|  |  |
| --- | --- |
|  | 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

СЪДЪРЖАНИЕ

[Списък на използваните съкращения 4](#_Toc194915861)

[Въведение 5](#_Toc194915862)

[Глава 1. Проблеми на информационното осигуряване при управление на поръчките от клиенти 11](#_Toc194915863)

[1.1. Управление на веригите от поръчки и доставки чрез корпоративни системи за планиране на ресурси 11](#_Toc194915864)

[1.2. Рационализиране на процесите по управление на поръчките чрез персонализирана информационна система, конфигурирана към конкретна компания 27](#_Toc194915865)

[1.3. Възможности за централизация на процесите по управление чрез прилагане на облачни технологии 37](#_Toc194915866)

[1.4. Управление на бизнес процесите чрез ориентиран към домейн дизайн 52](#_Toc194915867)

[Глава 2. Архитектура на облачна система за управление на поръчки от клиенти 64](#_Toc194915868)

[2.1. Концептуален модел на облачната система за управление на поръчките 64](#_Toc194915869)

[2.2. Логически модел на облачна система за управление на поръчки 74](#_Toc194915870)

[2.2.1. Модули за управление на поръчки и доставки 75](#_Toc194915871)

[2.2.2. Декомпозиция на модулите за поръчки и доставки на ниво микроуслуги 82](#_Toc194915872)

[2.2.3. Модул за управление на потребителските профили 87](#_Toc194915873)

[2.3. Комуникационен модел между модулите 95](#_Toc194915874)

[2.4. Функционалност и потребителски интерфейс 102](#_Toc194915875)

[Глава 3. Изграждане и използване на облачна система за производствено предприятие 113](#_Toc194915876)

[3.1. Обща характеристика на дейността на компания „Хейделберг Цимент Девня“ АД 113](#_Toc194915877)

[3.2. Избор на технологични средства за реализация на системата 121](#_Toc194915878)

[3.3. Физическа реализация на системата 134](#_Toc194915879)

[3.4. Приложение на системата чрез технологичните средства за реализация 142](#_Toc194915880)

[3.4.1. Тестване на облачната система 142](#_Toc194915881)

[3.4.2. Системен мониторинг 147](#_Toc194915882)

[3.4.3. Изчисляване на разходите за използване на облачна услуга 150](#_Toc194915883)

[Заключение 153](#_Toc194915884)

[Използвана литература 158](#_Toc194915885)

[Приложения 176](#_Toc194915886)

# Списък на използваните съкращения

|  |  |
| --- | --- |
| Съкращение | Пълно наименование |
| ПОСУП | Персонализирана облачна система за управление на поръчките |
| ACID | Atomicity, Consistency, Isolation, Durability |
| API | Application Programming Interface |
| AMQP | Advanced Message Queuing Protocol |
| CQRS | Command Query Responsibility Segregation |
| CRUD | Create, Read, Update and Delete |
| CQS | Conveyancing Quality Scheme |
| DDD | Domain Driven Design |
| ERP | Enterprise Resource Planning |
| ES | Event Sourcing |
| ESB | Enterprise Service Bus |
| 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 |
| SOAP | Simple Object Access Protocol |
| SOA | Service Oriented Architecture |
| TMS | Transport Management System |

# Въведение

През последните години производствените компании все по-често се сблъскват с проблеми, свързани с обработката на поръчки за продажба от бизнес клиенти, в рамките на веригите си на доставки. Въпреки използването на съвременни системи за управление на ресурси и информационна логистика, динамика на процесите в тази област изисква постоянни подобрения. Проблемите във веригите на доставка са разнообразни и обхващат управлението на множество клиентски поръчки и съответните 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]](#footnote-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. Например, при интеграцията на вътрешни корпоративни системи за планиране на ресурсите с външни бизнес платформи могат да се появят затруднения с преноса и формата на данни, както и 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. Освен това, производствените предприятия и техните търговски организации са длъжни да се съобразяват със законите и стандартите на държавата, в която работят. Това изисква както добро познаване на нормативните актове, така и тяхното интегриране в информационните системи.

**Актуалността на изследваната тема** е обуслoвена 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 все повече компании използват възможностите, предоставени от облачните платформи, за реализиране на иновативни идеи и подобряване на конкурентоспособността си. В тази връзка, настоящото проучване разглежда проблемите и решенията, свързани с внедряването на облачни информационни системи, както и съпътстващите ги технологични аспекти.

Исторически погледнато, логистиката и управлението на веригата на доставки се основават на сложни и често ръчно управлявани процеси, които включват множество участници – от диспечери и доставчици до крайни потребители. Макар през 90-те години на миналия век да се появяват системи за планиране на ресурсите на предприятието (Enterprise Resource Planning – ERP) и за управление на веригата на доставки (Supply Chain Management – SCM), те функционират отделно от останалите корпоративни подсистеми, което затруднява комуникацията и обмена на информация между различните отдели и партньори. Тази изолация възпрепятства надграждането на процесите по приемане и предоставяне на информация и създава бариери пред ефективното управление на бизнеса.

В началото на 2000-те години водещи технологични корпорации като Amazon, Microsoft и Google полагат основите на облачните услуги, разработвайки мащабируеми инфраструктури за съхранение и обработка на данни в отдалечени центрове. Amazon Web Services (AWS), стартирал през 2006 г., е един от първите доставчици на облачни услуги, които предоставят достъп до виртуални уеб сървъри. Това позволява 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 (Stoyanova, 2023). Облачни информационни системи, като SAP S/4HANA, Oracle Fusion, Microsoft Dynamics 365 и Blue Yonder, предоставят набор от възможности за оптимизиране на процесите по управление на поръчките, транспорта и дистрибуцията, като същевременно 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“ (от английски Software as a Service - SaaS) откриват нови възможности за дигитализация на бизнес процеси в логистиката (Петров и др, 2020). Интегрирането на тези модели премахва необходимостта от поддържане на локална инфраструктура, характерна за така наречените „on-premise“ услуги, и предоставя на компаниите възможност да внедряват софтуерни продукти, съобразени с техните конкретни нужди. Гъвкавостта на тези модели позволява на производсвените 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.

**Тезата**, 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 система (custom business system), 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), персонализираната система е специално разработена и конфигурирана, за да отговаря на конкретните нужди и процеси на дадена компания, като същевременно се интегрира със съществуващи или нововъведени подсистеми. Макар първоначалната инвестиция да е сравнително по-голяма, персонализирането намалява зависимостта от външни доставчици и осигурява непрекъснат контрол върху развитието и поддръжката на системата.

В тази връзка, 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 допълнителни аспекти остават извън неговия обхват.

**Обект** на изследване са процесите във веригите на доставки в производствено предприятие, което 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 им до крайния потребител. Наред с това се изследва практическото прилагане на облачни технологии за управление на поръчки, логистични операции и управление на веригите на доставки, като се приемат специфичните изисквания на бизнес клиентите и оптималното използване на ресурсите.

**Предмет** на настоящото изследване са методите за рационализация и автоматизация на бизнес процеси, реализирани посредством съвременните възможности на облачни платформи и технологии. Въз основа на методите за рационализация и различни архитектурни подходи се разработва персонализирана информационна система, която се приспособява към променящите 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 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. 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.
   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 и да отговарят 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 конкретно 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.

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

Основната задача на настоящата глава е да се проучат теоретичните основи и проблеми на информационното осигуряване при управлението на поръчки от клиенти. Анализират се същността и принципите на веригата на доставки, както и тяхното интегриране в корпоративнте 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. В тази връзка, 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 крайния потребител, докато обратната логистика представлява процеса на движение на стоките в обратна посока - от крайния потребител 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]](#footnote-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 таб. 1.1 са сравнени характеристиките на правата и обратната верига на 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 качесво 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, правите вериги на доставки поставят акцент върху движението на продукти от производителите към крайните потребители, с цел максимизиране на печалбата и минимизиране на разходите, докато обратните вериги на доставки проследяват 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 действат в тясна координация.

Важен елемент в управлението на веригата на доставки е логистиката (Василев, 2015). Европейската логистична асоциация[[3]](#footnote-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*“. Тези дейности се характеризират с различни параметри, като 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). В тази смисъл, авторският колектив Василев, Николаев и Милкова (2023) разглеждат 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, като 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 (Cълова, 2020; Molamohamadi et al., 2021). В тази връзка, логистичният мениджмънт включва вземане на стратегически, тактически и оперативни решения, свързани с развитието на логистичната дейност и взаимодействието с доставчиците и клиентите по веригата на доставки. Sandberg (2025) допълва, че стратегическият логистичен план има за цел да реализира поставена стратегия и да осигури функционирането на логистичната мрежа. Освен това, 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 регламентиране и 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 важно да се разгледат различните стратегии за управление, които представляват 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, осигуряване 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, като същевременно намаляват излишните запаси и разходи. Това в крайна сметка носи ползи както за клиентите, така и за бизнеса.

В обхвата на научния труд разглеждаме същност и основни компоненти на стратегията за доставка на продукция „точно навреме“ за продажба (от английски: just in time). Други стратегии остават извън обхвата на труда, тъй като тяхното включване би изисквало разглеждането на множество допълнителни термини и технологии. Избраната стратегия 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). Важно е да се отбележи, че съществуват и други компоненти, които също оказват влияние върху SCM, но те не са разгледани в настоящия труд.

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 позволява интегрирането на различни процеси и операции в единна платформа. Използването на ERP системи в SCM включва оптимизация на инвентара, планиране на производството, управление на складовите запаси и логистика (Димитров, 2020). Също така, ERP подкрепя сътрудничеството между различните отдели в организацията, както и с външни партньори. ERP системите предоставят възможности за автоматизация на множество рутинни задачи, което намалява човешките грешки (Vasilev & Stoyanova, 2019). В тази връзка, фиг. 1.1 представя модел на взаимосвързаността между основните компоненти на SCM стратегията за доставка на продукция. Моделът е базиран на теоретичната рамка, анализирана в разгледаните проучвания.

A diagram of a company

Description automatically generated

**Фиг. 1.1. Модел на взаимосвързаността 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 ръководи цялостната посока на процесите. Съществуват множество софтуерни подсистеми, които подпомагат тази част от модела, като например софтуер за бизнес анализ (Verdouw, 2010). Въпреки това, крайните резултати следва да се регистрират в ERP системата. Следва планиране, при което избраната стратегия се превръща в конкретни действия (Sánchez-Flores et al. 2020). Планирането на веригата на доставки служи като стратегическа рамка за операциите, съгласувайки търсенето с производствения капацитет и определяйки графика за доставка на готови продукти (Chiang et al. 2021; Lee et al. 2022). В този контекст, ERP системите управляват информационните потоци за поръчки, доставки и движения на продукти. Това е основата на текущото проучване и затова следва подробно да се изследва.

В описания по-горе модел 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 предоставят информация за суровини, прекъсвания и работни инструкции (Văcar, 2019). Те също са свързани с ERP и 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 (на 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. Управлението на околната среда регистрира данни за емисии на 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“ (Internet of Things – IoT) и услуги за машинно обучение (Sazanavets, 2022; Сълова, 2023). 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: идентификация на превозни средства като регистрационен номер, нива на гориво, 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, тол такси).

Когато се интегрират с ERP системите, 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, като 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.

В този смисъл, същността на ERP системите 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 контекста на технологията са предварително зададени стъпки и правила, които позволяват оптимизация и автоматизация на процесите (Radev, 2023). ERP модулите са получили нормативна регламентация, което гарантира тяхната ефективност и съответствие с международно признати стандарти. Примери са Material Requirements Planning[[4]](#footnote-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]](#footnote-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 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 контекста на настоящото изследване терминът „SAP“ се отнася до софтуера SAP S/4HANA. SAP предоставя 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 нуждитна 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. Освен това 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. Въпреки това, този корпоративен софтуер предоставя инструменти, които автоматично получават, съгласуват и извършват определени действия. 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 предоставят 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, като 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.

Значението на основните и трансакционните 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 акцентират върху 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“. Тези организационни единици осигуряват рамката, в която модулите функционират.

Според Bilovodska et al. (2018), мастър данните са основни градивни елементи за транзакционите данни. Мастър данните остават относително статични, докато данните за транзакциите, като 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-високо ниво. 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. Това е 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. В този контекст, 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 - включително 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 доставките, 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 на това 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. Компаниите 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. Освен това глобализацията създава непредсказуемост в търсенето на продукти, като дружествата са принудени да реагират на сезонни промени или икономически кризи (Василев, 2017).

При въвеждането на ERP системи, обикновено се поставя акцент върху управлението на вътрешните операции и ресурси, 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. От своя страна, SCM системите предлагат 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. Тези проблеми могат да станат критични при значителни промени в търсенето или затруднения с доставките. Подобна липсата на интеграция между ERP и SCM създава необходимост от персонализирана система, разработена и конфигурирана спрямо нуждите и изискванията на конкретно производствено предприятие. Тази система обединява данни и процеси от основни компоненти на веригата на доставки.

## 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 контекста 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, предоставящи възможности за анализ и автоматизация на бизнес процеси.

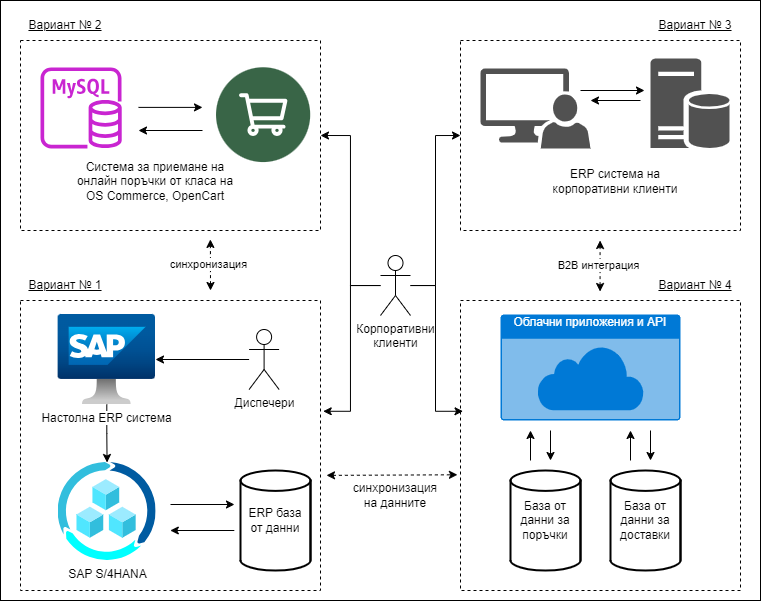
Oracle NetSuite представлява изцяло облачна ERP платформа, подходяща за предприятия с различен мащаб, която предлага набор от функционалности, сред които модули за финанси, CRM и електронна търговия. JD Edwards EnterpriseOne, също разработка на Oracle, може да бъде внедрена както локално, така и в облак, и е приложима в редица отрасли и бизнес процеси. Microsoft Dynamics 365 интегрира ERP и CRM функционалности, като 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 осигурява оперативна съвместимост и позволява надграждане на съществуващи ERP и CRM системи.

В допълнение, софтуерни продукти за онлайн поръчки като OS Commerce и OpenCart служат като инструменти за електронно управление на търговските процеси в производствените предприятия. Тези платформи предоставят възможности за приемане, обработка и изпълнение на поръчки, управление на продуктовия каталог, обработка на плащания и автоматизирана комуникация с бизнес клиентите (Sulova, 2023).

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 предназначение.

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. Тази констатация е свързана с 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.

Систематичният анализ 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 като част от SCM стратегията за доставка на готова продукция. Прегледът 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 като 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 тези 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, като данните се съхраняват в нейната база от данни. 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 клиентите.

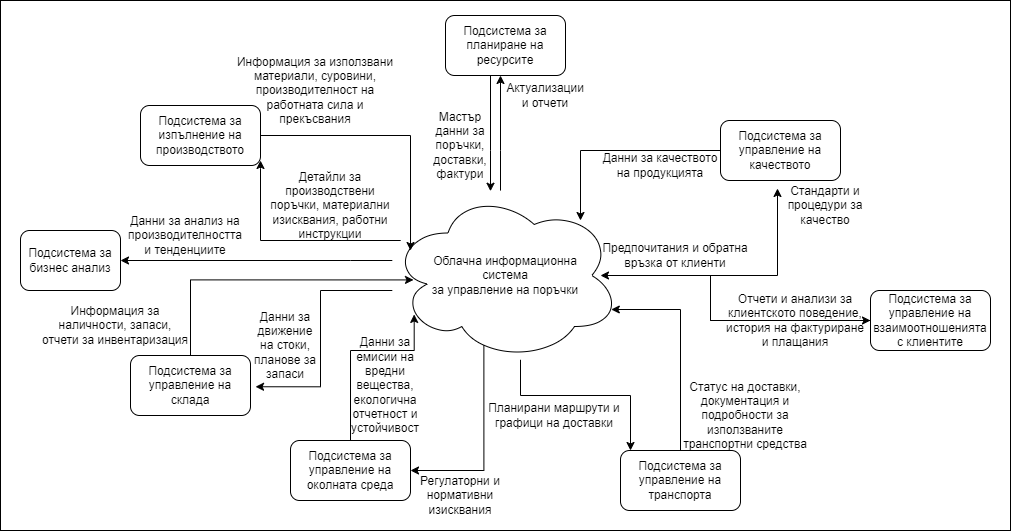
Във втория вариант се интегрира платформа за електронна търговия от типа на OpenCart или OS Commerce. Тези платформи работят на отделни сървъри, със собствени бази от данни (например MySQL), които трябва да се синхронизират със съответните бази от данни на ERP. Платформата за електронна търговия позволява на бизнес клиентите сами да въвеждат поръчките, като по този начин облекчава натоварването на диспечери и оператори. Въпреки това, този подход изисква допълнителна поддръжка и е ограничен от възможностите на самата платформа за електронна търговия. Освен това е необходимо 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 осигурява 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.

Интеграцията на външни партньори в корпоративната мрежа налага прилагане на многостепенна защита. Въпреки това, 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. Всички тези аспекти оказват негативно въздействие върху ИТ отдела, който носи отговорност за осигуряване на сигурността и поддръжката между различните партньорски 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, предлагащ адаптивност и интеграция с различни видове външни и вътрешни системи.



**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, представяща 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, като същевременно позволява приемането на заявки и предоставянето на данни към бизнес клиентите чрез мобилни приложения. Тази система е проектирана да консолидира данни от подсистемите на компонентите на SCM стратегията, изложени в таблица 1.2 от преходната точка, както и да приема и предоставя информация на крайните потребители.

Проектирането, също така, следва да се съобрази с изискванията на конкретното предприятие, като се вземат предвид 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 крайния потребител. Централизираното информационно управление стои в основата на системата, което позволява рационализация и подобряване на достъпа до актуална информация (Сълова, 2019).

Подсистемите обхващат различни аспекти, от управление на вътрешнофирмени ресурси, като логистика и складови наличности, до външни взаимодействия, като клиентски връзки и контрол на качеството на продуктите. Тези подсистеми са взаимосвързани така, че да осигуряват постоянен поток на информация в организацията (Aleksandrova, 2021; Sullivan & Kern, 2021).

Както бе отбелязано, ERP осигурява 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 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, като предоставя бизнес метрики, които могат да подпомогнат стратегическото развитие (Ren et al., 2019; Schniederjans et al., 2020). Тя също така предоставя прогнози, които подпомагат вземането на бизнес решения (Ramakrishna, 2022). Подсистемата за мониторинг на производството предоставя възможности за наблюдение и контрол на производствените процеси, информирайки останалите подсистеми за състоянието 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 като SAP Netweaver Gateway[[6]](#footnote-6) (Bönnen et al. 2018). Основната цел на тази система 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.

Освен това, персонализираната система позволява внедряването 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, като 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, предоставят на 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, като 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 начин се осигурява защита на вътрешните подсистеми, като се избягва 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. За тази цел процесите по пренасочване следва 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 са предоставени като план-график на диспечерите.

За тази цел, в реално време се анализират данните, получени от IoT устройства, прикрепени към превозните средства, които предоставят постоянна обратна връзка относно местоположение, скорост и състоянието 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.

Определено производствено предприятие може да комуникира с клиентите си чрез имейли, телефонни обаждания до центъра за обслужване, мобилни и уеб приложения, за да улесни достъпа до актуална информация и да осигури бърза обратна връзка. Ако тези канали за връзка не са интегрирани, могат да възникнат определени проблеми, като например жалба, подадена по имейл, да бъде пропусната от диспечера на смяна. Централизираната система позволява интегрирането на различните комуникационни канали посредством т.нар. „омниканален“ (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, неинтегрирани канали. В статията също така се подчертава и ролята на изкуствения интелект и машинното обучение в обслужването на клиенти.

В тази връзка, чрез персонализираната система могат да се внедрят алгоритми за извличане на съдържание от входящи имейли. Тези данни следва да бъдат автоматично въведени в системата, като всеки запис 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, като по този начин се избягва ръчно въвеждане. Системата автоматично генерира и изпраща отговор на имейла, като добавя хипервръзка към мобилно приложение, което да предоставя информация за конкретните поръчки и доставки в реално време.

Освен това, въз основа на емпирични данни като история на поръчките, свързанатa информация за доставки и алгоритми на изкуствения интелект от типа „обучение с утвърждение“, могат да се създадат модели на поведение на бизнес клиентите (Сълов, 2023). Тези модели позволяват 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, като се вземат предвид фактори като взаимоотношения, търсене и предлагане. Друг аспект 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, като 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*“ (agile) подходи, които позволяват 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 като 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, като се позволява надграждане и интегриране на нови функционалности в отделни части 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. Облачните услуги предоставят възможност за разработване и внедряване на информационна система, в която множество участници и подсистеми във веригата на доставки могат да работят съвместно и координирано. Софтуерната архитектура на това софтуерно решение позволява обмен на данни в реално време и гарантира контролиран достъп до основна информация.

В този смисъл, 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. Облак и облачните изчисления обхващат модела за предоставяне на изчислителна мощ „*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, която осигурява физическата и виртуалната среда за работа на облачните услуги.

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

Според дефиницията на организацията *Cloud Native Computing Foundation*[[8]](#footnote-8) (2018) „облачните технологии позволяват 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 критично значение за разнообразни бизнес единици. Автоматизирането на процесите дава възможност на инженерите да внедряват софтуерни промени с минимални усилия“*.*

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

Производителността измерва времето между Инернет 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 е интервалът от време, в което потребителят изчаква, за да види резултат след започване на действие (Esposito, 2016). *Времето за обработка* е времето, необходимо на системата за изчисляване на резултата след получаване на заявката. То включва задачи като заявка към база от данни, обработка и връщане на резултата към потребителя, като действия, които системата извършва, за да изпълни заявката. *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, като 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 подсилване на физическия хардуер (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 осигурява висока достъпност и толерантност към грешки.

В тази връзка, 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 доставчици предлагат услуги с 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 (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 аспекти, например време за реакция под 100 ms за 90-ия процент. В допълнение, индикаторът за ниво на обслужване (Service Level Indicator - SLI) е количествена мярка за определяне на съответствието със SLO. Той предоставя данните 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 до тук, можем да обобщим, че облачните технологии осигурят инфраструктура за приложения, предлагайки ресурси като сървъри, операционни системи, защитни стени, 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 позволява 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 позволява 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 позволява 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 |
| **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 позволяват 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]](#footnote-9). В тази връзка, предимствата на публичните облачни услуги надхвърлят потенциалните недостатъци.

В допълнение, 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 като виртуални машини и мрежи. При IaaS моделът се предоставя отделна физическа среда, освобождавайки компаниите от пряко управление на слоевете за виртуализация, като същевременно запазват контрол върху операционните системи и приложения. Надграждайки над IaaS, при PaaS доставчикът на облачната услуга отговаря за ОС, междинния софтуер и среди за изпълнение, като по този начин позволява на разработчиците да се съсредоточат единствено върху създаването и внедряването на приложения. PaaS улеснява процеса на разработка, като предоставя разнообразни инструменти и услуги за създаване на софтуерни решения. В третия SaaS модел се предоставят цялостни приложения, достъпни през интернет. Благодарение на този модел, разработчиците не се грижат за инсталация, поддръжка или управление на софтуера, тъй като всичко това се поема 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 модели като IaaS, PaaS и SaaS, може да подобри капацитета за обработка на данни, да повиши сигурността и да намали техническите разходи. Всеки от тези модели предлага различно ниво на контрол и сложност на управление, което ги прави подходящи за различни организационни изисквания и възможности, представени на фиг. 1.4.



**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, като се определя 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 позволява на организациите да плащат само за ресурсите и лицензите за софтуер, които действително използват, подобрявайки предвидимостта и управлението на разходите. Важно е да се отбележи, че при преминаването от локални центрове към IaaS, PaaS или SaaS, компаниите могат да се сблъскат с предизвикателства, 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), предоставят защитна мрежа, която позволява 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 модели на облачни услуги абстрахират различни аспекти на инфраструктурата, като могат да бъдат надградени с междинните модели Containers as a Service (CaaS) и Function as a Service (FaaS).



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

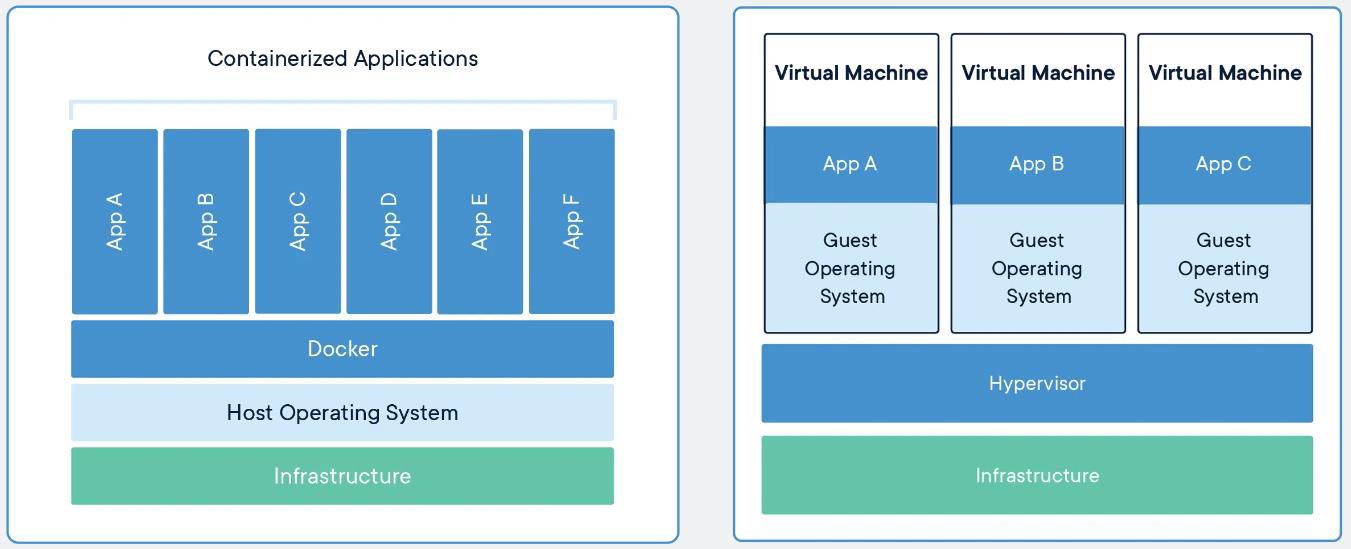
Източник: Garverick и McIver, 2023; Likness & Phillip, 2024.

Описани от Garverick и McIver (2023) като „изчисленията без сървър“ (от английски - serverles), CaaS и FaaS представляват друг аспект на облачните услуги, при който отделни функции се изпълняват в отговор на конкретни събития. Този модел допълнително намалява необходимостта от управление на ресурси, като предоставя икономически ефективна опция за приложения, които се нуждаят от специфична функционалност. Както е илюстрирано на фиг. 1.5, CaaS осигурява контейнерни услуги, предлагащи платформа за разработване, управление и изпълнение на контейнери (Likness & Phillip, 2024). FaaS е модел за облачни услуги, който позволява 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 работещa 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. В тази връзка те 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). Това е проект с отворен код за автоматизиране на внедряването на приложения като преносими, „самодостатъчни контейнери“, които работят еднакво както локално така и в облака. Docker контейнерите могат да работят върху Linux или Windows, като 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.

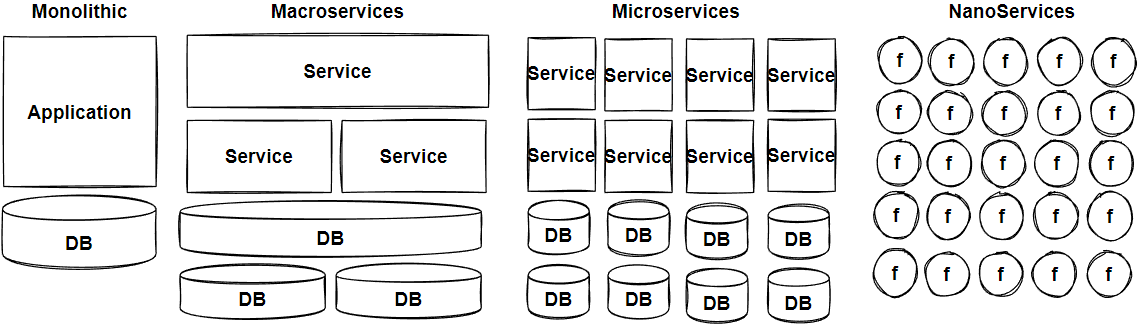
Docker контейнерите включват приложението и всички негови зависимости, като споделят ядрото на операционната система с други контейнери и функционират като изолирани процеси. Изключение правят Hyper-V контейнерите, при които всеки контейнер работи в отделна виртуална машина. Контейнерите са основен инструмент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

Един от най-популярните оркестратори на контейнери е Kubernetes, който предоставя набор от функции за автоматизация на внедряването и управлението на контейнеризирани приложения (виж приложение 1). Kubernetes осигурява 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 отварят през браузър (Nacheva, 2020). Микроуслугите представляват метод за разработване на сървърни приложения чрез използване на множество малки подуслуги. 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, като 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 предоставя 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), дванадесетфакторната методология осигурява основа за изграждането на облачни системи, тъй като може да се приложи към всякакъв тип уеб, настолни или мобилни приложения. Системите, изградени по тези принципи, могат да се внедряват и мащабират, като същевременно позволяват добавяне на нови или промяна на съществуващи функционалности.

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, разположенo 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. |
| **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 кеш, 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 минимизирането 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 тези 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, отразяващи най-съвременния дизайн на облачните приложения (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 удостовереният обект. |

Адаптация по: Hoffman 2016.

В допълнение, Karthikeyan (2021) предоставя набор от ръководни принципи, които се използват за подобряване на качеството на работното натоварване, което е показано в табл. 1.7 и представя пет стълба на т.нар. „добра архитектурата“.

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 позволяват 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 налага 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. В тази връзка, Evans (2004) описва 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 позволява ясно 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. Също така, 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 предоставя 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 англиийски 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). Във връзка с това, Fowler (2019) сравнява DDD с дизайна, управляван от данни, в контекста на времето и сложността при разработка на софтуер. Това сравнение е изобразено като диаграма на фиг. 1.8.

A diagram of a design pattern

Description automatically generated

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

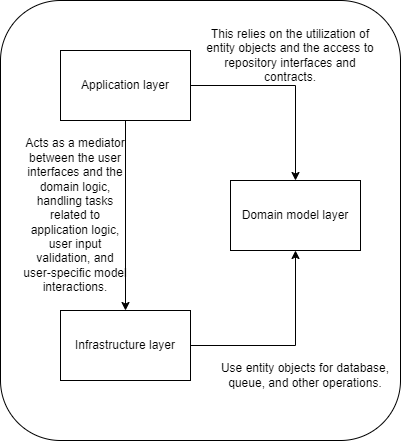
Източник: Fowler, 2019.

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 се развива постоянно. Това допринасят за поддържане на качеството на софтуерната архитектура. В този смисъл, сложността на бизнес логиката представлява индикатор за сложността на проблемната област, която софтуерът е предназначен да управлява. 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. DDD включва процесите, свързани с комуникация между отделните услуги, ефективна обработка на информацията и осигуряване на висока производителност. Тези аспекти са ключови за цялостното функциониране на системата и ще бъдат разгледани в следващата глава на дисертацията.

Основна характеристика на DDD е улесняване на комуникацията между експертите по домейна и софтуерните инженери, като се дефинира общ, **универсален език** (UL). Това е инструмент, който помага за обединяване на дизайнери и програмисти, така че те да могат да създадат модели на домейна и да ги приложат в практиката. Когато кодът е написан чрез UL, той може да даде индикатори за случаи и изисквания, които не са били достатъчно ясни 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 заинтересовани страни. Batista (2022) изтъква необходимостта от универсалния език за предотвратяване на недоразумения и неправилни предположения. UL се използва в различни аспекти при разработката на софтуер, включително в документацията, комуникацията между екипите, кода на приложението и кода за тестване. UL се развива и се поддържа с течение на времето, като предоставя 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]](#footnote-10). Те са отговорни за изпълнение на действия, които променят данни или файлове. Заявките се използват за извличане на информация, като предоставят данни, но без да променят стойностите.

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



**Фиг. 1.10. Надграждане на трислойния архитектурен модел**

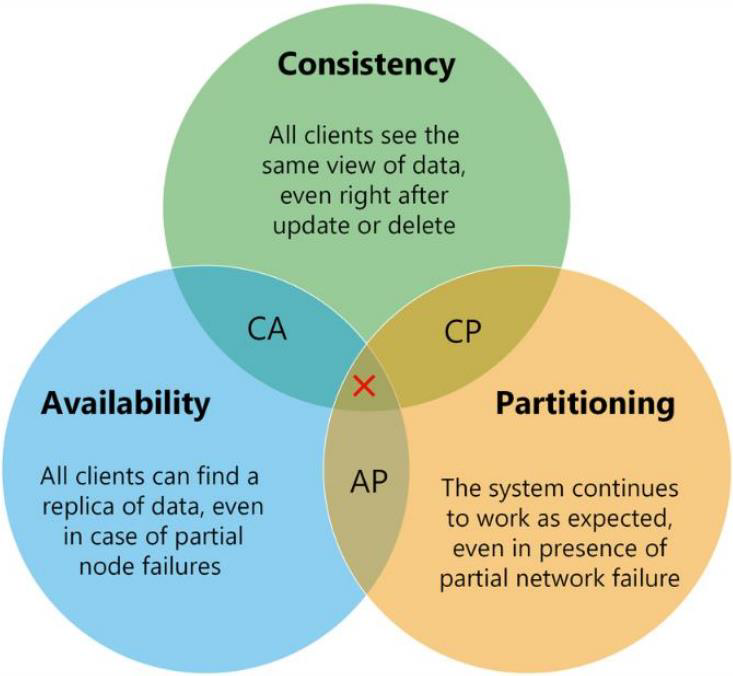
Източник: Vettor, Molenkamp & van Wijk, 2024.

Един от аспектите 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 тези действия 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“. В този контекст, 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 тези два типа са регистрирани като „*събития*“. Командите активират процеси в домейн моделите на системата, докато събитията са резултат от тези процеси. Именуването на тези операции 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 гарантира едновременно всички от следните 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 осигуряват съгласуваност на данните чрез обработка на команди. CQRS e 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 комбиниране на тези подходи производствените организации 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 акцентират върху 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 тези модели е подобряване на достъпа до актуална информация и надграждане на текущите ERP, CRM, TMS подсистеми на предприятието. Приемаме че, интегрирането на тези подсистеми е от основно значение за дигитализацията на бизнес процесите.

В допълнение са представени основните характеристики на облачните технологии, които позволяват 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 патрньори 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 осигуряват 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. Освен 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, следва да бъдат 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 идентифициране на основните бизнес сценарии, водещи до преглед от високо ниво. Вторият етап обхваща разработването на логически модели, включително ER диаграми (Entity Relationship Diagrams), диаграми на последователности и дейности. Тези диаграми представят информационните потоци, основните процеси и взаимодействията между компонентите в системата. Последният етап обхваща аспектите на взаимодействието на системата с крайните потребители, включително дефиниране на функционалности и интерфейси. Това позволява очертаването 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 важно да се представи модел на **прогнозиран растеж на системата**[[11]](#footnote-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 като 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.



**Фиг. 2.2. Модел на зрялост на архитектурата**

Адаптация по: Stuckenberg, 2014.

В първото ниво, наречено „при поискване“, всяка организационна единица[[12]](#footnote-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, като 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 сложна за изпълнение. За да се преодолеят тези проблеми, третото и четвъртото ниво предоставят по-висока степен на конфигурация, като на тези нива 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. В тази връзка 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 позволява множество клиенти на различни търговски организации да използват една и съща версия на системата в обединена инфаструктура. По този начин оперативните разходи могат да бъдат разпределени между отделните организации на производственото предприятие. Също така, 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. Тези механизми осигуряват оперативна достъпност дори при високо натоварване.

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, като „управление на потребителски акаунти“, „управление на поръчки от клиенти“ и „управление на доставки до клиенти“. Освен това, 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, тъй като той управлява оперативните задачи в системата. Неговият достъп до всички модули му позволява да създава, променя и изтрива акаунти, както и да задава или променя правата за достъп на другите потребители. Това осигурява централизирано управление на потребителите и поддръжка на системата. 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 поръчки.

За управлението на доставки, доставчикът има възможност да създава и обновява записи за доставки. По този начин информацията за доставките е актуална и може да бъде проследена в реално време, което е важно за своевременното и точно изпълнение. В същото време, диспечери и бизнес клиенти имат достъп до функционалности като преглед и проследяване на доставки. На диаграмата са дефинирани 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 като 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, като 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 тези изисквания 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.

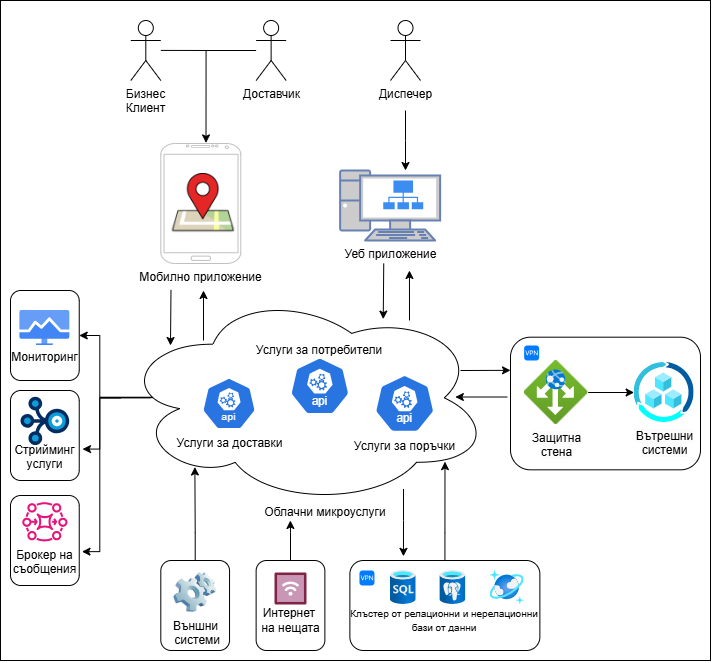
Изхождайки от анализа на литературата по управление на веригите за поръчки и доставки от първа глава, установихме, че основните изисквания към облачната система включват: регистриране и вписване на потребител, преглед на текущите поръчки, разглеждане на детайлите за определена поръчка и доставките към 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. Също така, трябва да позволява 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, осигуряваща непрекъсната и коректна работа на компонентите на системата, мащабируемост, която позволява 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 често се наричат „атрибути за качество“ на системата (Nacheva & Sulova, 2018).

Някои от нефункционалните изисквания към облачната система 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 / deployment);
* Системата следва да поддържа междуплатформен хостинг;
* Изисква се отговорът на системата да бъде върнат в рамките на секунди;
* Необходимо е да се спазват дефинираните SLA с ясни показатели и предвидени санкции при неизпълнението;
* 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. Илюстриран 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 функции като 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, тъй като текущото състояние на поръчка и местоположение на доставките се проследява 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 като 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, осигуряваща ефективност и надеждност на процесите.

Обхватът на уеб приложението включва балансиране на работното натоварване на превозните средства, позволява проследяване и коригиране както на поръчките, така и на доставките, осигурява предварително зададени решения. Диспечерите имат възможност да поправят грешни данни, като комуникират с клиентите или доставчиците. Същевременно всички промени се отразяват в базата от данни. Уеб приложението използва техники за оптимизация, които се изпълняват непрекъснато във фонов режим. Уеб приложението генерира ежедневни отчети като тези 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. Тези диаграми 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:

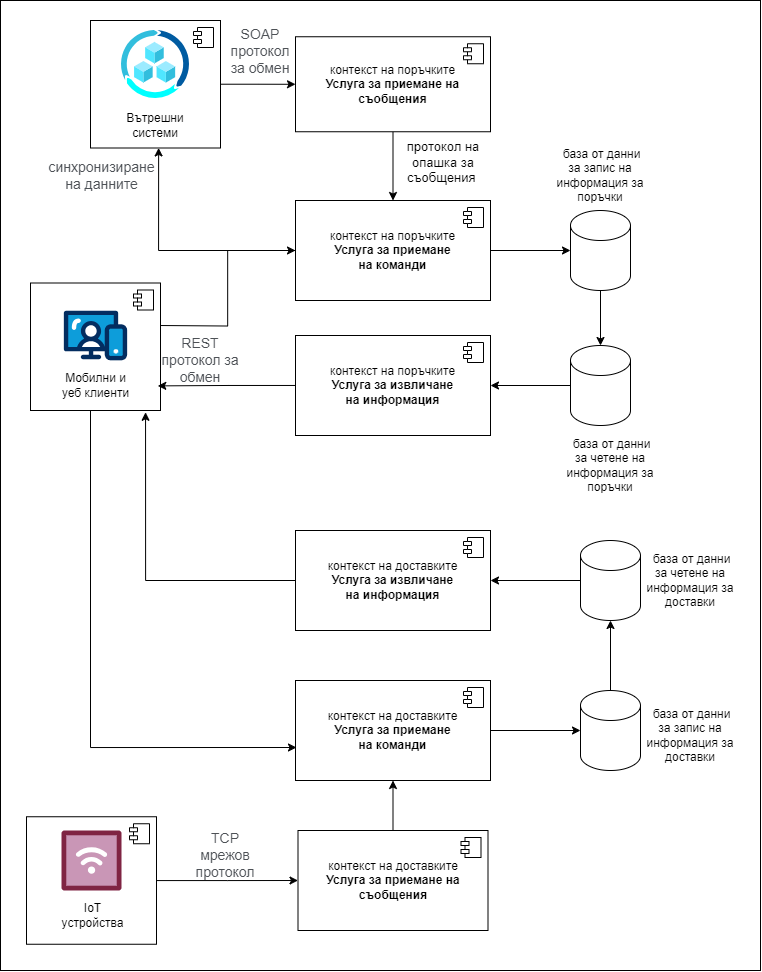
* Управление на поръчките: този модул включва набор от микроуслуги и NoSQL бази от данни за управлението на поръчки за продажба от бизнес клиенти. В него се обработва и съхранява информация за търговски поръчки чрез API, което интегрира вътрешна ERP подсистема.
* Управление на доставки: набор от микроуслуги 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 и 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 тези за доставки, са използвани „ограничени контексти“ (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]](#footnote-13) визуализира двата модула, техните контексти, микроуслуги и бази от данни, като 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. В тази връзка, фиг. 2.5 представя 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 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. Освен това 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, като по този начин своевременно се откриват и коригират евентуални несъответствия в състоянието.

Вътрешните системи осъществяват връзка с ПОСУП посредством услугите за приемане на съобщения, като 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, позволявайки интеграция на набор от сензори (Armiyanova, 2019).

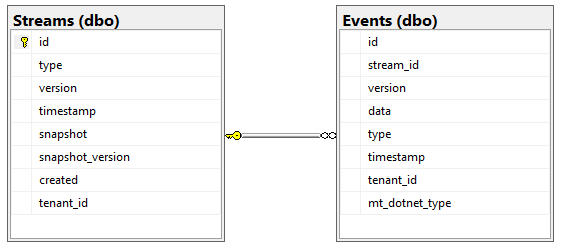
Хакерските атаки не бива да достигат до вътрешните системи, тъй като те поддържат основата на работните процеси. При най-лош сценарий, единствено облачната система следва да бъде засегната. За предотвратяване на подобни рискове, в ПОСУП се интегрира механизъм за контрол на входящия и изходящия трафик 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. По този начин, архитектурата се основава на принципите на CQRS, при които се разделят отговорностите за четене и запис, като се позволява паралелна обработка на големи обеми от данни, без прекъсвания или забавяния, причинени от блокиране на ресурси (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 могат да бъдат променяни, което осигурява точност и хронологичен ред. 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 осигурява сравнително лесно добавяне на нови типове потоци и събития, без необходимост от промени в структурата. Събитията се съхраняват хронологично, което позволява 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 като основа за 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, като създаване и актуализация. Тази организация позволява поддържането на история на всяка поръчка, като всяко събитие представлява конкретно действие или промяна в състоянието на поръчката.

В тази връзка, табл. 2.1 предоставя описание на 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)** | Tип 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 като са представени и допълнителни свойства и полета.

Таблица 2.2

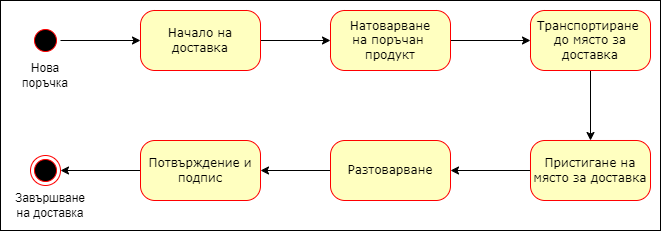
**Структура на таблица „Събития“**

|  |  |
| --- | --- |
| **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, отразяващ 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

В контекста на извършване на доставка, всеки етап от работния процес се отбелязва като отделно събитие и се съхранява в базата от данни. Това позволява 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. Тези диаграми предлагат визуално представяне на процеси и потоци от действия, което подпомага 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, като някои от етапите, като управление на рекламации и комуникация с клиента, са пропуснати. Теоретично, 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, като всяко от тях е свързано с един запис в таблицата за потоци. По този начин се проследява част от веригата на доставки, като се следи движението на продукцията в контекста на една клиентска поръчка. Подобна архитектура на данните позволява анализ на процесите в реално време, идентифициране на потенциални проблеми и оптимизация на различни аспекти на 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.

├───**Микроуслуга**

│ ├───**src** – индикира път към основния код

│ │ ├───**API**- логика за представяне

│ │ ├───**Business** – бизнес логиката на приложението

│ │ ├───**Core** – експонирани модели за клиентски приложения

│ │ ├───**Domain** – вътрешни DDD модели на приложението

│ │ └───**Persistence** – логика за достъпа до данните

│ └───**tests** – индикира път към тестовете на кода

│ └───**Tests** – пълен набор от автоматизирани (интеграционни) тестове

**Фиг. 2.8. Структурата на пакетите на всяка микроуслуга**

Разработка на автора

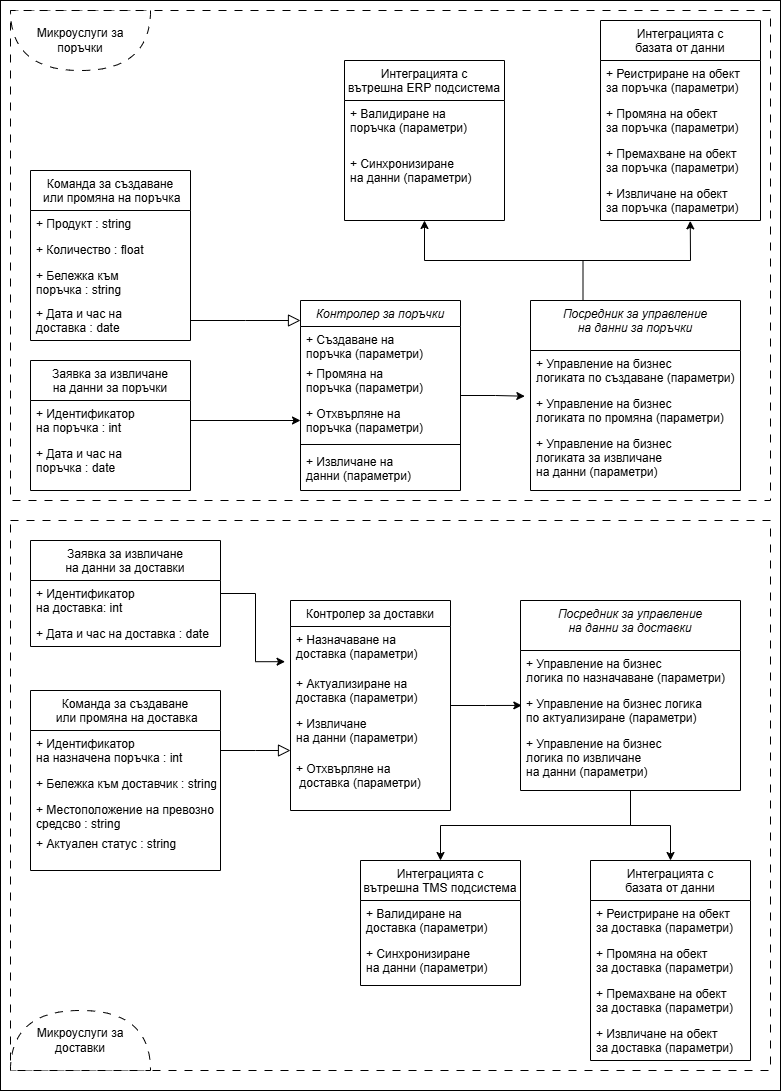
Нивото „src“ действа като централен посредник, координиращ взаимодействията между слоевете, представени от различни пакети и поддържащи принципите на DDD архитектура:

* API – служи като входна точка за комуникация, обработвайки SOAP, HTTP, TCP заявки и отговори;
* Основен (Core) – служи като 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. Също така поддържа 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) – служи като 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 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 идентификационни номера.



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

Разработка на автора

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. В този контекст, те взаимодействат с класовете 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. Успешното изпълнение изисква съвместни усилия от страна на екипите за разработка на софтуер и експерти в управлението на поръчки и доставки. Необходимо е всички заинтересовани страни да имат цялостно разбиране на изходния код, което да им позволява да предлагат или одобряват подобрения, както и да откриват потенциални проблеми. Лингвистичната рамка дава възможност за плавен преход от псевдокод[[14]](#footnote-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 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 реализира успешно или да доведе до неуспех. Mонада подпомага обработката на грешки, като например при проблем на микроуслуга с връзката към базата от данни. Този подход 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.

├─── Either<Резултат, Грешка>

│ ├─── променлива IsSuccessful

│ ├─── функция Match

│ └─── функция Flat Map

**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 прилага трансформацията 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 като 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. В тази връзка, фиг. 2.11. представя функция, изразена чрез псевдокод, която отговаря за създаване на нова поръчка в базата от данни.

├───**Заявка 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 е осигурявала собствена логика за управление на потребителските профили. Макар че този подход е често срещан и сравнително лесен за реализация, той има няколко съществени недостатъка:

1. Потребителското име и парола се предават кодирани в формат, който лесно може да се декодира;

2. Потребителското име и парола трябва да се предават с всяка заявка, което увеличава риска от прихващане на тези данни при незащитена връзка;

3. При базово удостоверяване липсва вградена поддръжка за управление на сесии;

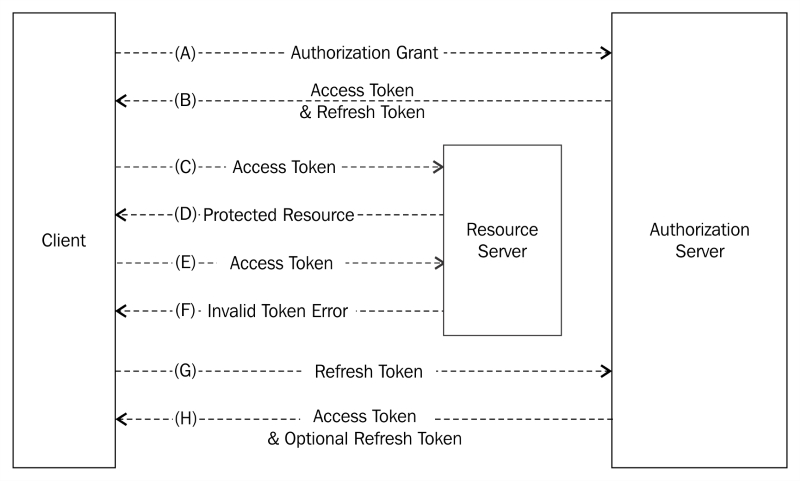
4. Управлението на потребителските имена и пароли, съхраняването и обновяването на идентификационни данни, както и осигуряването на тяхното криптиране и защита изискват допълнителна административни ресурси;

Тези 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, като 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, като предоставя единна рамка за прилагане на стандарти свързани с „Общия регламент относно защитата на данните“[[15]](#footnote-15) (Kesan et al. 2013).

В тази връзка, 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. Тези случаи изтъкват важността на мерките за сигурност и показват необходимостта от постоянно усъвършенстване на защитните механизми. Съществуват различни технологии за защита на данните, като криптографски алгоритми, биометрични методи за оторизация и машинно обучение за идентифициране и предотвратяване на хакерски атаки.

В този смисъл, модулът за управление на потребителски профили действа като първа линия на защита, създавайки рамка за удостоверяване и разрешаване на потребителски достъп, като 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 Secure Access Service Edge. 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 в контекста на управление на потребителски профили и контрол на достъпа са базирани върху OAuth 2.0 и OpenID Connect. OAuth 2.0 е протокол, основан върху стандарта RFC 6749[[16]](#footnote-16). Приложенията изискват и получават ограничен достъп до микроуслугите за поръчки и доставки като делегират удостоверяването на потребителя, без неговите идентификационни данни да бъдат компрометирани. Тази технология се използва широко в онлайн платформи като Facebook, GitHub и DigitalOcean. OpenID Connect допълва OAuth 2.0, като въвежда допълнителен слой за идентичност. Този слой добавя идентификационни токени, представени като JSON Web Tokens, които капсулират удостоверената потребителска информация. Това позволява по-сигурно и ефективно управление на потребителските профили и контрола на достъпа. На фиг. 2.12 е представена схема, илюстрираща взаимодействието между различните компоненти в контекста на OpenID и OAuth 2.0.



**Фиг. 2.12. Абстрактна диаграма на удостоверяване на потребител**

Източник: RFC 6749.

На базата на тази фигура, в облачната информационна система следва да се истанцира централизиран сървър за самоличност. Той имплементира OAuth и OpenID, прилагайки стандартизиран механизъм за управление на потребителския достъп, удостоверяване, издаване и разчитане на токени за проверка на самоличността. Тези стандарти осигуряват интеграция и сигурност при управлението на потребителския достъп. OAuth 2.0 предоставя различни видове разрешения за достъп, които са съобразени с разнообразните нужди на приложенията и специфичните нива на сигурност. Тези видове разрешения са представени в таблица 2.3.

Таблица 2.3

**Видове разрешения за достъп на OAuth 2.0**

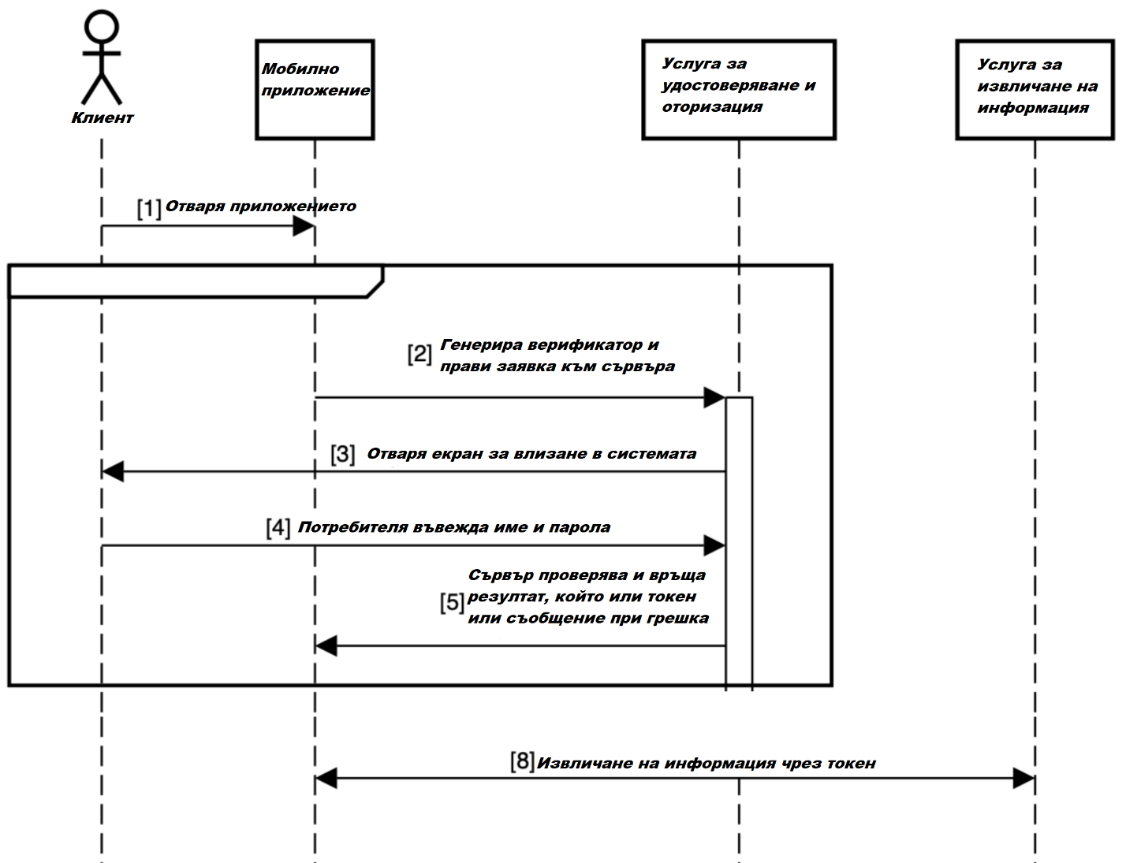
| **Вид** | **Случай на употреба** | **Описание** | **Съображение за сигурност** |
| --- | --- | --- | --- |
| **Идентификационни данни за парола** | Влизане с потребителско име и парола. | Позволява на клиентското приложение директно да поиска и използва идентификационните данни на потребителя, за да 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 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 aтаки за прихващане на код за оторизация, (man in the middle). |

Разработка на автора

*Код за оторизация* (*Authorization Code)* е често използван вид разрешение за достъп в OAuth 2.0, при което клиентско приложение пренасочва потребителя към страницата за вход в системата. Клиентското приложение разчита на т.нар. „клиентска тайна“ за изготвяне на HTTP POST заявка до крайната точка за токен за достъп.

При доказателствения ключ за обмен на данни (Proof Key for Code Exchange - PKCE), описан в RFC 7636, този процес се подобрява, като клиентската тайна се заменя с динамично генериран код, наречен „верификатор“. Верификаторът представлява произволен низ от символи, който се хешира чрез определен алгоритъм (например SHA-256).

Считаме, че PKCE е подходящ вид разрешение за достъп, който използва верификатор, за да достигне крайната точка за получаване на токен, осигурявайки сигурна потребителска сесия. В тази връзка, процедурните стъпки за осъществяване на удостоверяването в облачната информационна система са представени на фиг. 2.13.



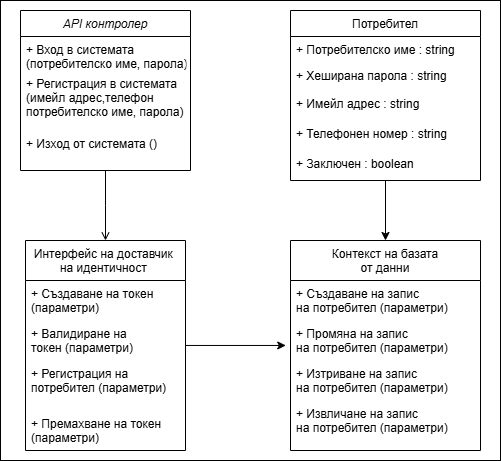
**Фиг. 2.13. Диаграма на удостоверяване чрез код и ключ за обмен**

Разработка на автора

Процесът на удостоверяване и извличане на информация е представен чрез диаграма на последователността. Диаграмата включва бизнес клиент, мобилно приложение, услугата за удостоверяване и оторизация, както и услуга за извличане на информация. Услугата за удостоверяване и оторизация представлява централизирания сървър за самоличност, отговорен за проверка на идентификационните данни на потребителя.

Мобилното приложение инициира процеса, като 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.

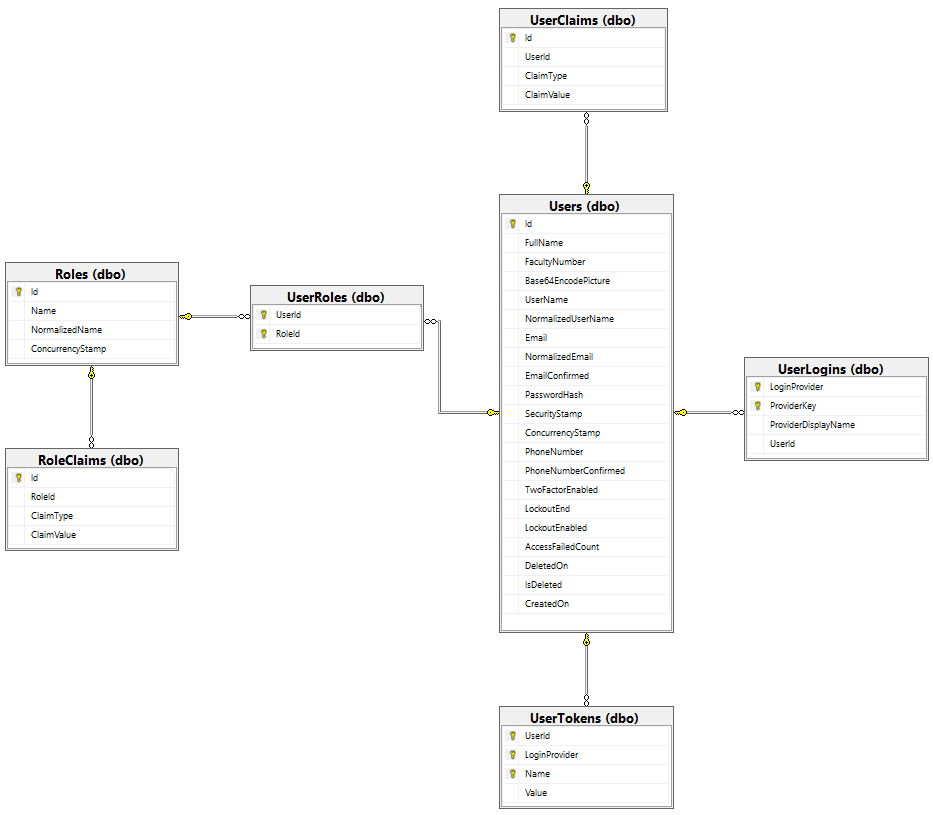


**Фиг. 2.14. Диаграма на класовете, отговарящи за удостоверяване**

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“, позволяващ 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 предоставят операции за създаване, актуализиране, извличане и изтриване на потребителски акаунти, както и функционалности за регистрация, вход и излизане от системата. Класовете, които имплементират интерфейса, взаимодействат пряко с класовете за потребител и с контекста на базата данни. API контролерът, отговарящ за обработката на HTTP заявки, свързани с потребителските акаунти, комуникира с тези класове, като осигурява връзка между мрежовия слой и бизнес логиката. API контролерът предоставя крайни точки, които мобилни и уеб приложения достъпват, за да задействат логиката в интерфейса.

Фиг. 2.15 представя релационен модел на базата от данни.



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

Разработка на автора

Базата от данни, която обслужва API на доставчика за самоличности, 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 на имейл и телефонен номер, двуфакторна идентификация и управление на заключване при неуспешен достъп;
  + Roles: таблица дефинираща ролите, които могат да бъдат зададени на потребители, като администратор, доставчик, диспечър или бизнес клиент;
  + UserRoles: служи като свързваща таблица между Users и Roles, което позволява на един потребител да има множество роли. Най-често това е комбинация между администратор и диспечер;
  + UserClaims и RoleClaims: Тези таблици обработват „твърдения“ (claims) за потребители и роли, като 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 платформи като Google и Facebook, 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). Той също така включва оценка на предимствата и недостатъците на всеки протокол в контекста на клиентски и уеб приложения, които се свързват 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 позволява на тези приложения да изпращат съобщения и да продължават с други задачи, като отговорите се обработват във фонов режим.

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. Освен това, 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).

UML диаграмата на последователностите показва как продуктите и услугите в ПОСУП си взаимодействат, за да изпълнят основната функционалност на системата. В тази връзка, 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 представи интерактивното сътрудничество между различните компоненти на системата. Тези диаграми са от съществено значение за подпомагането както на техническите, така и на нетехническите заинтересовани страни, като ги интегрират в процеса на изграждане на UML. Като част от поведенческите диаграми на UML, те се фокусират върху динамичното поведение на системата.



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

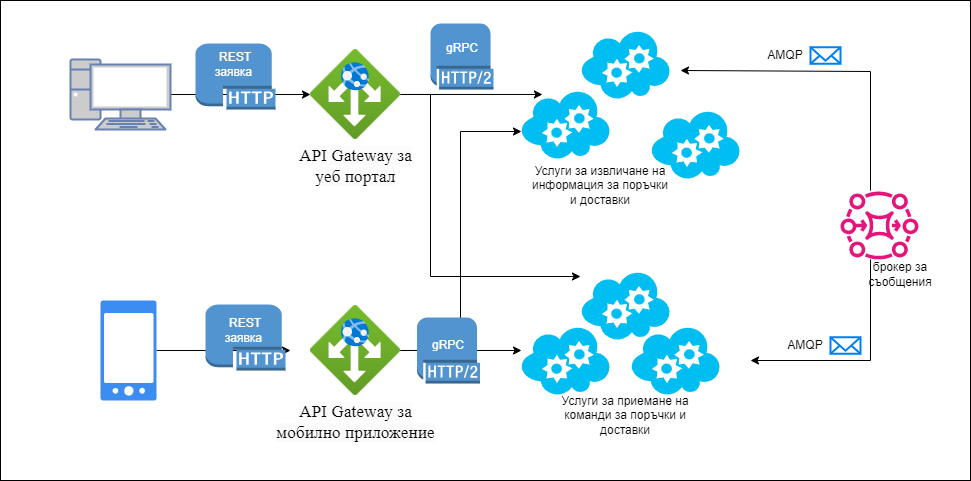
Разработка на автора

Диаграмата на последователността показва как протича комуникацията между бизнес клиент и мобилно приложение, включително услугите, които обработват заявките. 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 същестуваща 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 централизирана 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, като 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, като по този начин намалява времето за отговор и облекчава натоварването на микроуслугите.
* 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, като предоставя информация за производителността и възможните проблеми в системата.

Интегрирането на API Gateway в облачната инфраструктура подобрява сигурността, като същевременно осигурява поддръжката на връзките към микроуслугите (Xu, Jin & Kim, 2019). В тази връзка, фиг. 2.17 представя API Gateway в контекста на облачна информационна система за управление на поръчки.



**Фиг. 2.17. API Gateway в контекста на управление на поръчки**

Разработка на автора

За да предостави обобщена информация за поръчки и съответните доставки, когато мобилното или уеб приложение изпрати HTTP заявка към API Gateway, той я разделя на отделни части и 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, като 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[[17]](#footnote-17). Тези крайни точки използват Representational State Transfer (REST) архитектурен стил, за да осигурят стандартизирани интерфейси за комуникация. HTTP методите позволяват 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 предоставя 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 доставка по идентификатор |  | Актуализира данните за доставка по идентификатор, ако съществува | Премахва поръчка по 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 същесвуващ 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 беше разгледано в предходна т.3, функционалностите както в мобилното, така и в уеб приложението започват с потребителска автентикация и оторизация. Началната стъпка за достъп до системата включва въвеждане на потребителско име и парола. Първият екран и в двете приложения изисква идентификация, като предоставя и допълнителни механизми чрез външни доставчици, както и двуфакторна автентикация. Тези мерки са от основно значение за запазване на конфиденциалността и интегритета на потребителските данни.

В тази връзка, 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 определни 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. Освен това, са представени 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.

Екранът позволява всеки елемент от списъка да бъде натиснат за допълнителни действия, като преглед на детайли или редактиране на поръчката. Ако бизнес клиентите се съгласят на използване на бисквитки, всяко тяхно действие, като натискане на бутон, въвеждане на данни или отваряне на екран, се проследява и изпраща към облачните микроуслуги. Това позволява събирането на данни за поведението на потребителите, които 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, като записва 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 служат 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. В контекста на правната регулация, електронните доказателства за доставка трябва да съответстват на местните и международни стандарти и нормативни актове, като регламента на Европейския съюз за електронната идентификация и доверителните услуги (eIDAS). Това гарантира 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 позволяват на диспечерите да управляват оперативните процеси във веригата на доставки. Тези функционалности 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: Позволява създаване на нов запис в графика чрез въвеждане на необходимата информация за дата, час, дестинация и количество.
* Промяна: Тази функция предоставя възможност за редактиране на съществуващи поръчки или доставки, като например промяна на часа или пренасочване към друг доставчик.
* Уведомление: Този бутон 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 позволява планиране на множество задачи едновременно. Освен това интерфейсът включва опции за търсене на конкретна поръчка или филтриране по статус, доставчик или 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. Освен това, диспечерите имат възможност ръчно да променят статуса на дадена доставка, ако сигналът от сензорите на превозвачите бъде загубен. Също така, уеб порталът предоставя възможности за генериране на различни видове отчети и анализи, подпомагайки вземането на информирани решения.

Уеб порталът предоставя функционалност за изчисляване на времето, необходимо за достигане на дадено превозно средство от точка А до точка Б. Това позволява приложението да функционира 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 маршрутизиране**

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. Освен това, GPS системата осигурява прозрачност и контрол върху транспортните процеси.

Дизайнът на приложенията е създаден с цел поддържане на добавянето на нови функционалности и промяната на съществуващите, в съответствие с нуждите на компанията. Това осигурява разширяемост и адаптивност на системата, като гарантира гъвкавост и устойчивост на бъдещи промени.

**Изводи и обобщения към първа глава**

В настоящата глава предложихме концептуален, логически и комуникационен модел на облачната информационна система. Основната цел на тези модели е да представят взаимодействието между диспечери, доставчици и бизнес клиенти на производсвено предприятие. Процесите, които представят тези модели, са свързани със създаване, промяна или отхвърляне на поръчки за продажби, координацията по време на доставка и комуникацията между отделните страни. В този смисъл, 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, идентифициране на основните бизнес сценарии и преглед от високо ниво на концепцията. Логическият модел, от друга страна, има за цел да представи структура за взаимодействие между модули и компоненти. Той служи като план за проектиране на софтуера в съответствие с теоретичната основа, разгледана в първа глава. В логическия модел са описани обекти, свойства, взаимовръзки и процеси в системата. Това е постигнато чрез създаване на диаграми от тип обект-връзка, диаграми на последователности и 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 тези модули, всеки от тях бе подложен на отделен анализ. Също така е представен комуникационен модел между основните модули и протоколите за връзка между тях: HTTP/HTTPS, SOAP, REST и gRPC.

В контекста на клиентски и уеб приложения са оценени предимствата и недостатъците на различните методи за комуникация. 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.

В допълнение, е използвана технологията API Gateway като централизирана точка за управление на маршрутизацията, натоварването, кеширането и мониторинга на микроуслугите. Също така е представен и централизиран доставчик на идентичност, който служи за удостоверяване и оторизация.

Основни екрани и крайни точки на мобилните и уеб приложения са описани, включително потребителските интерфейси и API. Клеинтски и сървърни приложения изпозлват стандартни HTTP методи и различни формати за пренос на данни.

В настоящата глава се обосновава значимостта на проектирането на архитектура за облачна информационна система, която да се адаптира към динамично променящите се бизнес изисквания. В трета глава се изследва практическата приложимост на тази архитектура в конкретно производствено предприятие.

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

В трета глава се анализират практическите аспекти, свързани с внедряването на облачната информационна система за управление на поръчки от клиенти в производственото предприятие „Хейделберг Цимент Девня“ АД. Разгледани са основни характеристики на дейността на компанията и са избрани подходящи технологични средства за физическа реализация на системата. Това дава възможност системата да бъде апробирана в реална работна среда. Освен това, се оценяват възможностите и функционалностите, методите за мониторинг, както и прогнозните разходи за внедряване на облачната информационна система.

## Обща характеристика на дейността на компания „Хейделберг Цимент Девня“ АД

„Хейделберг Цимент Девня“ АД е най-големият производител на цимент в България, разположен в град Девня, област Варна. Освен цимент, предприятието предлага различни бетонови смеси и специализирани строителни материали. „Хейделберг Цимент Девня“ е част от немската мултинационална компания Heidelberg Materials. Според бизнес доклад на Капитал (Георгиева, 2025) „Хейделберг Цимент Девня“ е определена като най-рентабилната компания за строителни материали в класацията за 2023 г. В доклада се посочват проектите, върху които Heidelberg Materials се фокусира – както на глобално, така и на местно ниво. Въвеждането на облачни информационни системи е сред технологичните иновации, които компанията планира да интегрира в своите търговски организации в над 50 държави по света.

Основната дейност на „Хейделберг Цимент Девня“ АД включва производство и дистрибуция на цимент, инертни материали, готови бетонови смеси и асфалт. Дружеството произвежда и доставя бетонова смес, приготвена в централизирани съоръжения за дозиране. Тези смеси 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 да изпращат данни за нивото на водата, температура и основни характеристики на сместа в реално време, би гарантирало високо качество на продукта, тъй като неговите свойства могат да се променят по време на транспортиране. Счита се, че централизираното смесване е по-екологично от смесването на строителната площадка, тъй като генерира по-малко отпадъци и позволява 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 аспекти 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 гарантира надеждност на доставките.

Въз основа на проучванията, представени в предходните две глави, внедряването на ПОСУП в компания като „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. Адаптирайки се към специфичните нужди на компанията, ПОСУП цели да рационализира управлението на поръчки и доставки. Това е от първостепенно значение за предприятие с висока степен на сложност на логистичните и производствените процеси.

В настоящия текст 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 коригиране на плана за доставки и налага 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 като интензивен трафик, задръствания или внезапни промени, поискани от клиента. Подобни обстоятелства изискват моментална 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 комплексността 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, като 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. В рамките на настоящото изследване, посредством ПОСУП, предлагаме следния набор от подобрения:

* **Приемане на поръчка** - към момента поръчките се получават по имейл или телефон 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 като получат одобрение, потребителите могат да регистрират нови поръчки, както и да променят или отхвърлят съществуващи.
* **График -** след като поръчката 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 тези 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, като 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). В тази връзка, е 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.

От друга страна, публичните PaaS услуги предлагат планове за плащане според употребата, което означава, че предприятието заплаща само за реално използваните ресурси. Това позволява на предприятието да увеличава или намалява употребата на изчислителни ресурси в отговор на променящите се бизнес изисквания. Поради тази причина, считаме, че публичните облачни услуги са подходящи за нуждите на компанията. Както вече бе споменато, тези услуги предоставят възможност за адаптиране към нови пазарни условия и минимизират финансовия риск, свързан с придобиването на скъпоструваща компютърна техника.

Процесът на внедряване на информационната система може да бъде разделен на няколко етапа. На първо място се анализират текущите системи, определят се изискванията за облачна интеграция и оценка на готовността за промяна на съществуващата инфраструктура от подсистеми. След това се избират подходящи облачни IaaS, PaaS и SaaS услуги, които да отговарят на специфичните нужди на предприятието. При този процес 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. В този процес се идентифицират дефекти и възможни проблеми, което води до подобряване на качеството на програмния код и гарантира надеждността на крайния продукт. На този етап се изпълнява 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 промяната от страна на персонала може да затрудни внедряването на облачната система, особено ако не е осигурено адекватно обучение и подкрепа при използването на софтуера. Поради тази причина е важно редовно да се провежда обучение на персонала, за да могат служителите да се запознават с новите функционалности и подобрения. (Тодоранова, 2024). Наред с това, е необходимо да се опише документация, обхващаща всички аспекти на функционирането и поддръжката на ПОСУП.

Концептуалният и логическият модел на системата, предложени във втората глава, се базират на изследване на организацията на работния процес в „Хейделберг Цимент Девня“ АД. Облачната система има възможност за интеграция със съществуващите ERP, CRM и TMS подсистеми. В тази връзка е от значение да се направи прогноза за необходимата производителност на системата, базирана на параметри като броя на потребителите, броя на извършваните от тях действия и броя на HTTP заявки, генерирани от клиентските приложения към облачните микроуслуги. Изборът на технологични средства и доставчик на публични облачни услуги трябва да съответства на тази прогноза и да отговаря на местните и международни стандарти за сигурност.

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

Освен това, 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.

В този контекст, софтуерната компания „*TechEmpower*“ проведе сравнителен анализ на няколко уеб-базирани работни рамки. Те дефинират работната рамка като комбинация от сървърна технология и програмен език. В анализа са разгледани различни сървърни технологии, програмни езици и производителността им, измерена чрез броя на обработените HTTP заявки за секунда.

Резултатите от сравнителния анализ са представени в таблица 3.1.

Таблица 3.1

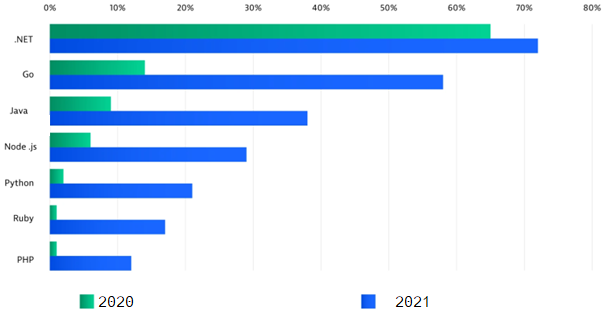
**Сравнение на работни рамки за разработка**

| **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: Тechempower 29.09.2023.

Данните на Techempower показват, че ASP.NET демонстрира по-висока производителност в сравнение с други алтернативни платформи за уеб приложения, като се отбелязва, че ASP.NET Core е поне два пъти по-бърз от Node.js, който е втори в класацията. В допълнение, всяка година от Microsoft публикуват план за предстоящ напредък и поддръжка на .NET, осигурявайки периодични подобрения.

Освен това GitHub предоставя информация за над 5,7 милиона активни разработчици, работещи по проекти с отворен код, които използват езика за програмиране C#, част от екосистемата на .NET и Microsoft (Gouigoux, 2024). От друга страна, в доклад на Stack Overflow се отбелязва, че .NET Core е водещата работна рамка за годините 2020 и 2021, както е показано на фигура 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 предоставя 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 като Windows, Linux и macOS, което осигурява бързо стартиране и изпълнение на приложенията.

Изборът на ASP.NET Core позволява интеграцията 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 като 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 като TensorFlow и ONNX (Mahasivabhattu & Bandi, 2024). С помощта на ML.NET се осигуряват и API-та за интеграция с други продукти на Microsoft, като например Outlook, което може да спомогне за реализирането на омниканалния подход, разгледан в т.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, като 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, предлага 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 гарантира оперативна съвместимост в рамките на екосистемата, поддържана от Microsoft. Това е от основно значение за успешната реализация на съвременни софтуерни проекти. В тази връзка, фигура 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 остават 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 като 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. Освен това, използването на общ софтуер би уеднаквило вътрешните процеси, което би улеснило сътрудничеството и комуникацията между различните дъщерни фирми (Vasilev & Kehayova-Stoycheva, 2019).

В тази връзка, Azure разполага с 64 центъра за данни по целия свят (фиг. 3.4). Те са разположени в различни региони, за да предоставят услуги с максимална производителност и надеждност. Тези центрове за данни са свързани чрез високоскоростни оптични връзки, което осигурява 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 Azure**

Източник: De La Torre, 2024.

Въз основа на проведените анализи, може да се заключи, че изборът на .NET и Azure за изграждане на облачно базирана система за управление на поръчките представлява рационално и ефективно решение. Тази комбинация предлага множество предимства при разработването и внедряването на иновативни приложения. .NET осигурява висока производителност и модулност, което улеснява бъдещото развитие и поддръжка на системата. В съчетание с Azure, водещ доставчик на облачни услуги, се създава надеждна среда за разработка и изпълнение.

В контекста на този избор, е важно да се отбележи, че различните микроуслуги, които информационната система използва, имат различни изисквания за съхранение на данни. Azure предлага няколко различни вида хранилища за данни. В таблица 3.2 са представени съответствията между различните облачни услуги и характеристиките на съхранените данни. Тези хранилища за данни включват релационни и NoSQL бази от данни, файлови системи и оптимизирани за „*големи данни*“.

Таблица 3.2

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

Scatter chart

Description automatically generated with medium confidence

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:

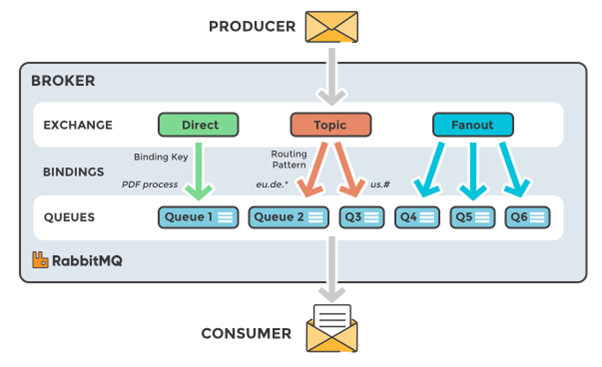
* Azure SQL Database - облачна услуга за релационни бази от данни, която предлага вградени функции за висока производителност, автоматизирани актуализации и архивиране, както и защита на данните чрез криптиране. Тази услуга поддържа пълна съвместимост с Microsoft SQL Server, което улеснява процеса на локална разработка, като позволява на програмистите да използват SQL Server на личните си компютри за тестване на приложенията. Освен това, Azure SQL предоставя интелигентни функции като автоматично настройване на обема използвана памет, репликация на данни в различни географски региони, маскиране на данни за определени потребители, както и одит на всички действия, 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 осигурява високо ниво на надеждност чрез автоматично репликиране на данни между различни региони. Освен това, Cosmos DB предлага 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, като текстови или двоични данни. Тази услуга е предназначена за различни сценарии, включително архивиране и възстановяване на данни от други бази, като например Azure SQL Database и Cosmos DB, както и за файлове като документи, снимки или видео клипове. Azure Blob Storage предлага три нива на съхранение, които позволяват оптимизиране на разходите спрямо нуждите от достъп до данните. 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 позволява 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 осигурява висока степен на интеграция с разнообразни аналитични услуги и инструменти, включително и Azure Synapse Analytics.

За нуждите на микроуслугите в ПОСУП може 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 от втора глава, могат да използват Azure Cosmos DB. Това осигурява съвместимост между четирите бази от данни за четене и запис на обекти за поръчки и доставки. Освен това, тази комбинация позволява създаването на т.нар. „*материализирани изгледи*“, които могат да комбинират данни за поръчки и доставки в една заявка от две бази за четене.

От друга страна, 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 осигуряват 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 осигурява 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 позволява асинхронна комуникация между различни микроуслуги чрез прилагане на протокола AMQP. RabbitMQ е софтуерно приложение, което изпълнява ролята на брокер на съобщения. Това е проект с отворен код, който е съвместим с различни операционни системи и сървърни технологии. Приложенията могат да установяват асинхронна комуникация чрез обмен на съобщения през т.нар. „опашки“. Фигура 3.5 представя вътрешните компоненти, които поддържат работата на брокера, включително опашкa, 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. Освен това, съществуват и множество други технологии, които работят на различни платформи като iOS, Android, HarmonyOS и Tizen. В този контекст, таблица 3.3 предлага сравнителен обзор на различните технологии.

Таблица 3.3

**Сравнителен анализ на 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 телефона** | 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, като X-Code и Android Studio. Тези приложения имат пълен достъп до функциите на устройството, като камерата, GPS и различните сензори. ***Хибридните приложения***, от друга страна, използват т.нар. уеб изглед (web view), в който се вгражда уеб съдържание, разработено чрез HTML, JavaScript/TypeScript и CSS, с помощта на платформи като Cordova и Onsen. Целта е тези приложения 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*** (Progressive Web Apps), подобно на хибридните приложения, използват работни рамки като Ionic с Angular или Vue и Blazor със C# (Sulov, 2024). Тези приложения функционират като уеб сайтове, но включват допълнителни функционалности, които им позволяват да работят офлайн и да се инсталират на мобилни или десктоп устройства. Достъпът им до функциите на телефона или компютъра е ограничен, но има потенциал за усъвършенстване в бъдеще. Считаме, прогресивните уеб приложения като подходящ избор за реализацията на уеб портала.

***Междуплатформените приложения*** (cross-platform), разработени с технологии като .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 като Android, iOS, HarmonyOS и Tizen. Тези технологии предлагат възможности 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, тези ограничения 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 като технология за уеб портала;
* .NET MAUI за разработка на мобилното приложение.

Интеграцията на различни технологии осигурява надеждност и мащабируемост в ПОСУП, като същевременно опростява процесите по разработка и поддръжка, благодарение на координацията в екосистема на Microsoft. Това води до съвместимост и взаимовръзка между различните компоненти на системата (back-end и front-end), както и предоставя висока производителност и сигурност за потребителите.

## 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 предлагат възможност за увеличаване или намаляване на компютърни 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 позволява автоматизирано компилиране на програмния код, тестване и създаване на контейнер, който да се внедри в хостинг услугата.

В тази връзка, таблица 3.4 предоставя основните хостинг услуги на Azure и съответстващите им случаи на употреба.

Таблица 3.4

**Услуги на Azure за хостинг и съответстващи случаи на употреба**

Chart, scatter chart

Description automatically generated

Източник: Moniz et al., 2021.

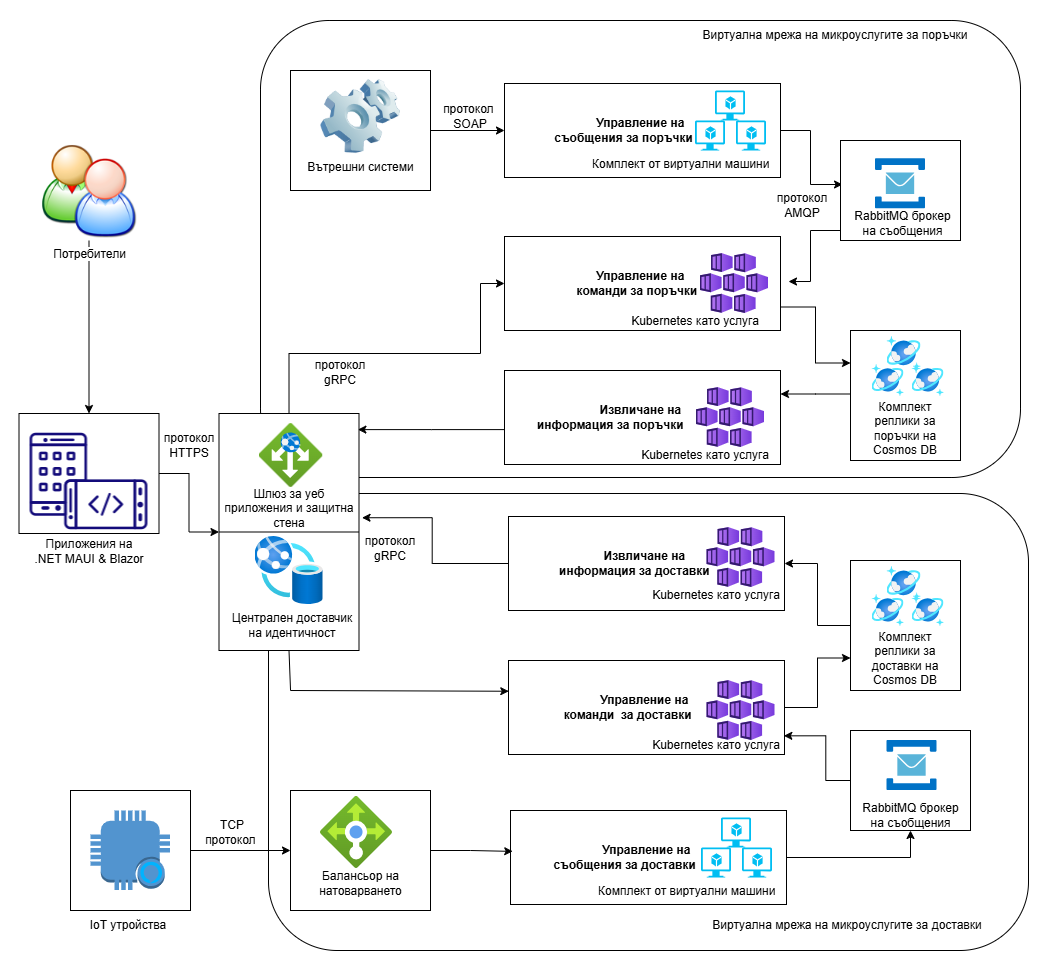
***Azure*** ***App Services*** е един от начините за хостване на приложения. Той е подходящ за използване при монолитна архитектура. Хостингът е надежден и работи в 99,95% от времето (SLA). Тази PaaS услуга предлага функции като автоматично мащабиране, актуализиране на нови версии, без прекъсване на работещи услуги и удостоверяване чрез централен доставчик на идентичност (т.2.3). Също така позволява отстраняване на грешки в приложението, докато работи в производствена среда с инструмент, наречен Snapshot Debugger. По подразбиране, приложението е достъпно в Интернет, без необходимост от настройка на домейн или конфигуриране на DNS.

Друг избор, с подобни на App Services функции, но от тип IaaS, е услугата ***Azure Virtual Machines***. Тя поддържа набор от протоколи за обмен на информация, включително TCP и SOAP, и позволява преместване на съществуващи приложения от други виртуални машини. Чрез услугата се предлагат  ISO файлове с изображения, като например Windows Server, който работи с IIS и има инсталиран и предварително конфигуриран ASP.NET, както и собствени софтуерни лицензи (като тези 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. Тези pods 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. Тези компоненти работят съвместно, за да предоставят надеждност, устойчивост и висока производителност. Също така, по този начин се оптимизират разходите, като се следят внезапни пикове в трафика или 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 тези данни, считаме, че AKS е подходяща услуга за хостинг на микроуслугите, която също така отговаря на нефункционалните изисквания за информационната система за управление на поръчки, представени във втората глава на дисертацията. 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.

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[[18]](#footnote-18).

Тези платформи предоставят уеб-базиран интерфейс, който 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 като 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 таб. 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 |
| **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 като GitHub Actions, Jenkins, CircleCI и Travis CI. GitHub предоставя 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.

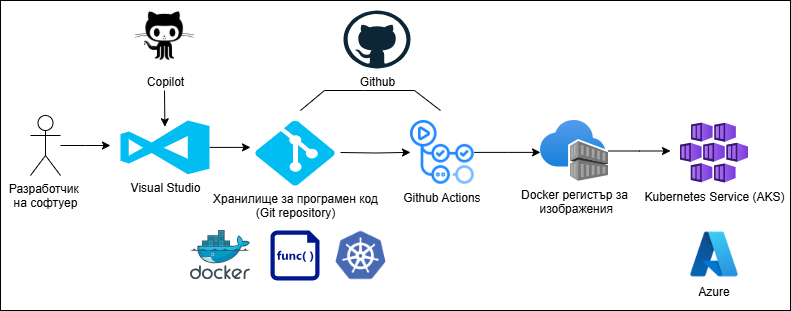
Интуитивният потребителски интерфейс улеснява сътрудничеството между разработчиците чрез функционалности като известия за промени в кода (pull requests) и управление на проблеми (issues). GitHub предоставя и среда за разработка (codespaces), която предоставя предварително конфигурирани виртуални машини 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 като Visual Studio и Codespaces, Copilot осигурява възможност за автоматично генериране на цели класове и функции, въз основа на контекста и 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, може да бъде достъпено на следния линк: <https://github.com/profjordanov/cloud-based-management-information-system>.

В контекста на физическата реализация и поддържане на програмния код, е важно да отбележим значението на виртуализацията при експлоатацията на информационната система, и в частност 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 позволява 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, от друга страна, е метод, който улеснява тяхното внедряване и управление. Всяка микроуслуга е „капсулирана“ в свой собствен контейнер, който бива интегриран в AKS.

Виртуализацията и контейнеризацията подпомагат практиките по „развитие и операции” (DevOps). Тези практики автоматизират разработката и внедряването на облачни услуги чрез технологии като Docker, GitHub Actions и AKS. Основната цел на DevOps е да изгради интегрирана среда, в която рутинните дейности по изграждане, тестване и актуализиране на информационната система се осъществяват по план. DevOps практиките позволява непрекъсната доставка и внедряване, като същевременно осигуряват по-високо качество на софтуера и намаляват времето за реализация на нови функционалности. 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.



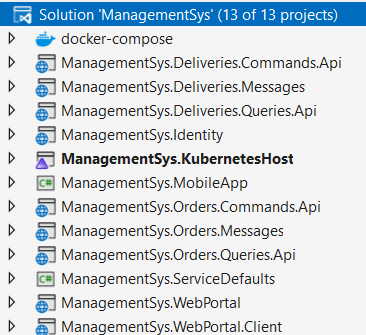
**Фиг. 3.7. Работен поток, съчетаващ процесите на разработване и актуализиране на облачни услуги**

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 компилационни грешки (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, като 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. Тези стратегии са предназначени да оценят дали системата функционира според очакванията на потребителите. По този начин те помагат 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 „А“ (контролна) и версия „B“ (тестова), като се измерват предварително дефинирани параметри. Това позволява вземането 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 позволяват 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 А** - 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 симулирана среда, която имитира реалните условия на работа на системата. Тестовите процедури обхващат различни сценарии, за да осигурят контрол върху системата в различни ситуации. В този контекст, тестовете от група А се фокусират върху управлението на информацията за поръчки, докато тези от група B са насочени към информацията за доставки.

След провеждане на необходимите тестове и техния анализ, резултатите са представени в таб. 3.7 и таб. 3.8. В тях са представени:

* ID на поръчка - aвтоматично 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 тестването за 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, таб. 3.8 илюстрира промените в някои от поръчките и доставките. Тези промени също са отразени във вътрешните подсистеми.

Таблица 3.8

**Резултати от A/B тестването за 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 |

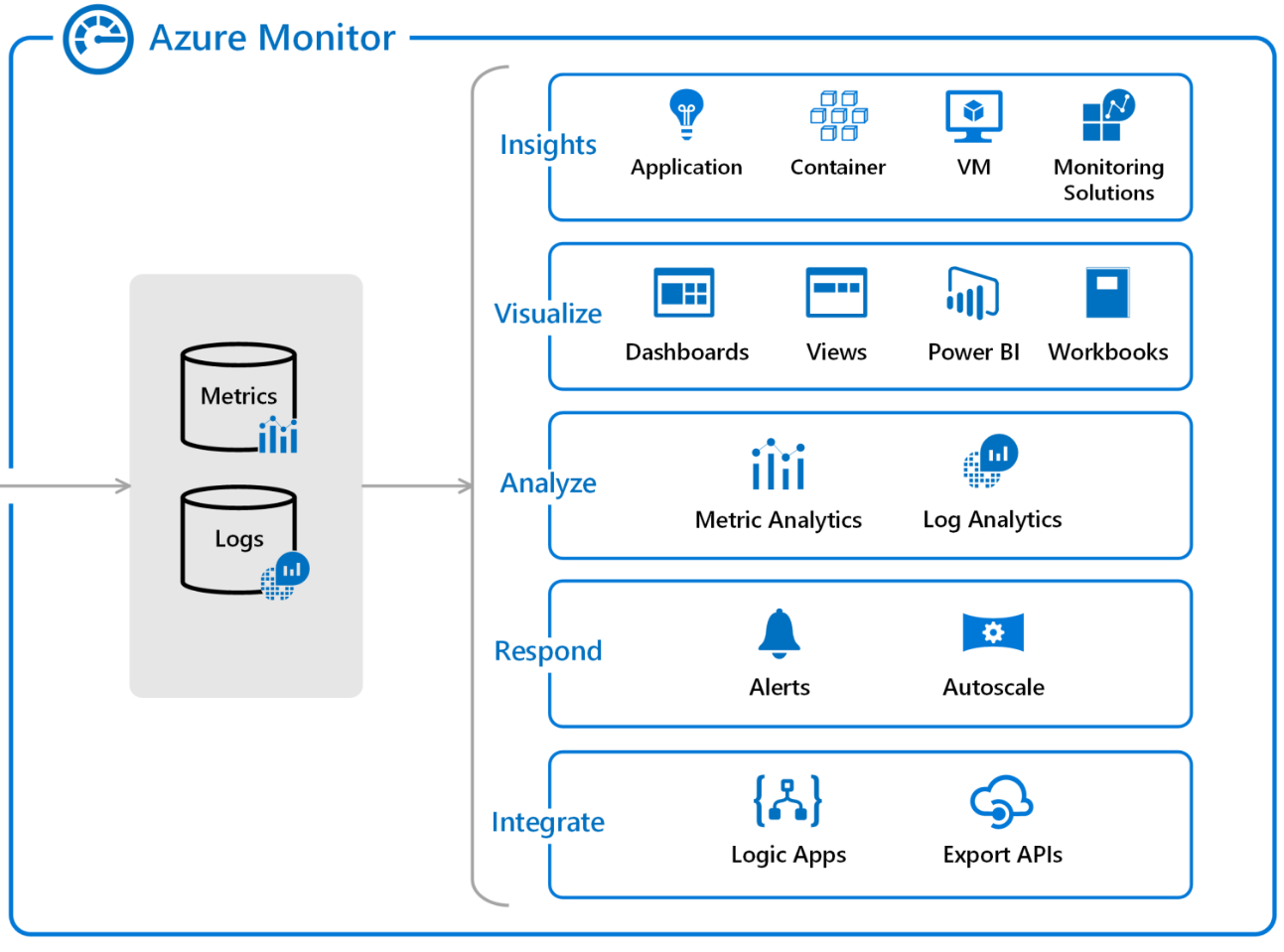
### 3.4.2. Системен мониторинг

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 позволяват проследяване на системните ресурси, времето за отговор на микроуслугите и натоварването;
* Създаване на индикатори за производителност - определяне на показатели като броя на приетите поръчки чрез системата. Това осигурява на мениджмънта на компанията представа за капацитета на системата и подпомага нейното 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 е интегриран в облачната платформа Azure, предлагайки услуги за наблюдение, поддържане на системен дневник и следене на индикаторите за производителност. Тези услуги могат да бъдат използвани от разработчици, бизнес специалисти и мениджъри. Мониторингът може да се раздели на две категории: мониторинг на инфраструктурата и мониторинг на приложенията. Мониторингът на инфраструктурата включва оценка и контрол на системни ресурси като процесор, памет, дисково пространство и мрежов трафик. От друга страна, мониторингът на приложенията се фокусира върху наблюдение на функционалността и ефикасността на отделните услуги в системата, разглеждайки параметри като време за реакция, честота на грешки и проследяване на транзакции между микроуслугите и ERP.

Azure Monitor предоставя услуга за диагностика в реално време, като обобщен модел на този процес е представен на фиг. 3.9.



**Фиг. 3.9. Обобщен модел на Azure Monitor**

Източник: Vettor 2023.

Един от основните аспекти на мониторинга е 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.

Значението на постоянното наблюдение е свързано с очакванията на бизнес клиентите за постоянната достъпност на системата – 24 часа в денонощието/7 дни в седмицата. Както бе споменато в глава първа, мониторингът също така определя и изпълнението на SLA.

Освен това, възможни са случайни колебания, предизвикани от непредвидени събития като 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 диспечерите в реално време би осигурило навременна реакция, способна да предотврати допълнителни затруднения.

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). Това е 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 предоставя възможност за сравнение на различни ценови планове, включително Free, Shared, Basic, Standard, Premium и Isolated. Всеки от тези планове предлага различни характеристики по отношение на процесорна мощност, памет, съхранение и допълнителни функции като автоматично мащабиране и интеграция с виртуални мрежи. Изборът между различните планове позволява 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 остават валидни към началото на 2025 г.*

Разработка на автора

В прогнозата за месечните разходи са използвани Standard и Premium ценови планове, които позволяват мащабиране на системата. Тези ценови планове са основни за компании с голям обем от данни, като „Хайделберг Цимент Девня“.

**Изводи и обобщения към първа 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. „Хайделберг Цимент Девня“ АД 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 позволява централизиране и управление на данните. Функционирането на ПОСУП се осигурява 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.

В заключение, 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 също така целят автоматизация на процесите във 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. Предлага се рационализиране на процесите чрез внедряване на персонализирана информационна система, конфигурирана към конкретно предприятие.

В първата глава на изследването 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 и микроуслуги като IaaS, PaaS и SaaS. Освен това са разгледани 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 по управление на поръчки, втора глава представя концептуален, логически и комуникационен модел, както и 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 елемент от втора изследователска задача и представлява основен принос на настоящото изследване.

Въз основа на създадените модели, в третата глава на изследването се разглежда общата характеристика на дейността на компанията „Хейделберг Цимент Девня“ АД. След това се извършва подбор на технологични средства за физическа реализация на системата. Практическата приложимост на изследването се представя чрез прилагане на А/В тестване. ПОСУП подпомага изпълнението на поръчките, подобрява управлението на ресурсите и предоставя възможности за автоматизация и мащабируемост.

С оглед на нарастващата сложност на производствените процеси и необходимостта от надеждни прогнози за търсенето, считаме, че внедряването на изкуствен интелект би допринесло за повишаване на гъвкавостта на предприятието спрямо динамичните 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. В тази връзка, Azure предлага езикови модели на OpenAI и предоставя достъп до когнитивни услуги и инструменти като OpenAI Studio.

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

**Списък с фигури и таблици**

* Фигура 1.1. Модел на взаимосвързаността между основните компоненти на SCM стратегията за доставка на готова продукция
* Фигура 1.2. Технологичен модел, представящ различни варианти за управление на клиентски поръчки в производствено предприятие
* Фигура 1.3. Модел на централизирана облачна система за управление на поръчки
* Фигура 1.4. Сравнение на моделите на облачни изчислителни услуги (IaaS, PaaS, SaaS) и традиционната локална инфраструктура, определяйки отговорностите за управление
* Фигура 1.5. Слоеве на облачна абстракция при основни и междинни модели
* Фигура 1.6. Сравнение между компонентите на традиционна виртуална машина и Docker контейнер машина
* Фигура 1.7. Развитие между различни софтуерни архитектурни стилове за разработка и внедряване на облачни услуги
* Фигура 1.8. Сравнение на ориентиран към домейн дизайн с дизайн, управляван от данни, в контекста на времето и сложността при разработка на софтуер
* Фигура 1.9. Трислоен архитектурен модел
* Фигура 1.10. Надграждане на трислойния архитектурен модел
* Фигура 1.11. Диаграма представяща теоремата на CAP
* Фигура 2.1. Итеративен процес за проектиране на концептуален модел
* Фигура 2.2. Модел на зрялост на архитектурата
* Фигура 2.3. Диаграма на главен бизнес сценарий
* Фигура 2.4. Концептуален модел от високо ниво
* Фигура 2.5. Основни компоненти на облачна услуга за управление на поръчките и връзките между тях
* Фигура 2.6. Релационен (E-R) модел на структурата на потоци и събития тях
* Фигура 2.7. Последователност на етапите за извършване на доставка
* Фигура 2.8. Структурата на пакетите на всяка микроуслуга
* Фигура 2.9. Диаграма на класовете и техните връзки
* Фигура 2.10. Структура на Either монад
* Фигура 2.11. Примерен код за създаване на поръчка чрез използването на Either монад
* Фигура 2.12. Абстрактна диаграма на удостоверяване на потребител
* Фигура 2.13. Диаграма на оторизация чрез код и ключ за обмен (PKCE)
* Фигура 2.14. Диаграма на класовете, отговарящи за удостоверяване
* Фигура 2.15. Релационен модел на базата от данни за потребителите
* Фигура 2.16. Диаграма на последователността на бизнес сценарий за изпълнение на клиентски заявки
* Фигура 2.17. API Gateway в контекста на управление на поръчки
* Фигура 2.18. Скица на екран за вход в системата
* Фигура 2.19. Скица на основен екран на мобилно приложението
* Фигура 2.20. Скица на екран за доставка
* Фигура 2.21. Скица на екран за електронно доказателство за доставка
* Фигура 2.22. Главен екран в уеб портала, използван от диспечерите
* Фигура 2.23. Екран за маршрутизиране
* Фигура 3.1. Сравнение на работни рамки във връзка с използването им от проекти и предприятия
* Фигура 3.2. Доставчици на облачни услуги, използвани от предприятия
* Фигура 3.3. Категоризираща на публичните облачните компании
* Фигура 3.4. Географски региони на Azure
* Фигура 3.5. Схема на RabbitMQ
* Фигура 3.6. Архитектурна диаграма на софтуерните технологии, изграждащи облачната система
* Фигура 3.7. Работен поток, съчетаващ процесите на разработване и актуализиране на облачни услуги
* Фигура 3.8. Структура на подпроектите във Visual Studio
* Фигура 3.9. Обобщен модел на Azure Monitor
* Фигура 3.10. Пример за колебание при търсенето с Azure Monitor
* Таблица 1.1. Сравнение между права и обратна верига на доставки
* Таблица 1.2. Основни компоненти на стратегията за доставка на продукция „точно навреме“
* Таблица 1.3. Организационни единици в SAP модули за управление на финанси, продажби и на материали
* Таблица 1.4 Сравнение на публични, частни и хибридни облаци
* Таблица 1.5. Обобщение на методологията на дванадесетте фактора
* Таблица 1.6. Допълнение на методологията на дванадесетте фактора
* Таблица 1.7. Добри практики на облачната индустрия
* Таблица 2.1. Структурата на таблицата за „Поток“
* Таблица 2.2. Структурата на таблицата за „Събитие“
* Таблица 2.3. Видове разрешения на OAuth 2.0
* Таблица 2.4. Крайни точки на API Gateway и техните REST ресурси
* Таблица 3.1. Сравнение на работни рамки за разработка
* Таблица 3.2. Сравнение на услуги за данни със структура и характеристика
* Таблица 3.3. Сравнение на мобилни и уеб технологии за разработка
* Таблица 3.4. Услуги на Azure за хостинг и съответстващи случаи на употреба
* Таблица 3.5. Сравнение на уеб базирани платформи за софтуерно разработване и контрол на версиите
* Таблица 3.6. Стратегии за тестване на облачна система
* Таблица 3.7. Резултати от A/B тестването за създаване на поръчки и доставки
* Таблица 3.8. Резултати от A/B тестването за промяна на поръчки и доставки
* Таблица 3.9. Прогнозни месечни разходи за инфраструктура на облачни услуги

# Използвана литература

1. Александрова, Я. (2020). *Архитектура на аналитична система за управление на взаимоотношенията с клиентите*. Варна: Знание и бизнес.
2. Армянова, M. (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 стротелни 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. - c. 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 Калянджиев", 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/](https://www.capital.bg/biznes/promishlenost/2025/03/13/4737829_top_10_proizvoditeli_na_stroitelni_materiali_na/%20) > [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 Калянджиев.
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. Сълов, В. (2023). *Психология на изкуствения интелект*. Дигитализация, големи данни, изкуствен интелект, Варна: Наука и икономика, 2023, 74-76.
22. 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.
23. Сълова, С. (2023). *Методика за извличане и обработка на големи неструктурирани данни в логистиката*. Дигитализация, големи данни, изкуствен интелект : Сборник с доклади от научен семинар Икономически университет - Варна, 31.03.2023 г., Варна : Наука и икономика, 2023, 61-67.
24. Сълова, С., Банков, Б. & Стоянова, М. (2024) *Уеб технологии*. Варна : Наука и икономика.
25. 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.
26. 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.
27. 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.
28. Aleksandrova, Y. (2021). *Predictive Analytics Implementation in the Logistic Industry*. Economics and Computer, Varna : Knowledge and Business, 7, 2021, 2, 6-22.
29. 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), р. 63.
30. 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, рр. 703 – 708.
31. 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.
32. 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.
33. Atchison, L. (2020). *Architecting for scale: How to Maintain High Availability and Manage Risk in the Cloud*. O’Reilly Media.
34. 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
35. Batista, F. (2022). *Developing the ubiquitous language* - The Domain Driven Design. <<https://thedomaindrivendesign.io/developing-the-ubiquitous-language/>> [10.12.2023]
36. Becker, U., Herhuth, W., Hirn, M. (2016). *Pricing and the condition technique in SAP ERP*. SAP Press.
37. 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.
38. 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), рр. 1835 – 1856.
39. Bilovodska, O., Syhyda, L., & Saher, L. (2018). *Supply chain management: world’s companies experience*. MIND Journal, (5), 1-17.
40. 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), р. 171.
41. Bönnen, C., Drees, V., Fischer, A., Heinz, L., Strothmann, K. (2018). *SAP Gateway and OData*. SAP Press.
42. Braun, S., Bieniusa, A., Elberzhager, F. (2021). *Advanced Domain-Driven Design for Consistency in Distributed Data-Intensive Systems*. European Conference on Computer Systems.
43. Brewer, E. (2012). *Pushing the CAP: Strategies for Consistency and Availability*. IEEEComputer, 45 (2), рр. 23 – 29.
44. 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].
45. 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).
46. Caserio, C., Trucco, S. (2018). *Enterprise Resource Planning and Business intelligence systems for information quality: An Empirical Analysis in the Italian Setting*. Springer.
47. 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), рр. 536–553.
48. Ćatović, A., Buzađija, N., & Lemes, S. (2022). *Microservice development using RabbitMQ message broker*. Science Engineering and Technology, 2(1), 30–37.
49. Chen, Y. (2020). *Intelligent algorithms for cold chain logistics distribution optimization based on big data cloud computing analysis*. Journal of Cloud Computing, 9.
50. Chiang, C., Kou, T., Koo, T. (2021). *A Systematic Literature Review of the IT-Based Supply Chain Management System: Towards a Sustainable Supply Chain Management Model*.Sustainability, 13 (5), р. 2547.
51. 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), рр. 209 – 238.
52. 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]
53. Cloud Native Computing Foundation. (2023). CNCF Annual Survey 2023. CNCF. <<https://www.cncf.io/reports/cncf-annual-survey-2023/>> [17.05.2024]
54. 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]
55. De la Torre, C. (2023). *Containerized Docker application lifecycle with Microsoft platform and tools*. Microsoft Corporation.
56. De la Torre, C., Wagner, B., Rousos, M. (2024). *.NET microservices: Architecture for containerized .NET applications*. Microsoft Corporation
57. Debski, A., Szczepanik, B., Malawski, M., Spahr, S., Muthig, D. (2018). *A Scalable, Reactive Architecture for Cloud Applications*. IEEE Software, 35 (2), рр. 62 – 71.
58. 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.
59. Dotson, C. (2019). *Practical Cloud Security: A Guide for Secure Design and Deployment*. O’Reilly Media.
60. 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]
61. 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), рр. 500 – 514 .
62. 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.
63. Erl, T. (2007). *SOA Principles of Service Design*. Prentice Hall PTR eBooks.
64. Esposito, D. (2016). *Modern Web Development: Understanding Domains, Technologies, and User Experience*. Microsoft Press.
65. Evans, E. (2014). *Domain-Driven Design Reference: Definitions and Pattern Summaries*. Dog Ear Publishing.
66. Evans, E., Evans, E. J. (2004). *Domain-driven Design: Tackling Complexity in the Heart of Software*. Addison-Wesley Professional.
67. Fields, J., Harvie, S., Fowler, M., Beck, K. (2009). *Refactoring: Ruby Edition*. Pearson Education.
68. Fowler, M. (2010). *Domain-Specific Languages*. Pearson Education.
69. Fowler, M. (2012). *Patterns of Enterprise Application Architecture*. Addison-Wesley.
70. Fowler, M. (2019). *Agile Software Guide*. <https://martinfowler.com/agile.html> [10.12.2021]
71. 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]
72. Flexera. (2023). Flexera 2023 State of the Cloud Report <<https://info.flexera.com/CM-REPORT-State-of-the-Cloud> > [14.05.2024]
73. 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).
74. Gargeya, V. B., Brady, C. (2005). *Success and failure factors of adopting SAP in ERP system implementation*. Business Process Management Journal, 11 (5), рр. 501 – 516.
75. 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.
76. 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.
77. Gaur, M. (2020). *ERP Migration Challenges and Solution Approach for Digital Transformation to SAP S/4HANA for SAP customers*. Social Science Research Network.
78. Gartner. (2023). Magic Quadrant for Strategic Cloud Platform Services. < <https://www.gartner.com>> [30.03.2024]
79. González, R., Gascó, J. L., Llopis, J. (2024). *Towards organisation 4.0. An empirical study*. International Journal of Information Management, 75, 102746.
80. Gouigoux, J. (2024). *Enterprise Architecture with .NET: Expert-backed advice for information system design, down to .NET and C# implementation*. Packt Publishing Ltd.
81. Grafiati. (2022). *Academic literature on the topic “Twelve-factor app”* O’Reilly Media.
82. Guo, Z., Guo, C. (2013). *A cloud-based decision support system framework for order planning and tracking*. Springer eBooks, pp. 85 – 98
83. Gupta, S. M. (2016). *Reverse supply chains: Issues and Analysis*. CRC Press.
84. Hahn, G. J. (2019). *Industry 4.0: a supply chain innovation perspective*. International Journal of Production Research, 58 (5), рр. 1425– 1441.
85. Hartley, J. L., Sawaya, W. J. (2019). *Digital transformation of supply chain business processes*. Business Horizons, 62 (6), рр. 707 – 715.
86. 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.
87. Henning, S., Hasselbring, W. (2022). *A configurable method for benchmarking scalability of cloud-native applications*. Empirical Software Engineering, 27 (6).
88. 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]
89. Hildebrand, K. (2018). *Master Data Life Cycle – Management der Materialstammdaten in SAP*. Springer eBooks, pp. 299 – 310.
90. 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, рр. 432 – 445.
91. Hoffman, K. (2016). *Beyond the twelve-factor app: Exploring the DNA of Highly Scalable*. Resilient Cloud Applications.
92. 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), рр. 945 – 955.
93. Huang, D., Xing, T., Wu, H. (2013). *Mobile cloud computing service models: a user-centric approach*. IEEE Network, 27 (5), рр. 6 – 11.
94. Indrasiri, K., Suhothayan, S. (2021). *Design patterns for cloud native applications*. O’Reilly Media.
95. Ingeno, J. (2018). *Software Architect’s Handbook: Become a successful software architect by implementing effective architecture concepts*. Packt Publishing.
96. Kaspersky. (2024). *Security Bulletin Statistics*. Securelist <https://securelist.com/ksb-2024-statistics/114795/> [07.01.2025]
97. Karthikeyan, S. A. (2021). *Demystifying the Azure Well-Architected framework: Guiding Principles and Design Best Practices for Azure Workloads*. Apress.
98. 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), рр. 5318 – 5339.
99. 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), рр. 965 – 1002.
100. 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), рр. 341 – 472.
101. Khan, S. A. R., & Yu, Z. (2019). *Introduction to supply chain management*. EAI/Springer Innovations in Communication and Computing, pp. 1 – 22.
102. Khononov, V. (2021). *Learning Domain-Driven Design*. O’Reilly Media.
103. Knolmayer, G. F., Mertens, P., Zeier, A. (2012*). Supply chain management based on SAP systems: Order Management in Manufacturing Companies*. Springer Science & Business Media.
104. Kumar, V., Agnihotri, K. (2021). *Serverless Computing Using Azure Functions: Build, Deploy, Automate, and Secure Serverless Application Development with Azure Functions*. BPB Publications.
105. 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), рр. 139 – 154
106. Lano, K., Tehrani, S. Y. (2023). *Introduction to software Architecture: Innovative Design using Clean Architecture and Model-Driven Engineering*. Springer Nature.
107. Laszewski, T., Arora, K., Farr, E., Zonooz, P. (2018). *Cloud Native Architectures: Design high-availability and cost-effective applications for the cloud*. Packt Publishing.
108. Le, T. T. (2020). *Performance measures and metrics in a supply chain environment*. Uncertain Supply Chain Management, рр. 93 – 104.
109. Lee, K. L., Azmi, N. a. N., Hanaysha, J. R., Alzoubi, H. M., Alshurideh, M. T. (2022). *The effect of digital supply chain on organizational performance: An empirical study in Malaysia manufacturing industry*. Uncertain Supply Chain Management, 10 (2), рр. 495 – 510.
110. 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.
111. Likness, J., & Phillip, C. (2024). *Serverless apps: Architecture, patterns, and Azure implementation*. Microsoft < https://learn.microsoft.com/en-us/dotnet/architecture> [14.12.2024]
112. Magal, S., Word, J. (2013). *Business Process Integration with SAP ERP*. Packt Publishing.
113. Martin, R. C. (2008). *Clean Code: A Handbook of Agile Software Craftsmanship*. Pearson Education.
114. Martin, R. C. (2017). *Clean Architecture: A Craftsman’s Guide to Software Structure and Design*. Prentice Hall.
115. 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), рр. 299 – 334.
116. 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.
117. Meyer B. (2009). *Touch of class: Learning to Program Well with Objects and Contracts.* Springer Science & Business Media.
118. Millett, S., Tune, N. (2015). *Patterns, Principles, and Practices of Domain-Driven Design*. John Wiley & Sons.
119. 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), рр. 17 – 30.
120. Molamohamadi, Z., Tirkolaee, E. B., Mirzazadeh, A., & Weber, G. (2021). *Logistics and supply chain management*. Communications in computer and information science.
121. Moniz, A., Gordon, M., Bergum, I., Chang, M., & Grant, G. (2021). *Beginning Azure Cognitive services*. Apress eBooks.
122. Nacheva, R. & Sulova, S. (2018). *Approach to Exploring Users’ Expectations of Digital Services’ Functionality*. Ekonomiczne Problemy Usług Szczecin : Wydawnictwo Naukowe Uniwersytetu Szczecinskiego, 2018, 2 (131).
123. Nacheva, R. (2020). *Standardization Issues of Mobile Usability*. International Journal of Interactive Mobile Technologies, 14, 2020, 7, 149 – 157.
124. Nacheva, R., Sulova, S. & Penchev, B. (2022). *Where security meets accessibility: Mobile Research Ecosystem*. Springer eBooks, pp. 216 – 231.
125. 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
126. 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, рр. 296 – 314.
127. Nikolov.D. (2019). *Shipping pseudocode to production*. DotNetCurry.   
     <<https://www.dotnetcurry.com/patterns-practices/1497/deploy-pseudocode-production>>[13.01.2025]
128. Nivala, M., Seredko, A., Osborne, T., & Hillman, T. (2023). *Stack Overflow Annual Developer Survey.* < https://survey.stackoverflow.co/> [08.05.2023]
129. Oh, S., Koo, J., & Kim, Y. (2022). *Security interoperability in heterogeneous IoT platforms*. Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing.
130. 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)
131. 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
132. 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]
133. Palermo, J. (2018). *The Onion Architecture*. Programming with Palermo < <https://jeffreypalermo.com/2008/07/the-onion-architecture-part-1> > [15.03.2024]
134. Palermo, J. (2019). *.NET DevOps for Azure: A Developer’s Guide to DevOps Architecture the Right Way*. Apress.
135. 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]
136. 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.
137. 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.
138. Penchev, B. (2016). *Effectiveness of a Conceptual Model for Increased Mobile Banking Security*. Serdica Journal of Computing, 10, 2016, 1, p. 49–62.
139. Pjp, I. (2023). *Mastering SAP Master Data Governance (MDG): A Guide to Data Management and Governance*. Independently Published.
140. 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 Businessу Media, pp. 230 – 245.
141. Rademacher, F., Sorgalla, J., Sachweh, S. (2018). *Challenges of Domain-Driven Microservice Design: A Model-Driven Perspective*. IEEE Software, 35 (3), рр. 36 – 43.
142. 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.
143. Rajapakse, Duminda. (2023). *Integration Between ERP Systems And Supply Chain Management*. Studies in Communication and Media.
144. Ramakrishna, Y. (2022). *Handbook of Research on Supply Chain Resiliency, Efficiency, and Visibility in the Post-Pandemic Era*. IGI Global.
145. 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, рр. 1343 – 1365.
146. 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].
147. 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.
148. Rendle, M., Steiner, M. (2024). gRPC for WCF developers. Microsoft <<https://learn.microsoft.com/en-us/dotnet/architecture> > [30.03.2024]
149. 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).
150. Sandberg, E. (2025). *Strategic Logistics Management: Contemporary Principles and Practice*. Kogan Page Publishers.
151. Sánchez-Flores, R. B., Cruz-Sotelo, S. E., Ojeda-Benı́Tez, S., Ramírez-Barreto, M. E. (2020). *Sustainable Supply Chain Management — A Literature Review on Emerging Economies*. Sustainability, 12(17), р. 6972.
152. Sazanavets, F. (2022). *Secure SignalR endpoints for both web UI clients and IoT devices*. In Apress eBooks.
153. 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), р. 7557.
154. Schneider, R. (2020). *Practical Guide to SAP Business Partner Functions and Integration with SAP S/4HANA*. Espresso Tutorials
155. Schniederjans, D. G., Curado, C., Khalajhedayati, M. (2020). *Supply chain digitisation trends: An integration of knowledge management*. International Journal of Production Economics, 220, 107439.
156. 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, рр. 79 – 87.
157. 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), рр. 1019 – 1050.
158. Stoyanova, M. (2023). *Document Process Automation with Artificial Intelligence for Logistics Sector*. Izvestia Journal of the Union of Scientists - Varna. Economic Sciences Series, Varna : Union of Scientists - Varna, 12, 2023, 1, 190-197.
159. Sullivan, M., Kern, J. (2021). *The digital transformation of logistics: Demystifying Impacts of the Fourth Industrial Revolution*. John Wiley & Sons.
160. Sulov, V. (2024). *Render modes of .NET Blazor technology*. Journal of Informatics and Innovative Technologies, 1(6).
161. 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.
162. Sulova, S. (2023). *A Conceptual Framework for the Technological Advancement of E-Commerce Applications*. Businesses. Special Issue. Innovative Practices in E-Commerce and E-Management, Basel, Switzerland : MDPI Publ., 3, 2023, 1, 220-230.
163. 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]
164. 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]
165. Smith, S. (2025). *Architecting modern web applications with ASP.NET Core and Microsoft Azure*. Microsoft Corporation
166. Soper, C., Addie, S., Dembovsky, C. (2024). *DevOps for ASP.NET Core developers*. Microsoft Corporation
167. Stonis, M. (2024). *Enterprise application patterns using .NET MAUI*. Microsoft Corporation
168. Tang, C., Xia, H. (2023)*. Risk analysis and research of construction supply chain*. Highlights in Business Economics and Management, 11, рр. 155 – 160.
169. Templar, S., Hofmann, E., Findlay, C. (2020). *Financing the End-to-End supply chain: A Reference Guide to Supply Chain Finance*. Kogan Page Publishers.
170. 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]
171. 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), рр. 353 – 371.
172. Tunç, T., Büyükkeklik, A. (2017). *Reducing the Negative Effects of Seasonal Demand Fluctuations: A Proposal Based On Cost-Benefit Analysis*. *I*nternational Journal of Engineering Research and Applications, 07 (03), рр. 38 – 46.
173. Türkay, M., Saraçoğlu, Ö., Arslan, M. C. (2016). *Sustainability in Supply Chain Management: Aggregate Planning from Sustainability Perspective*. PloS One, 11 (1)
174. Toub, S. (2024*). Performance improvements in .NET 7*. Microsoft Corporation
175. 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, рр. 232 – 247.
176. Văcar, A. (2019). *Logistics and Supply Chain Management: An Overview*. Studies in Business and Economics, 14 (2), рр. 209 – 215.
177. Vasilev, J. (2015). *Providing Logistics Information by Web Services*. Directory of Open Access Journals (DOAJ)
178. Vasilev, J., Cristescu, M. P. (2019). *Approaches for information sharing from manufacturing logistics with downstream supply chain partners*. Conferences of the Department Informatics, 1, рр. 24 – 29.
179. Vasilev, J., Stoyanova, М. (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.
180. 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.
181. Vasilev, J., Nikolaev, R., Milkova, T. (2023). *Transport Task Models with Variable Supplier Availabilities*. Logistics, 7(3), р. 45.
182. 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), рр. 174 – 187.
183. Vernon, V. (2013). *Implementing Domain-Driven Design*. Addison-Wesley.
184. Vernon, V. (2016). *Domain-Driven Design Distilled*. Addison-Wesley Professional.
185. Verwijmeren, M. (2004). *Software component architecture in supply chain management*. Computers in Industry, 53 (2), рр. 165 – 178.
186. Vettor, R. (2023). *Architecting Cloud Native .NET Applications for Azure*. Microsoft Learn
187. Vettor, R., Molenkamp, S., & van Wijk, E. (2024). *Dapr for .NET developers*. Microsoft Corporation
188. Vieira, D. (2023). *Designing Hexagonal Architecture with Java: Build maintainable and long-lasting applications with Java and Quarkus*. Packt Publishing.
189. Villaça, L. A., Azevedo, L. G., Baião, F. A. (2018). *Query strategies on polyglot persistence in microservices*. ACM Symposium on Applied Computing.
190. Von Aspen, J. (2020*). First steps in SAP S/4HANA Sales and Distribution (SD)*. Espresso Tutorials.
191. Vettor, R., Smith, S. (2024). *Architecting cloud-native .NET apps for Azure*. Microsoft Corporation
192. Wlaschin, S. (2018). *Domain Modeling Made Functional: Tackle Software Complexity with Domain-Driven Design and F#.* Pragmatic Bookshelf.
193. Xu, R., Jin, W., & Kim, D. (2019). *Microservice Security Agent based on API gateway in edge computing*. Sensors, 19(22), 4905
194. Young, G. (2011). *Event Centric: Finding Simplicity in Complex Systems*. Addison-Wesley Professional.
195. 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), р. 962.
196. 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

* 1. Service.yaml

apiVersion: apps/v1

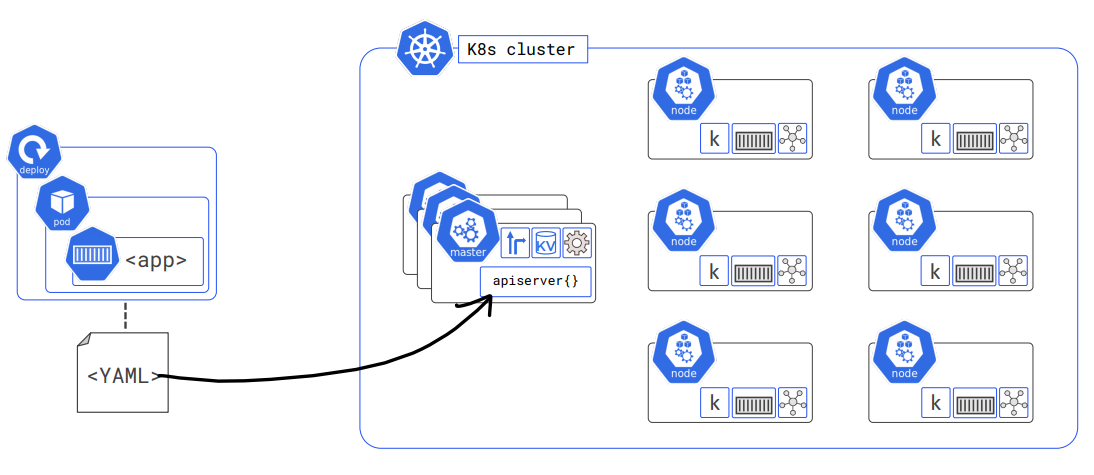
kind: Service

spec:

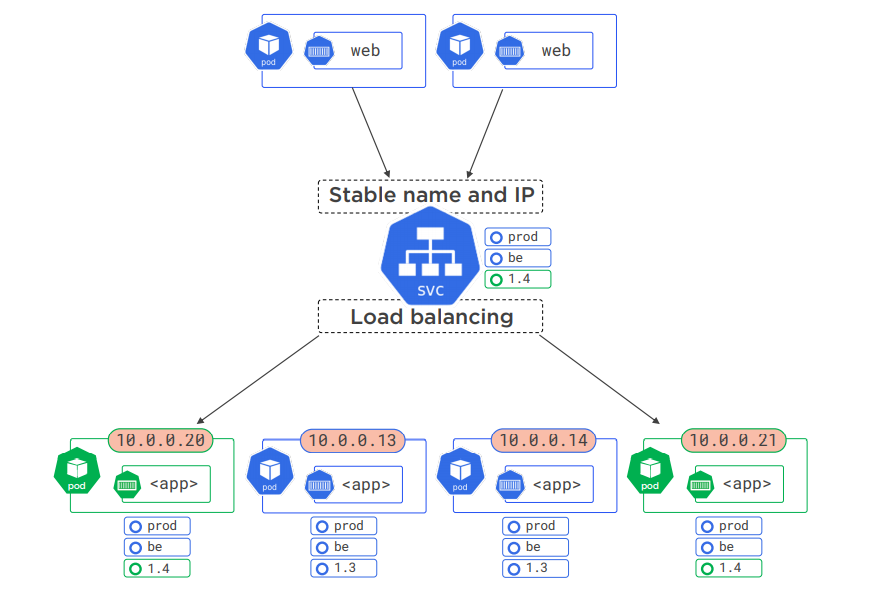
selector: app: manager-api

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.



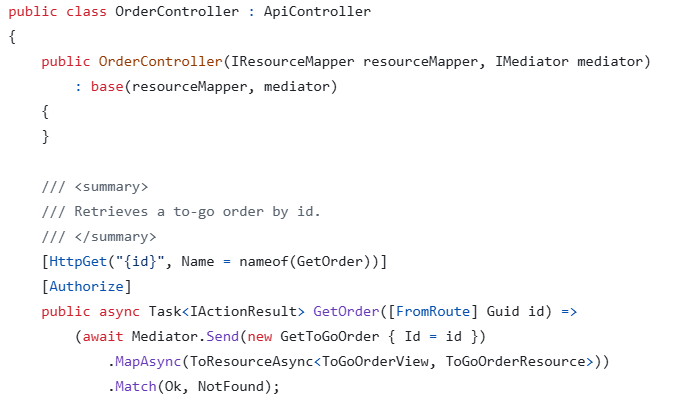
Deployment и Service ресурсите осигуряват IP адрес и DNS за достъп до микроуслугите, като същевременно балансират трафика между подовете и разпределят натоварването.



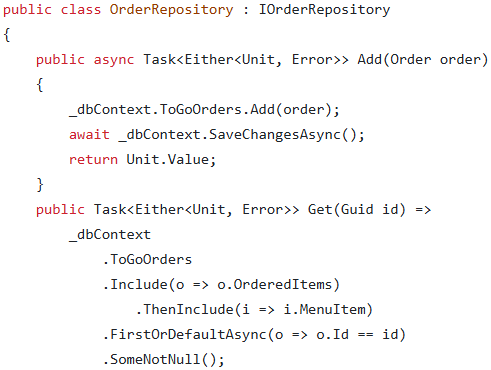
**Приложение 2**

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

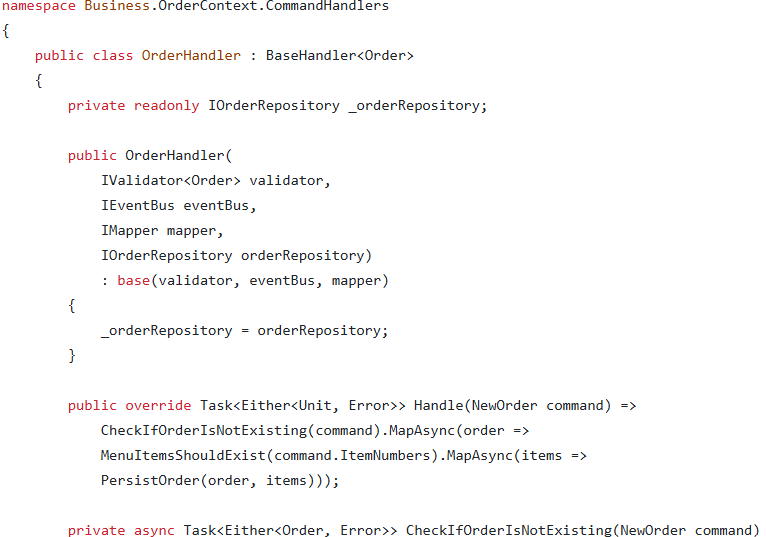
1. API контролер за поръчки



1. Връзка с базата от данни



1. Посредник за управление на данни



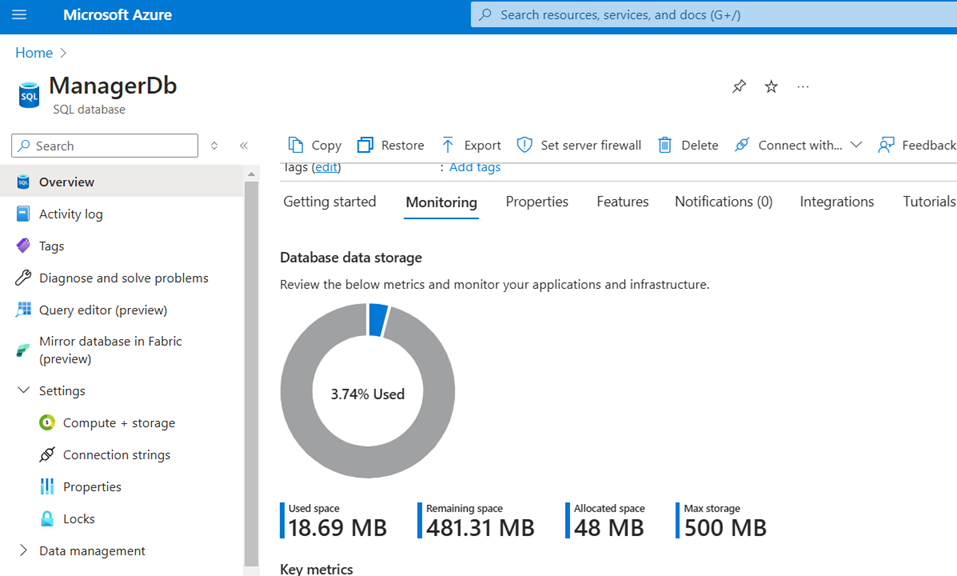
1. Синхронизация с ERP



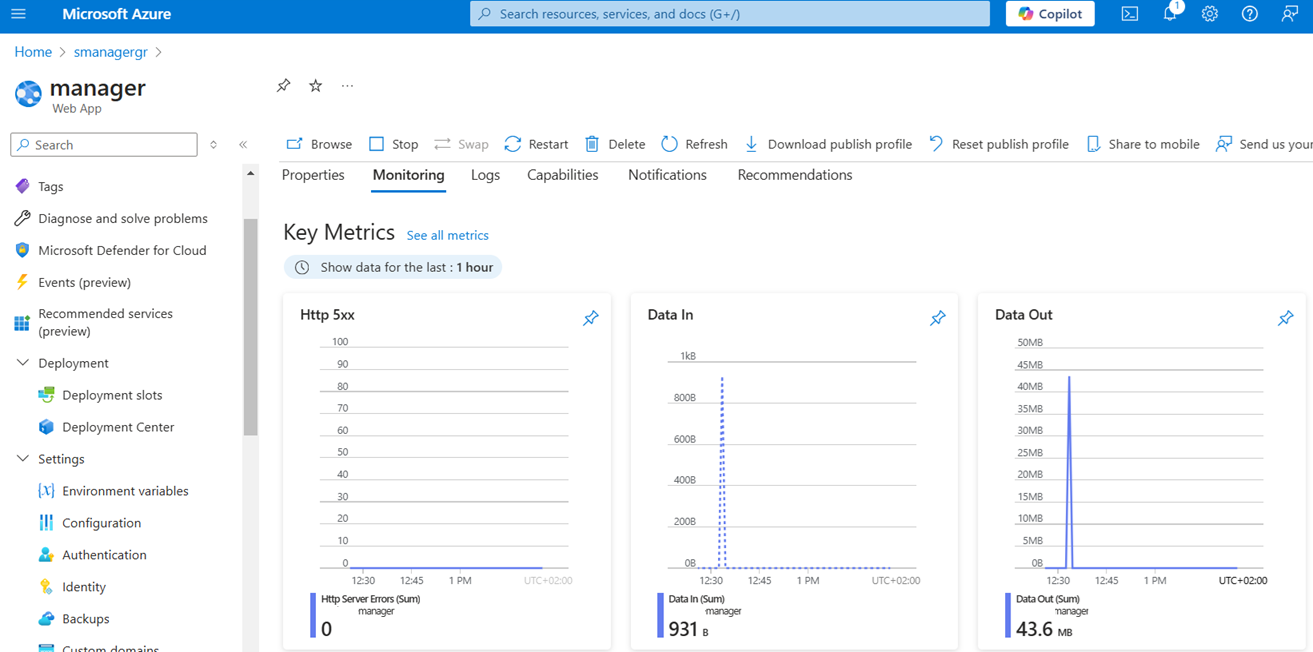
**Приложение 3**

**Изображения от облачните услуги на Azure, използвани по време на тестовете**

* 1. База от данни



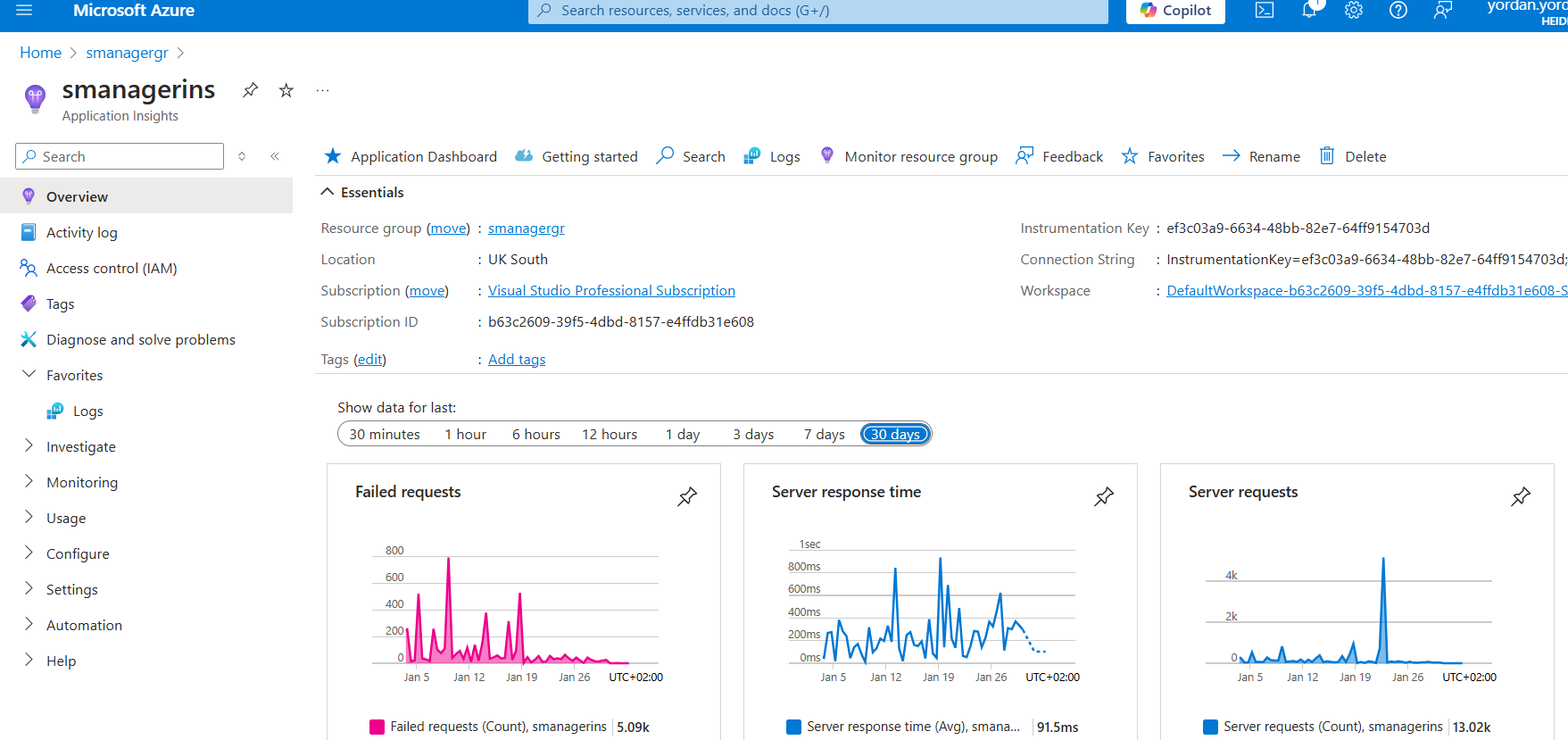
* 1. Хостинг услуга

****

* 1. Хранилище за файлове (Blob Storage)

****

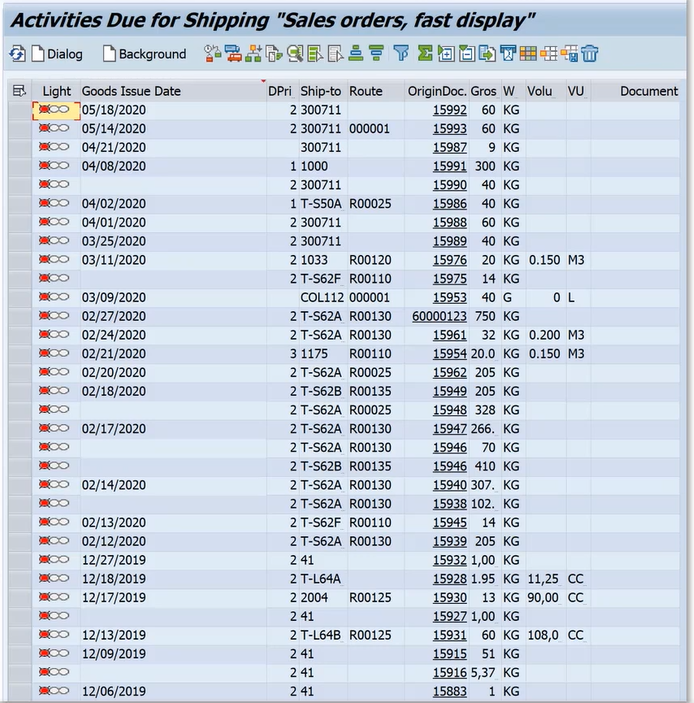
* 1. Мониторинг

****

**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.



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.



**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. 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)
3. 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)
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), рр.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.

**Справка за приносните моменти**

Основни приносни моменти на настоящия научен труд могат да бъдат обобщени по следния начин:

1. *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.

1. *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.

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. Основната цел на Индустрия 4.0 е да обедини производството с модерните информационни и комуникационни технологии, като същевременно да повиши ефективността и автоматизацията в бизнеса. [↑](#footnote-ref-1)
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. [↑](#footnote-ref-2)
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. [↑](#footnote-ref-3)
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 [↑](#footnote-ref-4)
5. S&P Global Ratings предоставя 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. [↑](#footnote-ref-5)
6. SAP NetWeaver Gateway е технология, която позволява свързване между SAP и персонализирани платформи. NetWeaver предоставя начин за интеграция чрез използване на API. [↑](#footnote-ref-6)
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. [↑](#footnote-ref-7)
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. [↑](#footnote-ref-8)
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. [↑](#footnote-ref-9)
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. [↑](#footnote-ref-10)
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. [↑](#footnote-ref-11)
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”. [↑](#footnote-ref-12)
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. [↑](#footnote-ref-13)
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. [↑](#footnote-ref-14)
15. Общият регламент относно защитата на данните (General Data Protection Regulation - GDPR), е законодателна рамка, приета от Европейския съюз с цел защита на личните данни на гражданите в ЕС. Налага на организациите задължения за прозрачност, отчетност и сигурност при обработка на лични данни, дава права на индивидите като достъп до данни, право на поправка, изтриване, ограничаване на обработката и преносимост на данни. Нарушенията на GDPR могат да доведат до санкции. [↑](#footnote-ref-15)
16. RFC 6749 очертава рамката за OAuth 2.0. Тази рамка въвежда основни принципи, които демонстрират как клиентските приложения могат да получават ограничен достъп до уеб услуги, без да разкриват идентификационните данни на потребителя.. [↑](#footnote-ref-16)
17. 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. [↑](#footnote-ref-17)
18. 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. [↑](#footnote-ref-18)