Въведение

За да изпълни целта, проучването е ориентирано като казус (case study). Този изследователски метод е често използван в социалните науки за изучаване на социални или културни феномени (Myers, 2009). Добре познат е в сферата на информационните технологии (Sarker at al., 2012). Yin (2019) разделя казусът на няколко варианта, въз основа на контекста на проучване, единица за анализ и др. Според Yin, дизайн на изследване на единичен случай (single case study design) е подходящ, когато се представя важен случай при тестване на добре формулирана теория с ясно дефинирани предложения. В комбинация с други техники, проучването стреми към добро разбиране на изучаваните единици за анализ.

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

1.1. Същност и принципи на информационните системи, поддържащи дейността на производствено предприятие

1.1.1. Характеристика на ERP системите

ERP информационните системи позволяват ефективното планиране на дейността на предприятие, в т.ч. разходи за обновяване на оборудването и инвестициите в производството на нови изделия (Банабакова, В. К, 2019). Те произхождат от RP технологията (Requirements/resource planning - планиране на потребностите/ресурсите) е една от прилаганите логистични технологии. Основната цел на RP е съкращаване на количеството на запасите от материали, незавършено производство и готова продукция, съгласуване на графика на доставките с работата на отделните производствени звена и процеса на закупуване (Филипов, Ст. Г., 2019). С логистичните технологии и тяхното приложение се цели осигуряване на оптимални решения в логистичната система (Благоев, Бл., 2010). Логистичната технология се определя като стандартизирана последователност (алгоритъм) на изпълнение на отделни логистични функции, и/или процеси в логистичната система или в отделни нейни функционални области (Сергеева, В., 2004). Някои от тези алгоритми и поддържащите ги информационно управляващи системи са получили и нормативна регламентация. Такива са MRP I и MRP II, за които са разработени и утвърдени международни стандарти ISO (Стоянов, Ст. Хр., 2019). ER спомагат преодоляването на проблеми свързани с управление на складовите наличности, като:

- Забавяне на постъпването на материали;
- Натрупване на излишни складови запаси;
- Нарушаване баланса на доставката;

- Намаление на ефективността на производство;
- Сложност при процеса на контрол и анализ на производствената дейност;

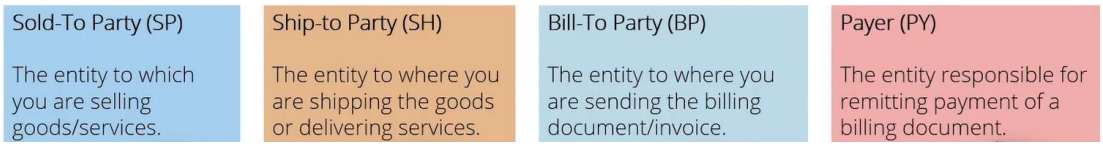
Подобренията от въвеждането на ERP се изразяват в увеличаване броя на изпълнените поръчки, повишаване качеството на логистичното обслужване към клиентите, възможности за промени в обема на поръчките, съкращаване на времето от поръчката до доставката и други (Банабакова, В. К, 2019).

Фокусът на дисертацията е върху модула за продажби и дистрибуция. Цялата функция на този модул е да „продава стоки и услуги на клиентите на предприятието“. Основен обект на модула за продажби и дистрибуция е „материал“. Той може да се закупи, произведе, продаде, върне и/или прехвърли. Фигура 1.1. представя данните, които са част от този запис: продажби и логистика, количества за материали и доставки и други:

***Фигура 1.1**: Данни, част от записа за материал в модул за продажби и дистрибуция.*

Вторият най-критичен обект от гледна точка на продажбите и дистрибуцията е главният запис на клиента. Той представлява субект, на който се продават стоки и/или услуги. Част от атрибутите на запис на клиент са: името и адреса на клиента, условия на плащане, специфични опции за ценообразуване, които може да се прилагат само за този клиент, както и различните партньорски функции. ERP системите поддържат няколко версии на клиент. Първата партньорска функция представлява субектът, на

който продаваме стоки и услуги (sold-to party). Партньор за доставка (sold-to partner) представлява мястото, където изпращаме стоки или услуги. С други думи, това е адреса за доставка на клиента, който може да е различен от адреса на купувача. Партньорът за фактуриране представя къде трябва да бъде изпратена фактура. Местоположението, на което се изпращат фактури, може отново да бъде напълно различно от това на адреса за доставка. Функцията партньор на платеца представлява субектът, който отговаря за плащането на фактура. В много случаи и четирите функции може да имат едни и същи данни, но в някои случаи тези четири функции са не само различни физически адреси, но и напълно различни обекти. Фигура 1.2. илюстрира на кратко четирите вида:



***Фигура 1.2**: Видове партньорски функции.*

TODO: ценообразуване / поръчките / продажби / доставки

1.2. Възможности за дигитализация на процесите по управление чрез прилагане на облачни технологии

В последните години облачните технологии се превърнаха във водеща тенденция в софтуерната индустрия. Те предоставят нов начин за изграждане на големи и сложни системи, като по този начин използват пълноценно съвременните практики за разработка на високо-качествен софтуер и налична инфраструктура. Това променя начина на проектиране, интегриране и внедряване на системите. Облачно базираните решения са проектирани да приемат бързо промените, да обслужват голям мащаб от хора и да бъдат устойчиви на всякакъв вид натоварване или хакерски атаки (Vettor, 2022).

Организацията Cloud Native Computing Foundation предлага следното определение: "*Технологиите, базирани на облак, дават възможност на организациите да създават и изпълняват приложения в модерни, динамични среди като публични, частни и хибридни облаци, чрез мрежи от услуги и микроуслуги. Качества на системите са устойчивост, висока наличност и достъпност, мащабируемост и управляемост, които са от критично значение за много от бизнес единиците. Автоматизацията на тези процеси позволява на инженерите да правят промени, с голямо въздействие, но с минимални усилия."*

Приложенията стават все по-сложни, като изискванията, от страна на потребителите, стават все повече и повече, главно насочени към бърза реакция и иновативни функции. Проблеми с производителността или повтарящи се грешки вече не са приемливи.

Предимствата на облачните системи поставят бизнеса една стъпка пред конкурентите. Бизнес системите се развиват от способностите на бизнеса да бъдат инструменти за стратегическа трансформация, която ускорява растежа на компанията. Облачно базираните системи се свързват главно с бързина (Smith, 2022).

…….

***Таблица 2.2**: Принципи за проектиране на облачни системи.*

име на български	име на английски	Описание (тук ще има доста цитати)
Разделяне на грижите	Separation of Concerns	всеки обект и модул трябва да бъде в своя собствена грижа и контекст
Капсулиране	Encapsulation	
Инверсия на зависимостта	Dependency Inversion	
Изрични компоненти	Explicit Components	
Единична отговорност	Single Responsibility	
Не се повтаряйте	Don’t Repeat Yourself	
Устойчивост и невежество относно инфраструктурата	Presentation Ignorance	
Ограничени контексти	Bounded Contexts	

1.3. Специфики при управление на поръчките от клиенти в производствено предприятие

Софтуерните приложения трябва да имат познание за потребителя или процеса, който ги извиква. Потребителят или процесът, взаимодействащ с приложение, е известен като принципал за сигурност, а процесът на удостоверяване и упълномощаване на тези принципали е известен като управление на самоличността (Vettor, 2022). Простите приложения могат да включват цялото им управление на самоличността в приложението, но този подход не се мащабира добре с много приложения и много видове принципали за сигурност. Windows, например, поддържа използването на Active Directory за предоставяне на централизирано удостоверяване и оторизация. Въпреки че това решение е ефективно в рамките на корпоративни мрежи, то не е предназначено за използване от потребители или приложения, които са извън домейна.

Удостоверяването е процес на определяне на самоличността на потребител. Упълномощаването е актът на предоставяне на удостоверено разрешение за извършване на действие или достъп до ресурс (Wike R, 2022).

Съвременните решения за самоличност обикновено използват токени за достъп, които се издават от защитена услуга/сървър (STS) на принципал за сигурност, след като тяхната самоличност бъде определена (Anil N, 2022).

Изискванията към STS:

- Защитават ресурсите;
- Удостоверяват потребителите;
- Осигуряват управление на сесии;

Важни функции на сървъра за самоличност са:

- услуга за удостоверяване, която да работи в централизиран процес;
- Единично влизане/излизане за множество приложения;

- Покрива индустриалните стандарти OpenID Connect и OAuth 2.0;
- Шлюз към Google, Facebook и др;

Удостоверяването чрез токени е механизъм без състояние, тъй като никаква информация за потребителя не се съхранява в паметта на сървъра или базата от данни, за разлика от бисквитките (Gichuhi , 2021).

Стандарт (RFC 7519) за уеб приложенията е JSON уеб токен (JWT). Той се състои от три части:

- Заглавна - JSON обект, кодиран във формат base64. Съдържа информация за типа на токена и алгоритъма за криптиране;
- Полезен товар - съдържа информация за текущия потребител (потребителско име, роля и др). Тук не трябва да се включват чувствителни данни, защотп лесно се могат да бъдат декодирани със публични сайтове като jwt.io;
- Подпис – Използва се от сървъра, за да провери дали токенът е валиден. Той се генерира чрез комбиниране на двете части (заглавна и полезен товар) заедно. Базира се на таен ключ, който само сървърът за удостоверяване знае. По този начин злонамерен потребител не може да фалшифицира валиден токен;

Фигура 1.3.1*: Пример за токен.*

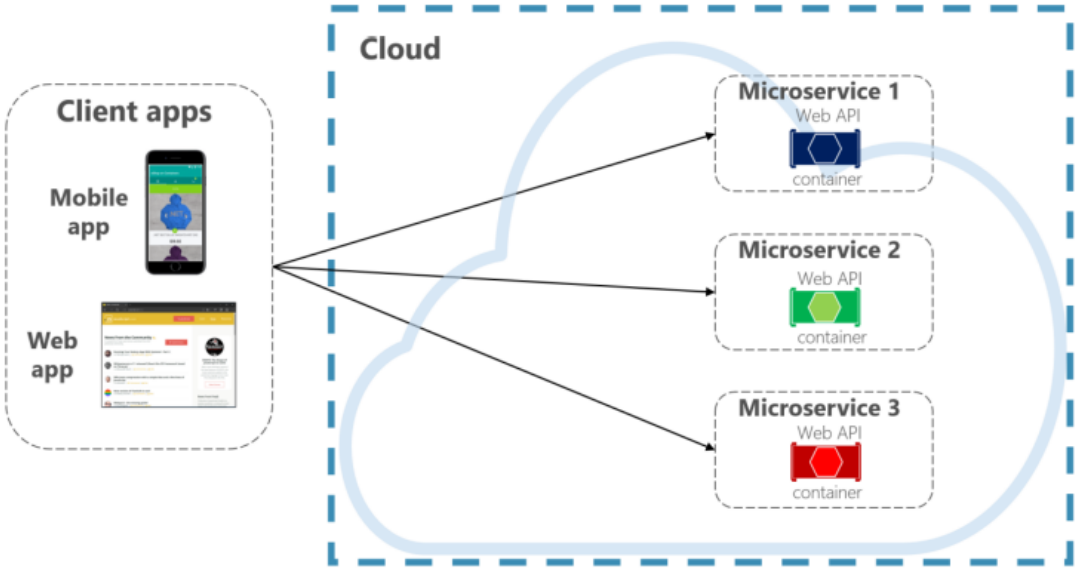
Глава 2. Архитектура на облачна система за управление на поръчки от клиенти

Тази глава разглежда решение от високо ниво, което да съсредоточава върху всички основни потребителски, бизнес и ИТ изисквания. Важна част от глава 2 са градивните елементи и интерфейси, изграждащи системата, както и комуникационните модели, които да ръководят композицията. Сложността на операциите стреми да бъде сведена до минимум. Представени са всички случаи на употреба и бизнес сценарии, съвместно с които се моделират приложенията за обслужване на клиенти. Освен това дизайнът обхваща функционалност, използваемост, устойчивост, производителност, икономически, технологични ограничения, компромиси и естетически проблеми на бекенд частта.

2.1. Същност, цел и обхват на софтуерната архитектура

Думата „**архитектура**“ често се използва в контекста от високо ниво, което е отделено от детайлите на по-ниско ниво (Martin et al., 2017). Софтуерният продукт, разглеждан в настоящия труд, се състои от 2 клиентски приложения, които се свързват към разпределена бекенд система, базирана на микроуслуги, работеща върху множество процеси и сървъри (хостове). Всяка услуга се изпълнява в отделен процес като контейнер, разположен в клъстер от виртуални машини. Това разделение на подсистеми и отделни нива и компоненти цели да постигне разбираемост и лесна поддръжка. Работните рамки са съвместими на всяко ниво, без да се дублират функционалности.

На фигура 2.1 са показани приложенията, които изграждат системата за управление на поръчките от клиенти.



Фиг. 2.1. Диаграма от високо ниво на главните приложения (разработка на автора)

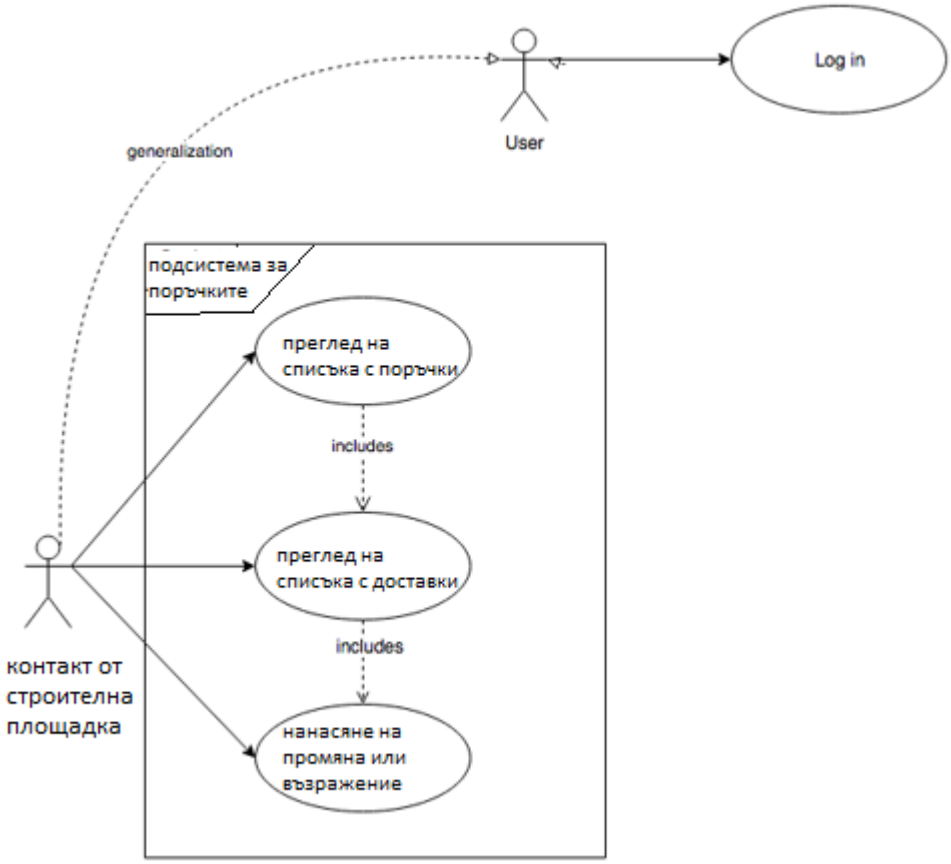
2.1.1. Ключови бизнес процеси и дейности свързани със системата за **управление на поръчките**

Тази подточка представя важни случаи на употреба, които са критични за бизнеса и са част от основния домейн. Използвани са UML диаграми на бизнес сценариите. Те идентифицират действия, които очакваме потребителите да направят.

Най-подходящ за взаимодействие с крайните потребители са мобилните приложения. Важни техни характеристики са, че поддържат функции като местоположение, камера и работят с уеб API. Клиентите на фирмата, които се явяват крайните потребители, управляват и проследяват поръчките и доставките в реално време с мобилно приложение. Целта му е да помага с планирането и логистиката, да въздейства върху крайния резултат с информация и данни. Тази информация, на смартфона, трябва винаги 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 **не**

активна съществуваща поръчка. Приложението може да се разпостранява безплатно чрез Google Play Store и Apple App Store.

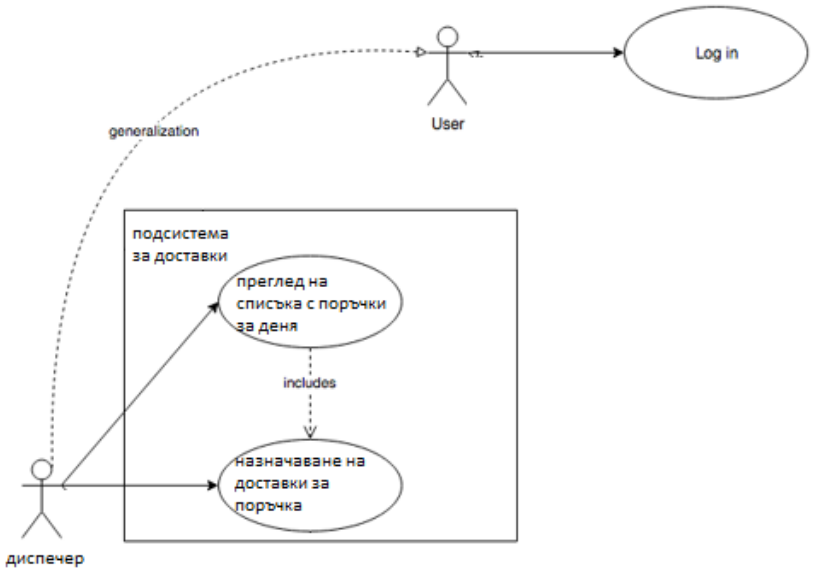
Обхватът на мобилното приложение, насочено към крайните клиенти, включва екран за вход, интерфейс за текущите поръчки и доставки към тях. Също така панел за създаване или промяна на поръчка. Фигура 2.2 представя процесите под формата на диаграма.



Фиг. 2.2. Диаграма на главен бизнес сценарий (разработка на автора)

Уеб порталът е софтуер насочен към диспечерите, е част от цялостната система за управление на транспорта (TMS). Чрез него могат да се създават поръчки и доставки, като същевременно се сравняват, за да се гарантира, че поръчките се доставят от най-подходящото превозно средство. Уеб порталът служи като инструмент за вземане на решения, с предварително зададени предложения, които могат да бъдат одобрени, или отхвърлени и променени, според гледната точка на диспечера на смяна. Вземайки под внимание текущите събития, подсистемите зад уеб портала насрочват за доставка това, което и когато клиентът е поръчал. Те разчитат на правилна информация за поръчка и актуализация на събития. Целта е да се минимизират разходите.

Обхват на уеб портала включва балансиране на работното натоварване на превозните средства, позволява проследяване и коригиране, както на поръчките, така и на доставките, осигурява предварително зададени решения, на база на които диспечерите могат да коригират и контролират броя на доставките. Диспечерите имат възможност да поправят грешни данни, като говорят с клиентите или шофьорите. Същевременно всички промени се отразяват в мобилното приложение.



Фиг. 2.2. Диаграма на главен бизнес сценарий (разработка на автора)

2.1.2. Функционални и нефункционални изисквания към бекенд системата

Ясно дефинираните изисквания са основата на успешен проект, тъй като включват набор от процеси като анализ, спецификация и валидиране. Функционалните изисквания са продуктови характеристики, които разработчиците трябва да внедрят, за да позволят на потребителите да изпълнят своите задачи. Като цяло функционалните изисквания описват поведението на системата при определени условия. Някои от основните изисквания са:

- Регистриране на акаунт;
- Вписване на потребител;
- Отписване на потребител;
- Преглеждане на текущите поръчки;
- Разглеждане на детайлите за определена поръчка и доставки;
- Филтриране на елементите по дата;
- Добавяне на нова поръчка;
- Промяна или премахване на съществуваща поръчка;
- Регистриране на нова доставка към поръчка;
- Промяна или премахване на съществуваща доставка;

Нефункционалните изисквания често се наричат „атрибути за качество“ на системата. Те са критериите за оценка на това как една софтуерна система трябва да работи.

Следващите точки отбелязват някои от основните изисквания:

• Системата трябва да е високо-достъпна и да може автоматично да разширява мащаба, за да отговори на увеличаващия се трафик (също така да намалява мащаба, след като трафикът спадне);

• Трябва да осигурява лесен диагностични дневници, за да помогне при отстраняване на неизправности или други проблеми, които би могли да възникнат по време на работа;

• Трябва да поддържа гъвкав процес на развитие, включително подкрепа за непрекъсната интеграция и внедряване (Continuous integration / deployment);

• Трябва да поддържа междуплатформен хостинг и развитие;

TODO: ИЗИСКВАНИЯ ЗА бързодействие и изпълнение / натоварване / ИЗИСКВАНИЯ ЗА ОБЕМ ДАННИ / брой потребители /

2.2. Концептуален модел на системата

Концептуалните модели са абстрактни представяния за това как трябва да протича изпълнението на задачите. Те представят визуално концепция или операция. За визуализиране и конструиране на елементите е използван унифицираният език за моделиране (Unified Modeling Language).

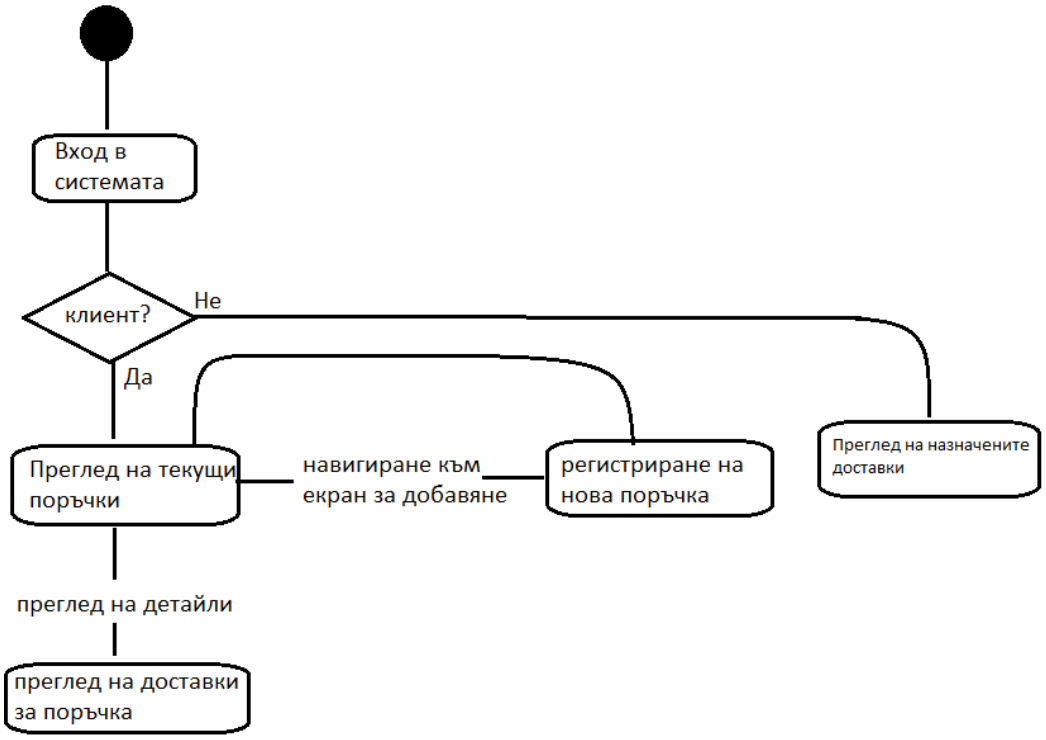
2.2.1. Поведенчески диаграми

Поведенческите диаграми идентифицират как различните елементи взаимодействат помежду си.

2.2.1.1 Диаграми за активност UML

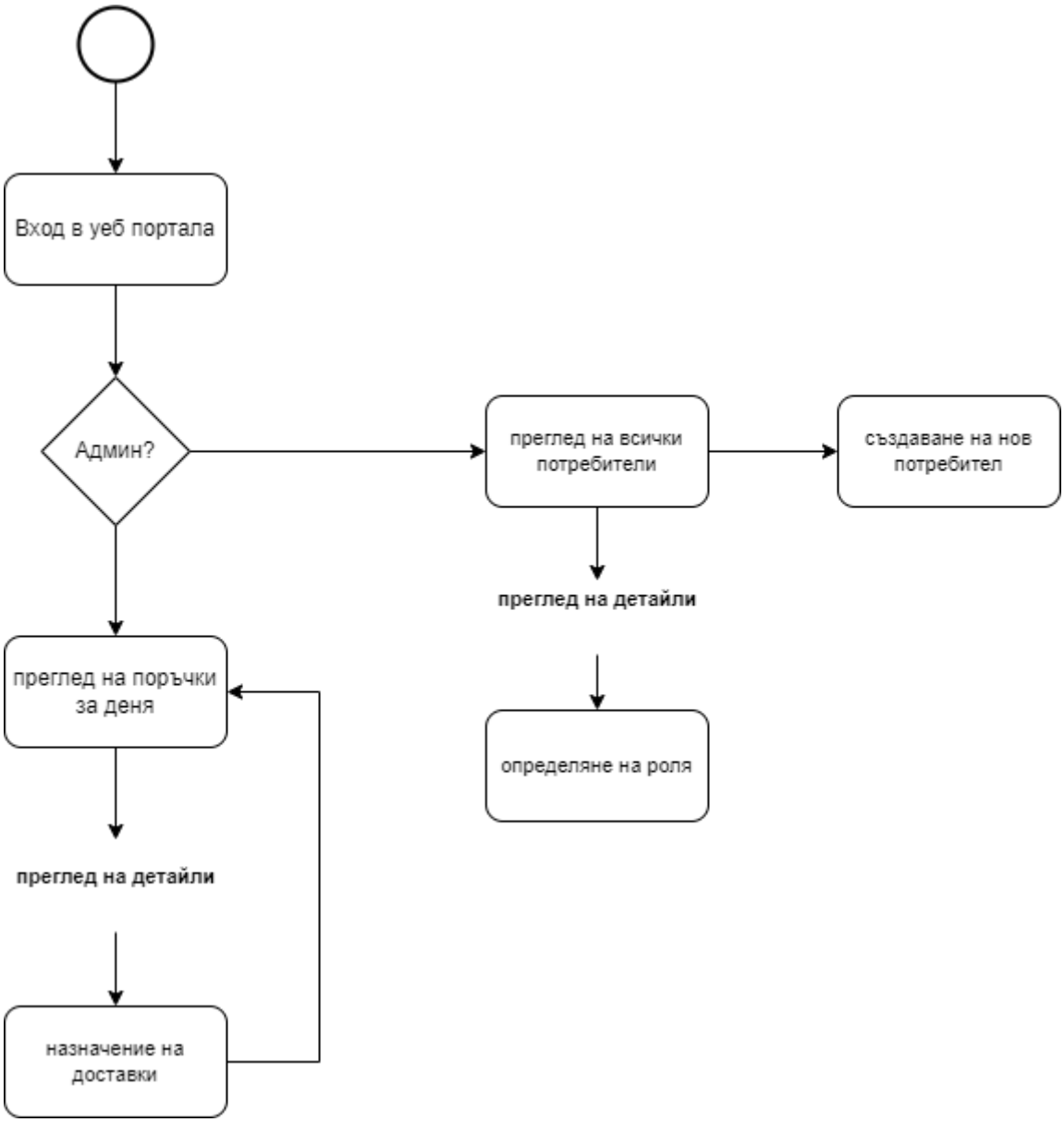
Диаграмите за активност изглеждат много подобни на блок-схемите. Наличието на тези прилики улеснява комуникацията между технически и не-технически лица (stakeholders).

Следната диаграма представя работни потоци и общи операции за мобилното приложение:



Фиг. 2.3. Диаграма на активноста за мобилно приложение. (разработка на автора)

Следната диаграма изобразява потока от операции в уеб портала:

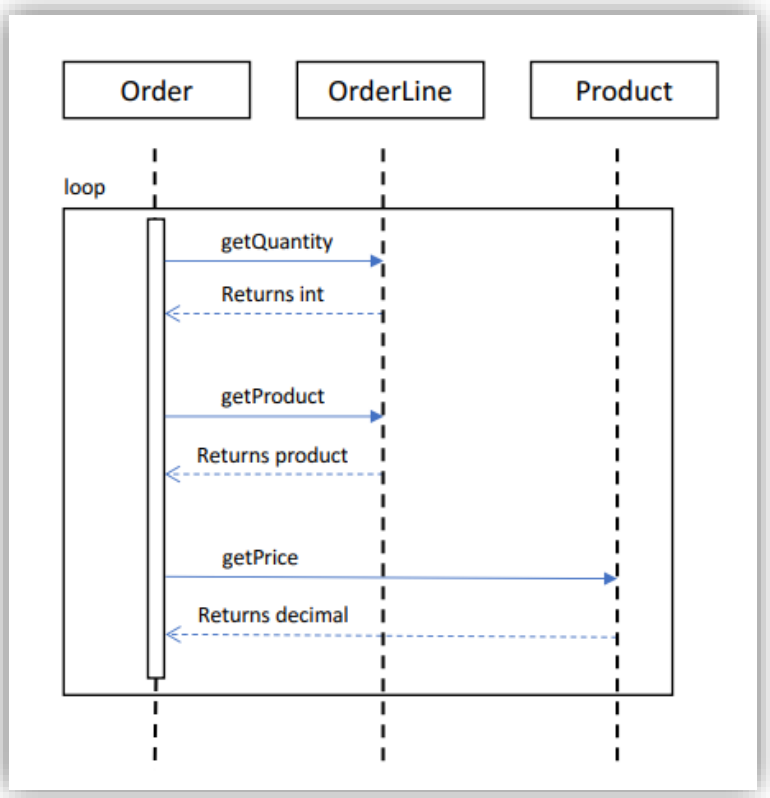


Фиг. 2.4. Диаграма на активност за уеб портал. (разработка на автора)

2.2.1.2. Диаграма на последователностите UML

Диаграмите на последователностите също са често използвани поведенчески диаграми в UML. Те идентифицират как обектите в система взаимодействат помежду си, за да реализират определена функционалност, като визуализират времевата линия и редът, в който се извършват операциите.

Пример може да разгледаме в контекста на подсистемата за поръчки:



Фиг. 2.7. Диаграма на последователностите. (разработка на автора)

https://blog.openreplay.com/jwt-authentication-best-practices

2.2.2. Структурни диаграми

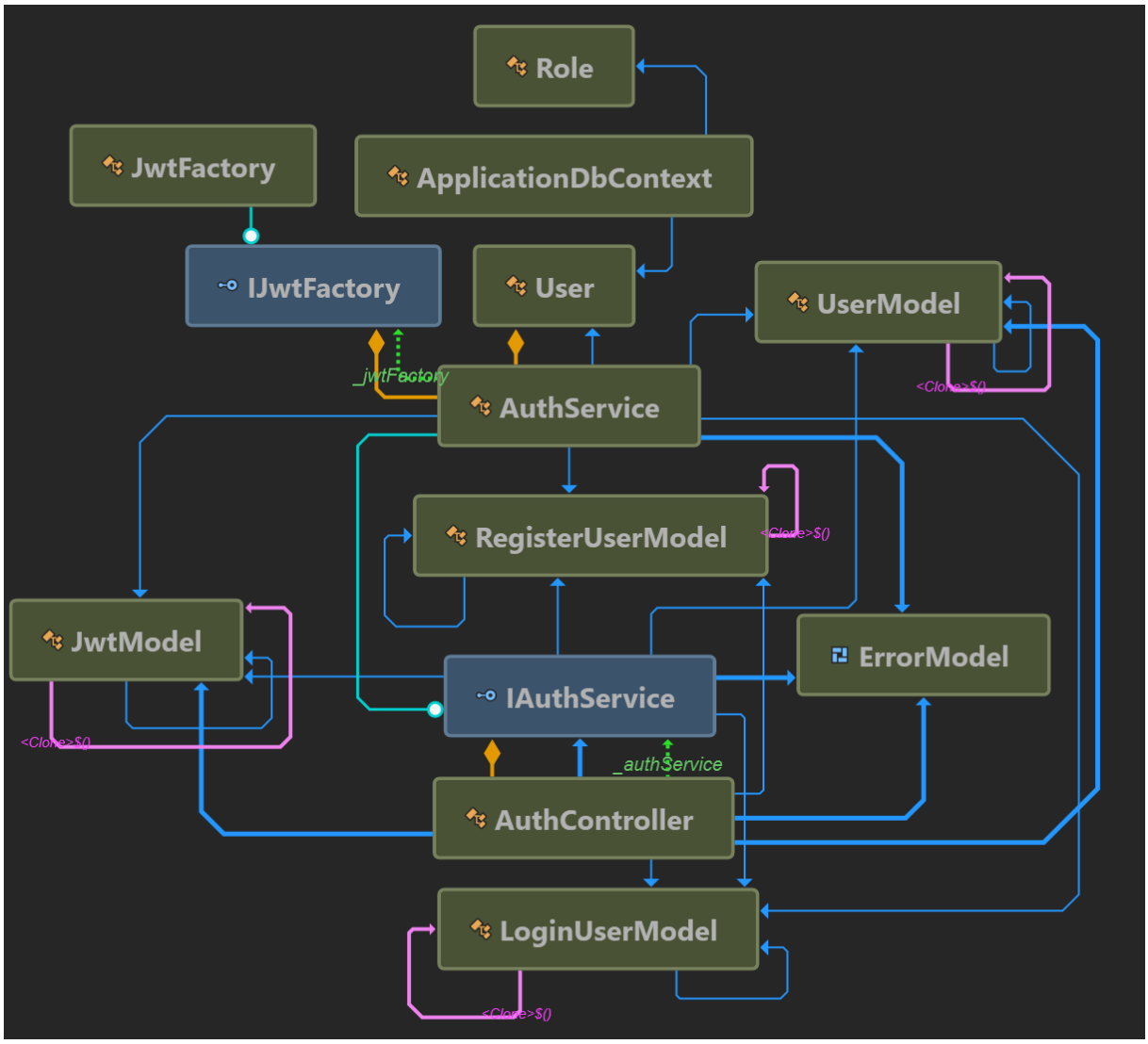
Структурните диаграми помагат за дефиниране цялостната структура системата, подобно на плана, който определя как изглежда една къща. Структурните диаграми моделират как изглежда системата в архитектурно отношение. Те ни помагат да дефинираме „речника“ на системата, гарантират съгласуваност от заинтересовани страни в проекта. Идентифицират различни връзки между различните частти.

Структурните UML диаграми изобразяват елементите на система, които са независими от времето и които предават концепциите и как те се свързват помежду си. Елементите в тези диаграми приличат на съществителните в естествения език.

2.2.2.1. Диаграма на класовете UML

Диаграмите на класове са едни от най-често срещаните, когато става на въпрос за разработката на софтуер. Едно от основните неща, които тези диаграми правят е да идентифицира речника на системата. Например, те определят връзките между обектите, които съответстват на основните съществителни.

Следващата част представя диаграма на класовете, свързани с удостоверяване. Това е процесът на определяне кой има достъп до системата. Елементите от приложението и зависимости, които обслужват тази част са визуализирани на фиг. 2.2. **DbContext** и **ApplicationUser** представляват комбинация от класове, които оперират с базата от данни. **AccountController** използва тези свойства чрез **UsersService**, който капсулира логиката по безопасен за използване начин.



Фиг. 2.8. Интерактивна диаграма за БСВС „Разработка на продукт / Планиране на проект". (разработка на автора)

Глава 3. Изграждане на облачна система за производствено предприятие HeidelbergCement AG

3.1. Обща характеристика на дейността на компанията HeidelbergCement

HeidelbergCement е немска мултинационална компания за строителни материали със седалище в Хайделберг, Германия. След придобиване на 45% акционерно участие в Italcementi, HeidelbergCement става производител номер едно на строителни инертни материали, номер 2 в цимент и номер 3 бетон в световен мащаб. В Forbes Global 2000 за 2020 г. HeidelbergCement бе класирана като 678-ата най-голяма публична компания в света. Дейността на фирмата е в около 60 страни с 57 000 служители, работещи в 3 000 производствени обекта. HeidelbergCement управлява 139 циментови завода с годишен циментов капацитет от 176 милиона тона, повече от 1500 места за производство на готов бетон и над 600 кариери за инертни материали.

3.1.1. Основни бизнес процеси в компанията

Продуктите на компанията се използват за изграждане на къщи, инфраструктура, търговски и промишлени съоръжения, като по този начин отговарят на нуждите на нарастващото световно население за жилища, мобилност и икономическо развитие. Основната дейност включва производство и дистрибуция на цимент, инертни материали, готови бетонови смеси и асфалт.С

3.1.2. Оптимизация на управлението и логистиката в индустрията чрез дигитализация

Дигитализацията е един от стълбовете на трансформацията. Тя преминава през всички бизнес линии и операции. Дигитални продукти, насочени към клиентите, целят да помогнат за успех в основния бизнес. (von Achten, 2022)

3.2. Избор на технологични средства за разработка и операции

Тази подточка, ще се опише едно от най-важните решения, най-вече защото е почти необратимо. Основни съображения за изпълнение на задачата са общност в Stack Overflow, Популярност според Google Тенденции, краен срок, тъй като напредналите технологии отнемат повече време и други.

Таблици 3.3 и 3.4 представят анализ на ползи и недостатъци на сървърни и мобилни технологии, като основните точки са:

- Тип - статичен или динамичен;
- Зависим от флатформа/инфраструктура;
- Обюност;
- Производителност;
- Крива на обучение;

***Таблица 3.3**: Сравнение на сървърни технологии за разработка.*

	App Types	Type System	Cross Platform	Community	Performance	Learning Curve
.NET	All	Static	No	Large	OK	Long
.NET Core	Web Apps, Web API, Console, Service	Static	Yes	Medium and growing rapidly	Great	Long
Java	All	Static	Yes	Huge	OK	Long
node.js	Web Apps, Web API	Dynamic	Yes	Large	Great	Medium
PHP	Web Apps, Web API	Dynamic	Yes	Large	OK -	Medium
Python	All	Dynamic	Yes	Huge	OK -	Short

***Таблица 3.4**: Сравнение на мобилни технологии за разработка.*

	Native	Hybrid	Cross Platform
Development Language & IDE	**iOS** – Objective-C or Swift, with X-Code & iOS SDK **Android** – Java with Android Studio & Android SDK	Thin wrapper around HTML, JavaScript, CSS	Xamarin (C#, Visual Studio) React Native (JavaScript)
Access to Phone's Features	Full control, no limits	Very limited	Catch-up with latest versions
User Experience	Exceptional	Inferior	Good, with limitations

Keep an eye on PWA!

3.4. Приложение на избраните технологии за изграждане на инфраструктурата в облачно базирана среда

По примери и указания от глава 2, тази подточка разглежда осъществяването на опростен във функционално отношение, облачен продукт, демонстриращ използването на .NET, Docker, Kubernetes в облачната среда на Microsoft Azure.

3.4.1. Софтуерното внедряване и поддръжка в облачна среда

(тази подточка ще бъде надградена с ориентир към системата за поръчки)

За изграждане, доставка и изпълнение на системи, изградени както като монолитни приложения, така и като ориентирани към услуги, се препоръчва използването на контейнеризирани технологии. Контейнеризацията е подход, в сферата на разработката на софтуер, при който кодът на приложение, всички негови зависимости и конфигурации са пакетирани в двоичен файл, наречен изображение. Изображенията са „шаблони“ само за четене и се съхраняват в регистър, който работи като хранилище или библиотека за изображения. Изображението се трансформира в работещ екземпляр на контейнер, който може да се стартира, спира, премества и изтрива. Създават се контейнери за различните части от приложението: уеб услуга, база данни, кеширане и др. Точно както транспортните контейнери позволяват транспортирането на стоки, независимо от товарите вътре, софтуерните контейнери се възприемат като стандартна единица за внедряване на софтуер, която може да съдържа различен код и зависимости. Контейнеризирането на софтуера дава възможност на разработчиците и ИТ специалистите автоматично да подновяват новите промени в различни среди. Контейнерите също така изолират приложенията едно от друго в споделена операционна система. Приложения се изпълняват върху хостът на контейнерите. От гледна точка на приложението, инстанцирането на изображение означава създаването на контейнер. Друго предимство на контейнеризацията е мащабируемостта.

Разширяването става бързо: създават се нови контейнери за краткосрочни задачи. Контейнерите предлагат предимствата на изолация, преносимост, гъвкавост и контрол в целия жизнения цикъл на приложението. Azure предоставя услуги, които могат да помогнат за постигане на много неща, варирайки от обикновени, като създаване на ново приложение с база от данни – до по-развити като създаване на работни потоци за непрекъсната интеграция (CI) и внедряване (CD). Това са само няколко примера за някои често срещани работни похвати. Много от тях трябва да бъдат създадени индивидуално, но облачната инфраструктура предлага всичко това като услуги. Силата на облака е, че ресурсите са невероятно устойчиви, малко вероятно е аварийно да спрат работа, тъй като центровете за данни са разположени по целия свят, състоящи се от десетки хиляди сървъри. Ако един сървър се повреди, друг поема управлението. Един от най-убедителните аргументи в полза на облака е, че може да разширява мащаба на услуги и ресурси почти безкрайно, в определени моменти, като например "Черен Петък" или голяма маркетингова кампания с промоции и намаления на артикули. Също така, когато натоварването намалее, мащабът може да се намали до обикновените си параметри. Уважавани и опитни облачни доставчици като Microsoft разпознават моделите на използване на нормалните потребители и тези на злонамерените. Инфраструктурата е предпазена от най-често срещаните атаки. Интелигентни инструменти за наблюдение, алгоритми за обучение и изкуственият интелект предоставят възможност да откриват атаки. При стартиране на приложения в Azure едно от първите решения, които трябва бъдат вземети, са планираните за използване услуги:

- Azure App Services - един от най-лесните и мощни начини за хостване на приложения. Той е предпочитан при монолитната архитектура. Услугите са достъпни и работят в 99,95% от времето. Споделят мощни функции като автоматично мащабиране, внедряване с нулев застой и лесно удостоверяване, позволяват отстраняването на грешки в приложението

докато работи в производствена среда (със Snapshot Debugger). По подразбиране приложението ще бъде достъпно в интернет, без да е необходимо да се настройва име на домейн или да се конфигурира DNS. Работи много добре с контейнери.

- Azure Virtual Machines - позволява преместване на съществуващи приложения от виртуални машини, които вече се изпълняват във център за данни. Има много предварително дефинирани изображения, които могат да бъдат използвани като Windows Server, който работи с IIS и има инсталиран и предварително конфигуриран ASP.NET на него, както и собствени софтуерни лицензи (като за SQL Server). Услугата е подходяща за мигриране на т.нар. „наследена система“, която да бъде използва като подсистема или източник на данни.

Следната таблица представя услугите и техните най-чести случаи на употреба:

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

Фиг. 8. Представя кои услуги на Azure са подходящи за различните типове.