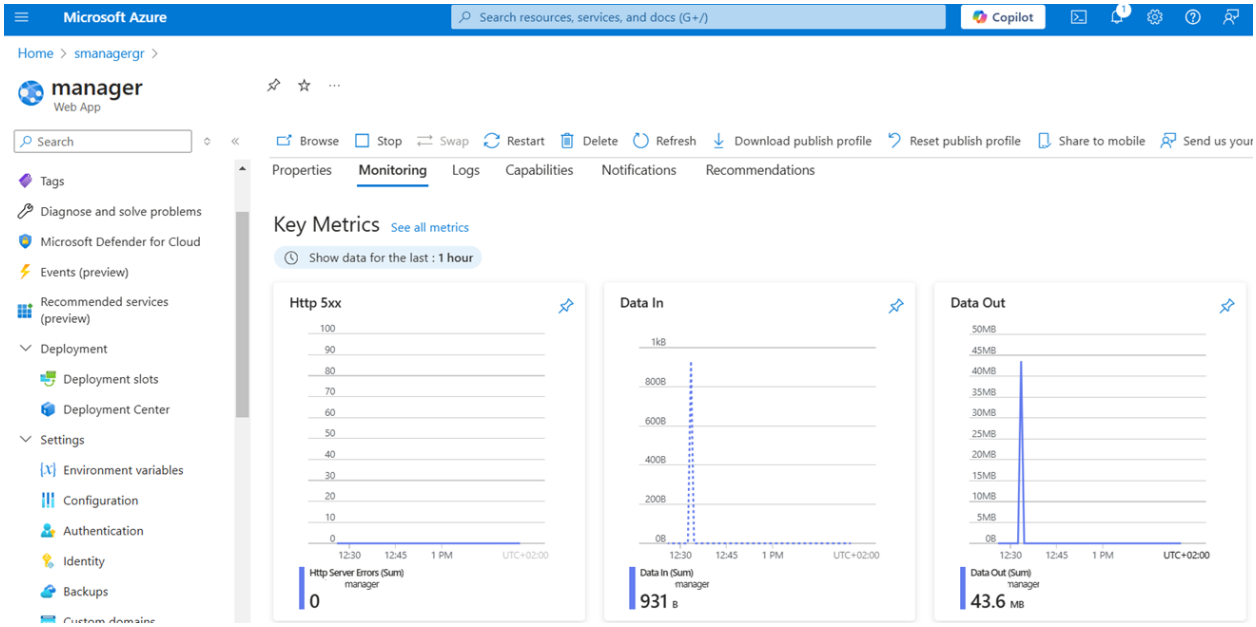
```
public class ErpSynchronization : IErpSynchronization
{
    protected Either<TCommand, Error> ValidateCommand(TCommand command)
    {
        var validationResult = Validator.Validate(command);

        return validationResult
            .SomeWhen(
                r => r.IsValid,
                r => Error.Validation(r.Errors.Select(e => e.ErrorMessage)))
            .Map(_ => command);
    }

    public async Task<Unit> Publish<TEvent>(Guid streamId, params TEvent[] events)
        where TEvent : IEvent
    {
        foreach (var @event in events)
        {
            _session.Events.Append(streamId, @event);
            await _mediator.Publish(@event);
        }
```

Приложение 3. Изображения от облачните услуги на Azure, използвани по време на тестовете

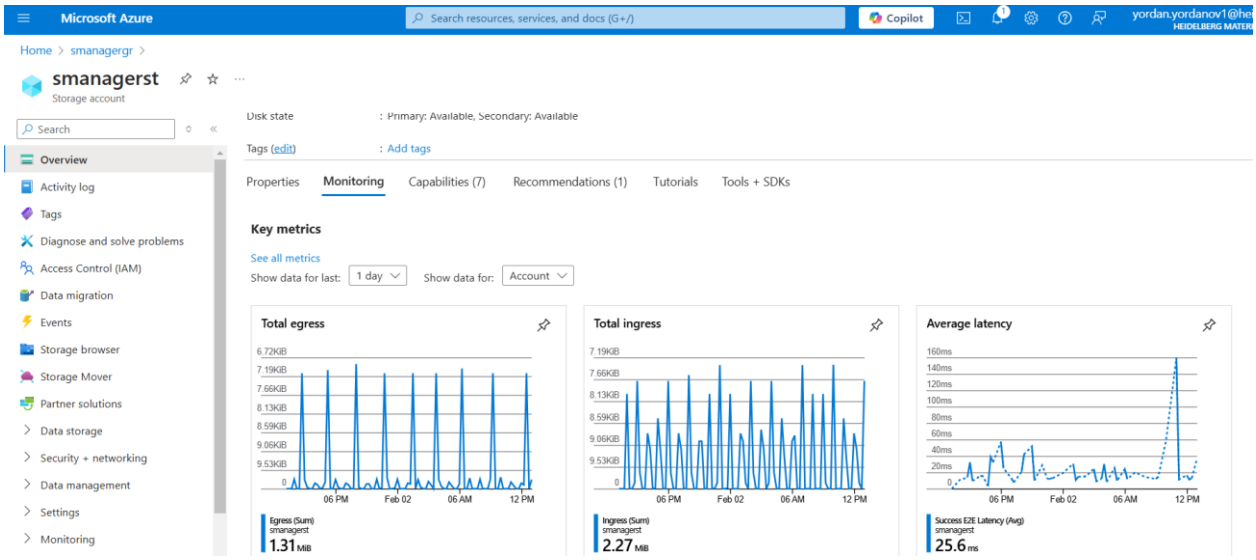
1) База от данни



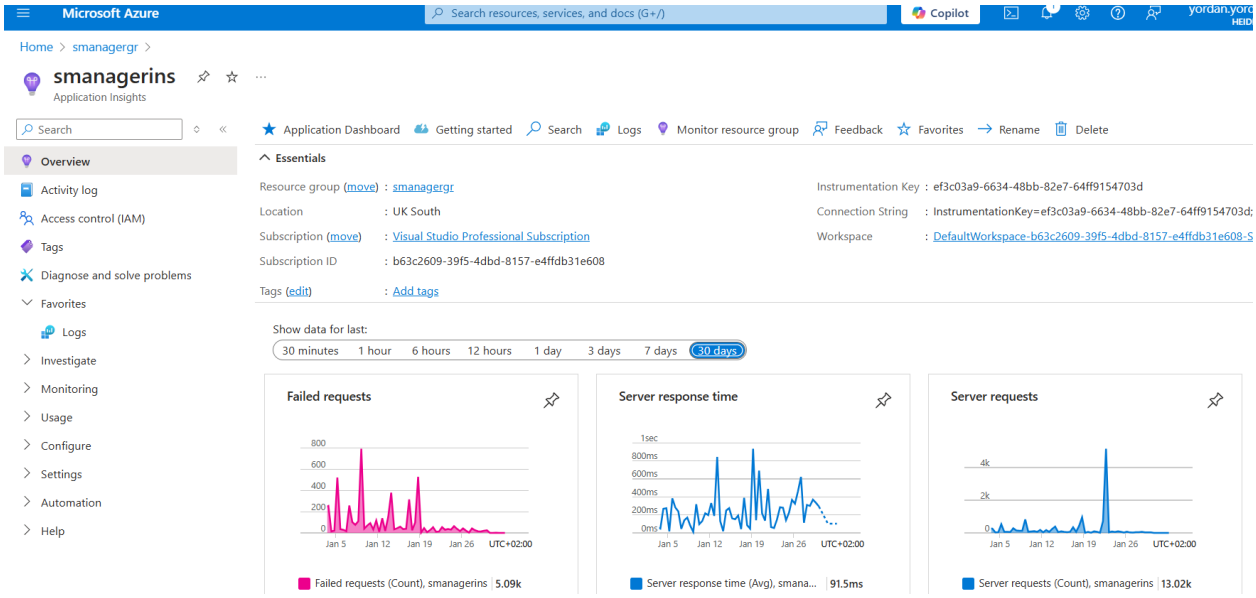
2) Хостинг услуга



3) Хранилище за файлове (Blob Storage)

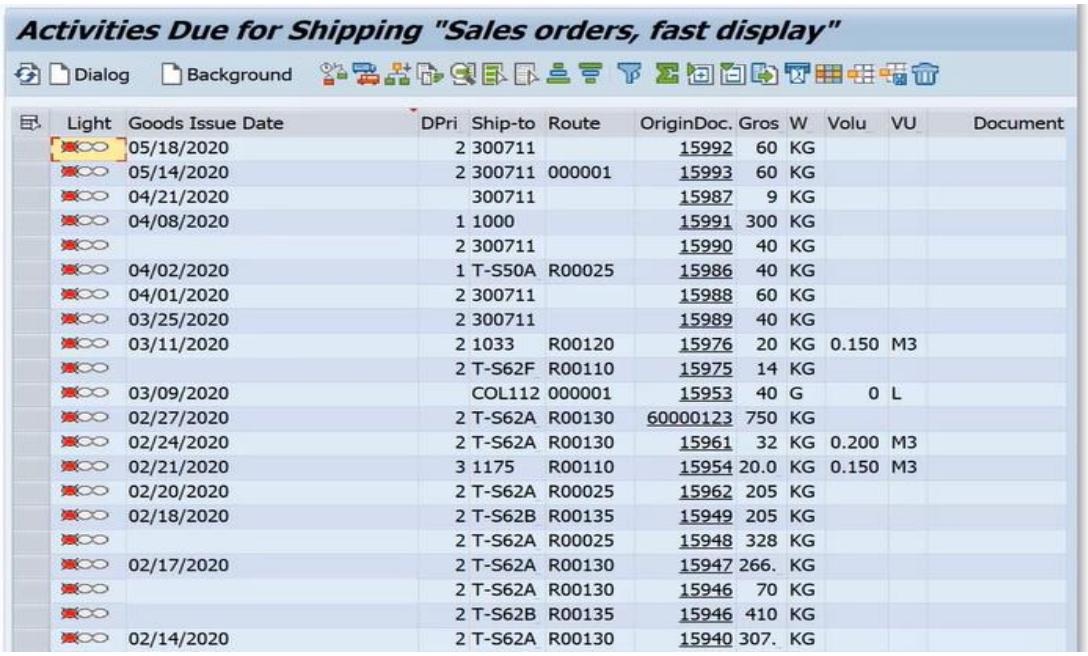


4) Мониторинг



Приложение 4. Изображения от модула на SAP ERP за продажби и дистрибуция, в които се синхронизират промените от облачната услуга

Екран на SAP „Activities Due for Shipping 'Sales orders, fast display“. Използван за проследяване и управление на поръчки, готови за доставка.



Activities Due for Shipping "Sales orders, fast display"

Dialog | Background

Light	Goods Issue Date	DPri	Ship-to	Route	OriginDoc.	Gros	W	Volu	VU	Document
	05/18/2020	2	300711		15992	60	KG			
	05/14/2020	2	300711	000001	15993	60	KG			
	04/21/2020		300711		15987	9	KG			
	04/08/2020	1	1000		15991	300	KG			
		2	300711		15990	40	KG			
	04/02/2020	1	T-S50A	R00025	15986	40	KG			
	04/01/2020	2	300711		15988	60	KG			
	03/25/2020	2	300711		15989	40	KG			
	03/11/2020	2	1033	R00120	15976	20	KG	0.150	M3	
		2	T-S62F	R00110	15975	14	KG			
	03/09/2020		COL112	000001	15953	40	G	0	L	
	02/27/2020	2	T-S62A	R00130	60000123	750	KG			
	02/24/2020	2	T-S62A	R00130	15961	32	KG	0.200	M3	
	02/21/2020	3	1175	R00110	15954	20.0	KG	0.150	M3	
	02/20/2020	2	T-S62A	R00025	15962	205	KG			
	02/18/2020	2	T-S62B	R00135	15949	205	KG			
		2	T-S62A	R00025	15948	328	KG			
	02/17/2020	2	T-S62A	R00130	15947	266.	KG			
		2	T-S62A	R00130	15946	70	KG			
		2	T-S62B	R00135	15946	410	KG			
	02/14/2020	2	T-S62A	R00130	15940	307.	KG			

При избор на един от записите се визуализира екран от SAP, който представя документ за поръчка. Включени различни елементи и данни, свързани с конкретната поръчка.

Display Standard Order 15994: Overview

Orders

Standard Order	15994	Net value	799.90 USD
Sold-To Party	300711	Holden & Associates / 2300 LOOP 410 S.W. / SAN ANTONIO TX 78245	
Ship-To Party	300711	Holden & Associates / 2300 LOOP 410 S.W. / SAN ANTONIO TX 78245	
PO Number		PO date	

Sales | Item overview | Item detail | Ordering party | Procurement | Shipping | Reason for rejection

Req. deliv.date	D 05/27/2020	Deliver.Plant	
Complete dlv.		Total Weight	30 KG
Delivery block		Volume	0.000
Billing block		Pricing date	05/15/2020
Payment card		Exp.date	
Card Verif.Code			
Payment terms	ZB01 14 Days 3%, 30/2%, 45	Incoterms	FOB From the Plant
Order reason			

All items

Item	Material	Order Quantity	Un	Description	S	Customer Material Numb
10	1400-400	10	PC	Motorcycle Helmet - Stand	✓	

Списък с публикации по темата на дисертационния труд

Статии

1. Jordanov, J., Petrov, P., Vasilev, J., Kuyumdzhiev, I. (2025). *Domain-Driven Design in Cloud Computing: .NET and Azure Case Analysis*. TEM Journal. 14(1), pp.44-54. (Scopus)
2. Jordanov, J., Simeonidis, D., Petrov, P. (2024). *Containerized Microservices for Mobile Applications Deployed on Cloud Systems*. International Journal of Interactive Mobile Technologies, 18(10), pp.48-58. (Scopus)
3. Vasilev, J., Petrov, P., Jordanov, J. (2024). *A practical approach of data visualization from geographic information systems by using mobile technologies*. International Journal of Interactive Mobile Technologies, 18(3), pp.4-15. (Scopus)
4. Jordanov, J., Petrov, P. (2023). *Domain Driven Design Approaches in Cloud Native Service Architecture*. TEM Journal, 12(4), pp.1985-1994. (Scopus)

Доклади

1. Йорданов, Й. (2024). *Възможности за рационализиране на бизнес процеси в производствени предприятия чрез прилагане на облачни технологии*. Международна научна конференция "Информационни и комуникационни технологии в бизнеса и образованието" (стр. 169-176). „Наука и икономика“, Икономически университет – Варна.
2. Simeonidis, D., Petrov, P., Jordanov, J. (2023). *Network Intrusion Detection Through Classification Methods and Machine Learning Techniques*. International Conference Automatics and Informatics (ICAI), pp.409-413. (Scopus)
3. Petrov, P., Nacheva, R., Jordanov, J., Dimitrov, G., Bychkov, O., & Petrivskyi, V. (2023). *Historiographical Study of the Evolution of the Geocoding Systems with Equiangular Tessellation*. 2020 4th International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT), 1–5. (Scopus)

Справка за приносните моменти

В теоретичен план:

- Проведено е изследване на основни проблеми, свързани с информационното осигуряване на веригите на доставки. Представени са възможностите за дигитализация и рационализация на бизнес процесите, свързани с управлението на поръчки за продажба от бизнес клиенти чрез персонализирана информационна система
- Представена е същността на облачните услуги, както и подходът за внедряване на бизнес логика чрез ориентиран към домейн дизайн.

В приложен план:

- Създадени са концептуален, логически и комуникационен модел на информационната система, визуално представени с помощта на утвърдени средства. Тези модели оформят архитектурата на системата за управление на клиентски поръчки и поставят основите за нейното последващо реализиране.
- Избрани са софтуерни технологии за физическа реализация на облачната система, като са взети предвид технически характеристики и възможности за интеграция със съществуващите подсистеми на предприятието. Изборът обхваща програмни езици, работни рамки и софтуерни инструменти, които съответстват на специфичните изисквания на проекта.
- Предложен е практически план за реализация на облачната система, който включва описание на различните етапи по внедряване и тестване на облачната система.
- За демонстрация на приложимостта на системата е избрана стратегия за A/B тестване в производственото предприятие „Хейделберг Цимент Девня“ АД. Апробация на тази стратегия е осъществена чрез ръчни и автоматизирани тестови процедури, които симулират потребителското поведение във временно създадена облачна среда.