**Shop-Storage система за разплащане, инвентаризация и логистика**

**Проектна група 12**

**Теодор Георгиев Николов, 77, 471218025, 12**  
**Томислав Валентинов Стаменов, 77, 471218038, 12**  
**Георги Анастасов Анастасов, 77, 471218044, 12**  
**Толга Сабахтин Ферад, 77, 471218047, 12**

**Въведение**

Известно е в световен мащаб, че днешния пазар е изключително забързан и динамичен. Развиват се множество малки и средни фирми, които се занимават с търговия, а по-големите преосперират все повече. От друга страна изискването на клиентите расте с всяка изминала година, те искат нещата да се случват бързо и лесно. Създават се множество магазини, складове, онлайн магазини и други обекти, които в забързания днешен живот се нуждаят от добро начало, с малко вложени инвестиции и с бърз и ефективен работен процес. Благодарение на развитието на технологиите и информационите системи, задоволяването на бизнеса и клиентите вече е възможно със създаването на софтуери насочени към всяка една сфера, строго специфични или универсални те помагат, развиват, забързват работните услуги.

Софтуерът, който създаваме е насочен към малки и средни магазини, складове и онлайн магазини с различна област в предлагането на стоки и продукти. Целта на софтуера е лесна организация на обекта, по-добро разпределение и по-достъпна информация за наличните продукти. Добра инвентаризация и проследяване на всички артикули в обекта. Възможност за лесно и бързо разплащане и проследяване на транзакциите и логистиката. Това ще е от полза както на управителите на обекта така и на служителите за по-бързи реакции и улеснение на работата.

**Обхват на проекта**

Софтуерът ще се разработва с пълна документация, която покрива целият жизнен цикъл на проекта с цел помощ на програмисти, разработчици, студенти за разработка на подобно приложение. Проектирането ще се осъществи изцяло на C# като програмен език с Windows Forms за графичен интерфейс предоставящ модули на потребителите. Разработването ще се осъществява на Visual Studio 2019. Ще се използват база данни на локален сървър предоставен от SQL Express 2019, а обработката им и лесното им управление ще се осъществява с SQL Management Studio 2019. Тази документация ще позволява на дадения човек, който ще разработва приложението, да разбере основите и как работно приложение може да бъде изградено от нулата. Тя позволява на разработчиците да разберат концепцията за приложението диаграми и различни сценарии.

За разработка на приложението се изискват познания в софтуерните архитектури, работата и обработката на база данни и основно писането и програмирането на C#. След изработване на приложението, то може да бъде пуснато за търговски цели или използвано от разработчиците.

**Цели и изисквания – приложение**

**Функционални изисквания:**

* Разпределяне на работата между служители и управители в два модула (модул служител, модул администратор)
* Лесна и бърза организация на обекта(служители и администратори)
* Разпределяне на продуктите (артикулите) в различни категории
* Добавяне, премахване, търсене и управление на продуктите
* Проследяване на наличните продукти на склад и техните спецификации
* Лесно и бързо разплащане и отпечатване на касови бележки
* Възлагане и обработка на доставки (логистика)
* Проследяване на транзакциите и логистичните операции
* По-добро разпределение и инвентаризация(преглеждане на склада)
* По-добра достъпност на информацията(модули с възможност за разделяне на информацията според спецификации)
* Преглеждане на архиви за по добра отчетност

**Нефункционални изисквания:**

* Производителност – системата трябва извършва обработка на заявки достатъчно бързо, за да не възпрепятства работата на служителите дори при увеличаването на операциите.
* Ефективност – системата трябва да бъде същевременно бърза и лесна за употреба за да бъде ефективна.
* Сигурност – системата трябва да гарантира, че данните няма да бъдат достъпни за трети лица и че няма да има външна намеса.
* Надеждност - системата трябва да бъде надеждна без възможност от спиране на софтуера поради грешка с цел добър работен процес.
* Използваемост - системата трябва да предлага добър софтуер с лесен графичен дизайн за оптимална комуникация между системата и служителя.
* Възможност за тестване - системата трябва да предлага възможност от провеждането на тестове за откриване на грешки и подобряване на работата.
* Разширяемост - системата трябва да предлага възможност за обогатяване на функциите и приложенията й.

**Актьори - участници**

Софтуерът се отнася до широка област от магазини, складове и онлайн магазини(физическите лица работещи в търговията)с голям и различен обхват в предлагането на стоки (продукти, артикули). Ще е предназначена за управителите, служителите и потребителите на дадения обект с цел по-добро и по-ефективно изпълнение на работата. Софтуерът ще улеснява и ще забърза работния процес в дадената фирма. Ще се отнася до различни по големина обекти като ще намали нуждата от голям ресурс от хора и високите разходи. Софтуерът ще превърне работната обстановка от чисто физическа в технологична. Ще бъде надежден, лесно разбираем с добър потребителски интерфейс и улесняващ работата.

Заинтересованите страни са хора или организации, които имат интерес или полза от системата. Заинтересованите странни в нашия проект са:

* Всеки, който управлява системата - Това ще са отделните администратори и служители, които са преките потребители на системата
* Всеки, който се възползва от системата – Това са купувачите или продавачите на стоки(продукти, артикули), които стоки минават през системата(тя спомага за инвентаризацията им записване на: количество, цена, информация, спомага както и за бързата работа: разплащане, принтиране на касови бележки, проследяване на транзакции).
* Всеки, който пряко или косвено участва в закупуването на системата - Имаме малки и средни бизнеси, които закупуват системата за своите обекти или големи компании интересуващи се от по нататъшно развитие на системата.
* Разработчиците на системата - Това сме ние екипа, които разработва софтуера и изцяло оформя неговия дизайн и работа.

**Перспектива на продукта**

Системата за разплащане, инвентаризация и логистика има голяма преспективав близкото и далечното бъдеще. Търговията заедно с информационните системи се разиват с високи темпове, взаймодействието и зависимостта между тях става вся по-голяма. Затова подобен вид системи ще продължат да навлизат, да се развиват и да се подобряват в тези сфери. Ако основните ползватели на системата подържа добре приложнието, зарежда го с различни стоки, актуализира неговото съдържание и го използват ефективно за работния процес, то тогава този софтуер никога няма да бъде извън употреба. Едно от важните качества на този вид системи са достъпността и използваемостта, тъй като те са достъпни и могат да се използват от всеки един служител от компютър или таблет.

**Архитектурен обзор**

Този раздел ще съдържа различните диаграми за описание на софтуера от всяка гледна точка и при всеки сценарии. Как разчилините участници със своите роли взаймнодействат помежду си. Какви са специфичните и общите изисквания и цели на системата. Също така се набляга на последователността от действия при изпълнение на програмата, различните класове и модули и займодействието между тях. Създаване на множество допълнителни диаграми, които могат да са в помощ при разработката и обяснението на приложението.

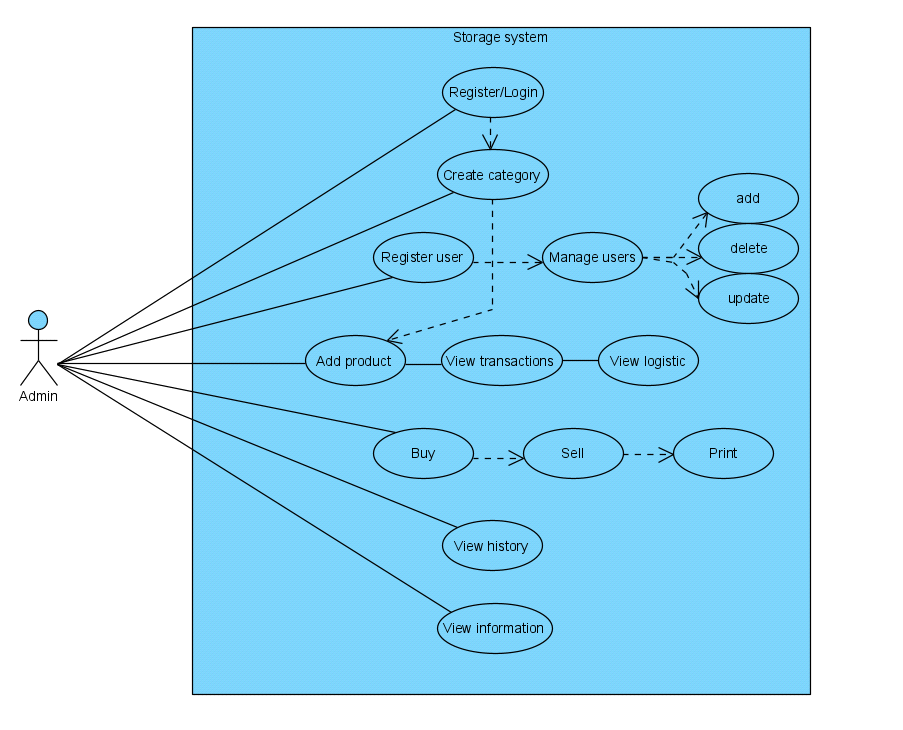
Резултатът от този раздел описва цялостта на софтуера, който е дефинирана като колекция от модули и подсистеми. Този етап създава основните, пъровначалните и фундаменталните компоненти от изграждането на приложението.

В описанието на софтуера ние ще наблегнем на абстракцията на разбработката. Ще се даде максимално абстрактна информация за софтуера, от какво ще бъде структуриран, как ще работи по време на изпълнение, какви функции ще има, как ще си взаймодействат участниците и какво трябва да се постигне. Абстракцията се описва с цел разбиране на софтуера по начина по който трябва да работи, а самото проектиране и разработване се оставя на преценката на разработчиците или се описва в друг раздел от документацията.

**Use-Case изглед**

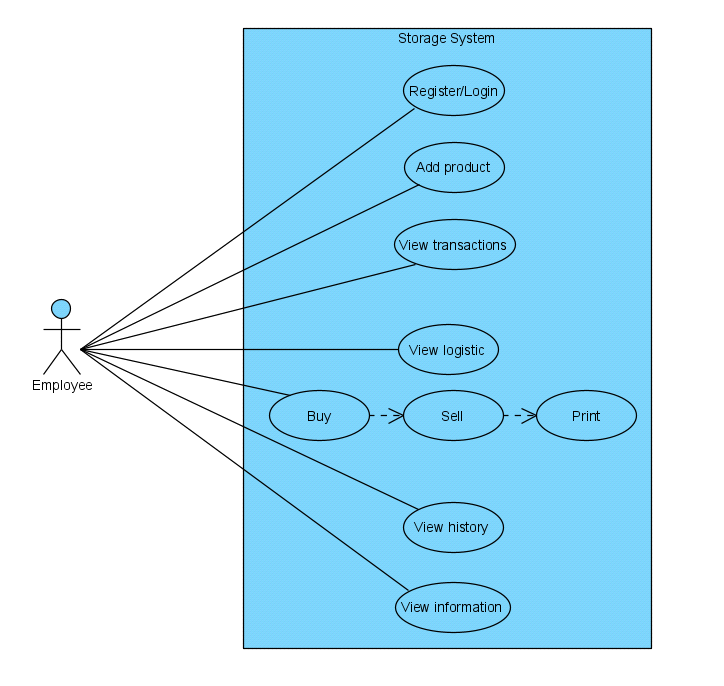
Представяме функционалните възможност на софтуера (основни сценарии) от гледна точка на ползвателите (актьори) с помоща на диаграми.

**Use-case diagram Admin:** Тази диаграма представя администратора, неговите функции и как той взаймодейства със системата. В началото управителят се регистрира или влиза в системата с потребителско име и парола. След което се прави проверка с дадените данни дали човекът е администратор или служител и според това той влиза в системата с една от двете възможности. Даденият човек е администратор и той влиза в модула на администатора, според който се предоставят функциите на системата. След като е влезнал в системата админът може да създава категории(група от продукти с характерни черти), за да улеснява и управлява инвентаризацията на обекта. Също така той може да регистрита и управлява служителите в системата(добавя още, премахва някои или актуализира тяхната информация).



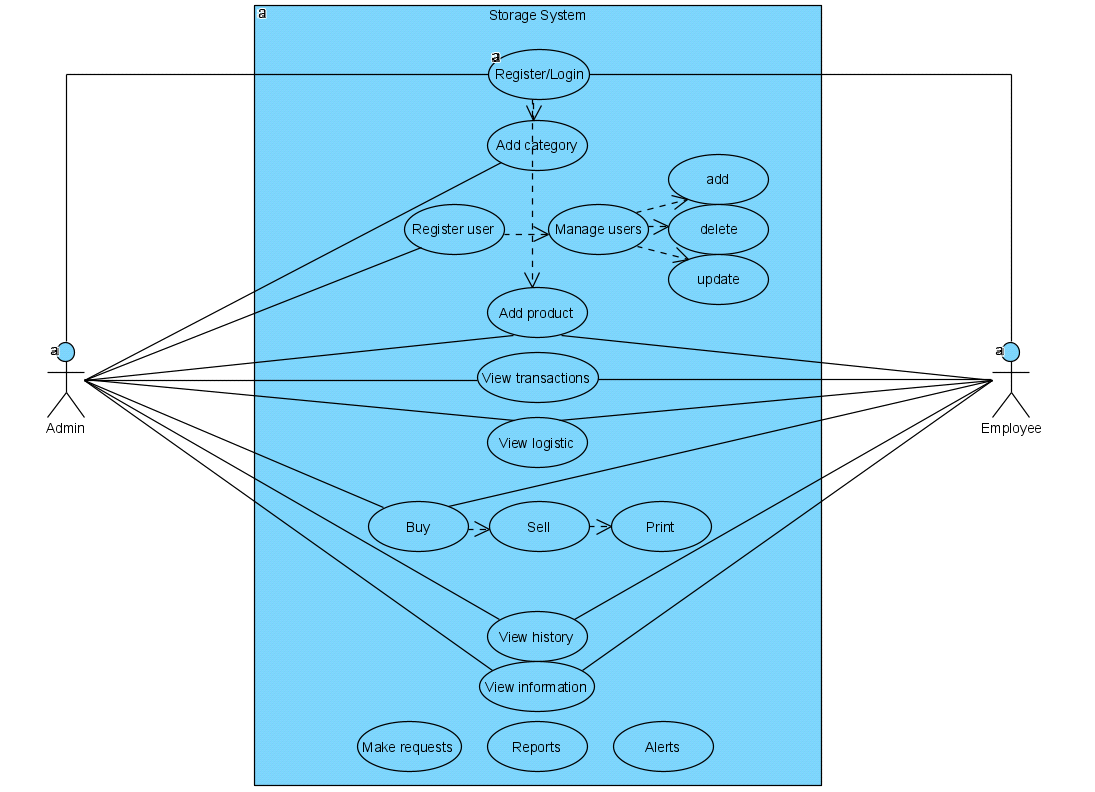
Админът е един вид служител в системата с повече правомощия, той може да добавя продукти под дадена категория, следователно да преглежда количеството продукти в склада и склада като цяло. Да наблюдава извършените транзакции от служителите или транзакции, които той е извършил и също така да възлага доставки на служителите и да променя някои от тях, да проследява логистичните операции, които са се осъществили. Админът за разлика от потребителя има право да проследява цялата история на обекта и да проследява важна информация за това какво се извършва в системата за даден период от време.

**Use-case diagram Empleyee:** Тази диаграма представя служителя, неговите функции и как той взаймодейства със системата. В началото служителят влиза в системата със своите потребителско име и парола и според неговите данни му се отваря модула за служители. Служителя може да добавя продукти, същевременно той може да изтрива, променя или да търси продукти в системата. Когато управителят(администратора) го няма в дадения обект, служителят може да добавя нови специални доставчици (големи фирми доставчици) или купувачи(специални клиенти на обекта). Служителя може да проследява транзакциите извършени от него, но само в определен период от време, той може да ги систематизира под различни направления(покупки или продажби).



Служителя може да продава и да закупува продукти и също така да отпечатва вид касови бележки. Служителя може да проследява историята на своето взаймодействие със системата(какви функции са извършени) отново за определен период от време. Ако му бъде възложена доставка служителят трябва да я изпълни в оказаното време. Също така той може да разглежда и информация предостъпена от системата спрямо ролята на служителя, като различна информация за продуктите, продажби покупки, количество, също така успешни неуспешни транзакции и логистични операции и важни и допълнителни съобщения от системата.

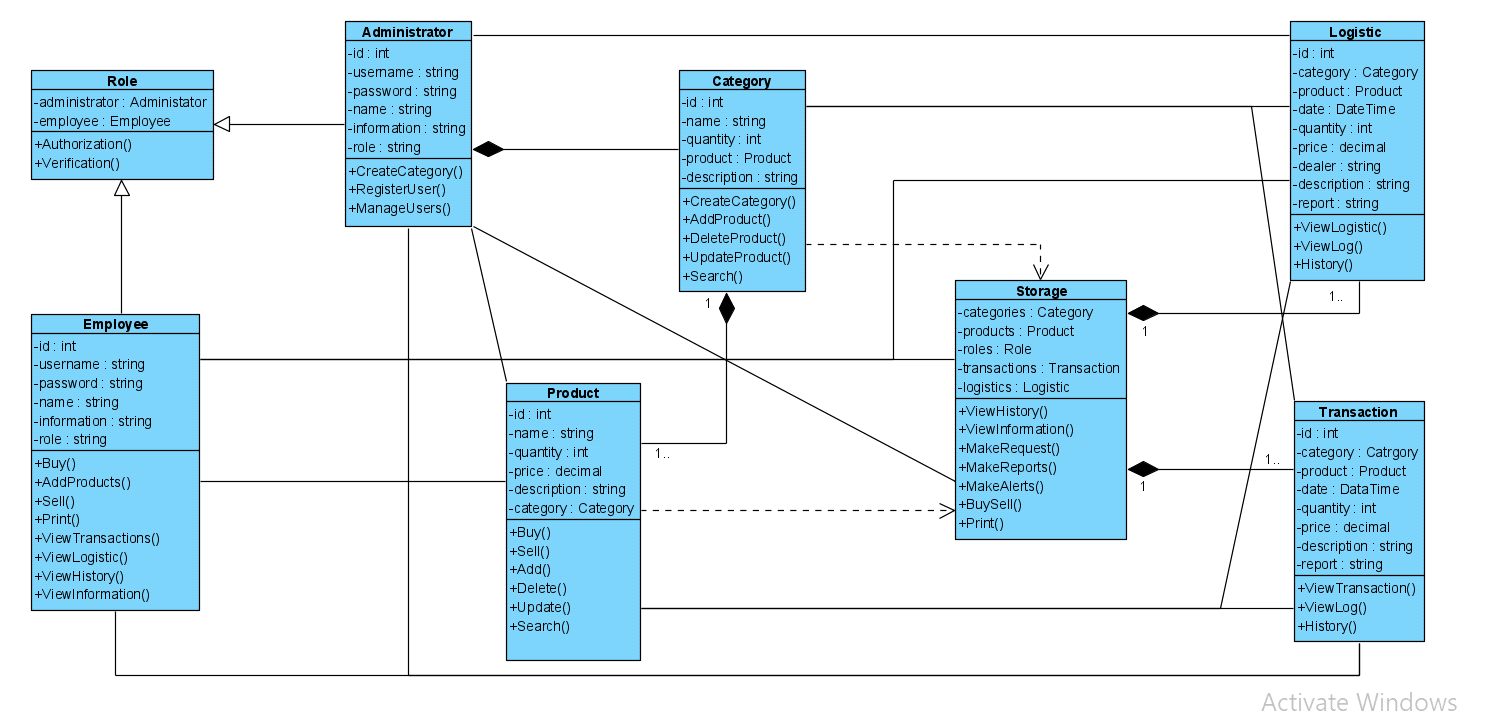
**Use-case diagram Admin/Empleyee:** Следващата диаграма представя всички участници с техните функции и как те си взаймодейства и изпълняват своите сценарии едновремено с някои допълнителни функции, които предлага системата. Цялата диаграма представя общата работа и абстрактната архитектура на системата за разплащане, инвентаризация и логистика и как се предполага тя да работи.



**Логически изглед**

Sофтуерът се представя като композиция от модули, пакети и класове. Използват се клас диаграми за да се илюстрират връзките между архитектурно зависимите компоненти, в комбинация с текстово описание.

**Class Diagram:** Тук представяме примерна клас диаграма на системата за разплащане, инвентаризация и логистика, която ще покаже класовете, които участват в системата, займодействието между тях и техните връзки. Целта на диаграмата е да въведе някои термини и да разясни архитектурата и организацията на системата. Може да се използва като един преход между бизнес анализаторите и разработчиците на софтуера, които заедно проектират системата.



На диаграмата виждаме разположението на класовете и техните връзки. В началото имаме клас роля, който клас ще определя ролята на администратора и служителя и ще отговаря за това всеки да влиза според ролята си в системата.

Следващия клас е класа на администратора, той има за атрибути id, което ще е много важна информация за системта и самата й работа за това кой какво извършва ще се следи според id на взаймодействащия със софтуера. Имаме атрибути потребителско име и парола, с които ще се влиза в системата според това дали са верни. Имаме допълнителни атрибути като име и инофмация, които ще дават допълнителни пояснения и естествено атрибут за ролята. Специалните операциите, които администратора може да извършва са създаване на категория, регистрация на потребител и управлението на вече съществуващи потребители както и всички други операции на служителя, които може също да извършва.

Следващия клас в системата това е класа на служителя, той отново има а атрибут id, което ще спомага много на системата да проследява, кой служител или администратор е извършил дадено плащане, купуване, транзакция, логистична операция и други. Имаме отново потребителско име, парола, и роля с които служителя ще влиза в системата. Имаме и допълнителните име и информация, които ще характеризират служителя. Неговите операции ще са свързани изцяло с управление на системата като купуване, продаване на продукти и оптпечатване на касови бележки. Ще може да добавя продукти в системата под дадени категории. Ще може да проследява транзакции, логистични операции, история и информация от системата като тези операции са възможни и от администратора.

Имаме и клас категория, който зависи изцяло от класа администратор защото той я създава. Категорията ще има отново id за помощ на системата. Категорията има атрибути: име, количество продукти в дадена категория, продукти и допълнително описание. С операциите ще може да се създава категория, да се добавят продукти в създадена категория, съответно да се изтриват, редактирати да се търсят в системата, като тези crud операции са възможни и за самата категория от странна на администратора.

Следващия клас е калса на продукта, който зависи от класа на категорията, защото без категория няма да има къде да се поставят продуктите в дадената система. Класа на продукта има също id като другите изброени класове. Продукта има подобни атрибути както на категорията: име, количество, стойност на продукта, допълнително описание и категория, в коят осе разпределя. Продукта може да се купува или продава, след което ако са закупени продукти те се добавят в системата. Продукта може да се премахва или редактира и да се търсят продукти по ключова дума или id.

Следващия и един от най-важните класове в системата е класа на самата система. Той ще определя почти цялата работа и ще предлага специфични функции. Атрибутите му се състоят от самите класове имаме атрибути: категории, продукти, роли в системта, транзакции и логистичните операции. Функциите и операциите, които ще изпълява са да може да се преглежда историята на това какво е извършено съ системата, да се преглежда и допълнителна информация. Той може да подава специални requests към служителите, да прави отчети или да дава информация за случващото се в даден момент и да отправя предопреждения към служитлите и администраторите при нужда. Чрез класа на системата може да се купува продава и да се отпечатват касовите бележки.

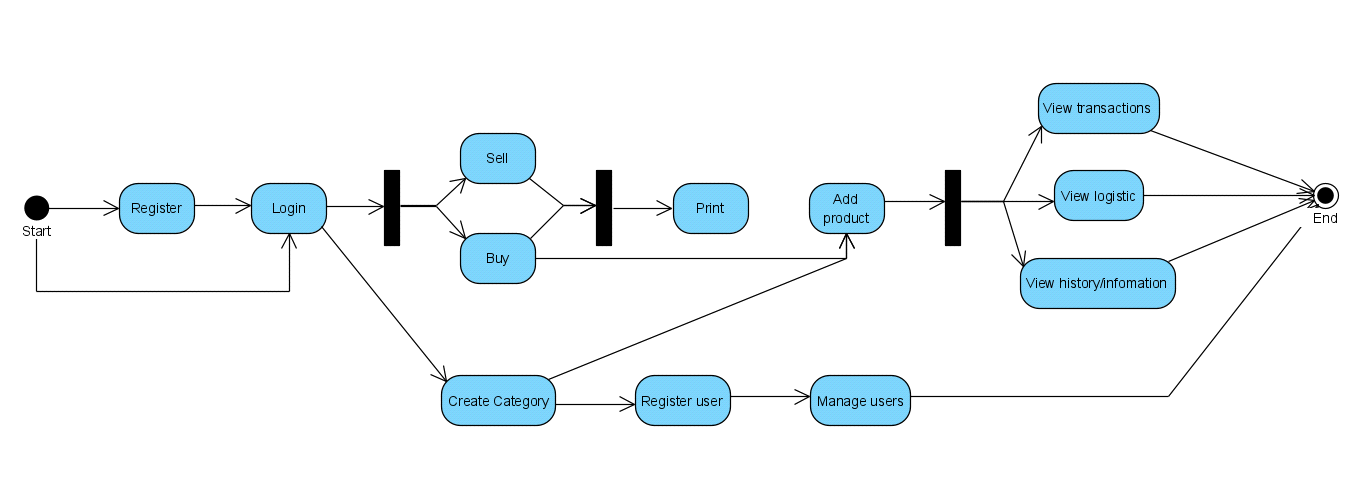
Следващия клас е класа на транзакциите, като в него отново първият атрибут е id, за да се проследяват всички транзакции нагледно. Следващите атрибути за категорията и продукта за да се знае какво е продадено или закупено в дадена транзакция. Имем и дата и час за повече информация кога се е случила транзакцията. Имаме допълнителните атрибути като количество, цена и допълнителна информация за извършената транзакция. И на последно място забелжка, ако се случило нещо важно при дадената транзакция. Класа на транзакциите има операции за проследяване на избрана транзакция или за проследяване на определен брой, както и преглед на цялата история.

Следващият клас в системата това е класа на логистичните операции, като тук отново на първо място стои атрибута id, който ще дава точна информация за дадена операция в системата. Следващите атрибути са сходни с тези на класа за транзакции имаме: категория, продукт, дата и час кога се е случила дадена логистична операция. Имаме количество, цена, допълнителна информация и ако е нужно забележка за някоя логистична операция. Тук имаме и атрибут доставчик, тоест записва се кой е направил доставката или от кого е получена доставката за по нагледно проследяване на операциите. Операциите, които ще изпълнява този клас са преглед на логистична доставка, преглед на много логистични доставки и предоставяне на цялата история.

**Процесен изглед**

Представя алгоритмите, реализиращи архитектурно значимите функции. Използват се activity диаграми в комбинация с текстово описания на действията.

**Activity Diagram:** Активити диаграмите помагат на разработчиците да разберат точно какви действия има и каква е тяхната последователност в системата или софтуера, който разбработват. Те помагат за точната представа какви функции има даденият потребител и как те трябва да следат една друга в процеса на работа на системата.



В нашата активити диаграма виждаме че първоначално човек трябва да се регистрира, ако вече има регистрация той може да влезе в системта. След като е влязъл в системата основните функции са разплащателните, тоест той може да купува или продава продукти и съответно да принтира касовите бележки, като ако е купил продукти той вече ги добавя в системата за инвентаризация. След което потребителя може да проследява транзакциите, които е извършил да разглежда логистичните операции или да проверява историята на това какво се е извършило със системта и да получава важна информация. Другото разклонение на активити диаграмата отговаря за допълнителните функции на администратора като той може да създава категории и да добавя след това продукти в тях. Също така да регистрира нови служители в системата или да управлява някои от съществуващите вече.

**Изглед на данните**

Представя модел на обработваните данни, включващ логическите връзки между тях. Ако приложението използва база от данни, се представя базата от данни, включваща типовете данни и релации.

Софтуерът ще използва голяма база данни с различни функционалности. Данните, които са важни за обектите ползващи софтуерът ще бъдат на първо място базата данни за потребителите, които използват системата.

* Потребители  
  - id: int  
  - first name: varchar  
  - last name: varchar  
  - email: varchar  
  - username: varchar  
  - password: varchar  
  - contact: varchar  
  - address: text  
  - gender: varchar  
  - user type: varchar  
  - added date: datetime  
  - added by: int  
  -added by name: varchar

Тук важно е да отбележем че базата данни на потребителите има един важен атрибут и това е ролята която определя дали потребителя е служител или администратор. Също така базата данни на потребителите има важните атрибути потребителско име и парола за да може да влизат в системата. Тази база данни ще е свързана с базата данни на транзакциите и на логистичните операции.

Имаме база данни за категориите от различни продукти, тяхното описание, броят категории и продуктите, които са в дадената категория.

* Категория  
  - id: int  
  - title: varchar  
  - description: text  
  - added date: datetime  
  - added by: int  
  - added by name: varchar

Дадена база данни от категория ще е свързана с база даннни от продукти, за да може продуктите да са вписани в дадена категория и под дадена категория да има множество продукти.

Следващите важни данни ще се отнасят до самите продукти, тяхното количество, цена, име и описание.

* Продукт  
  - id: int  
  - name: varchar  
  - category: varchar  
  - special number: varchar  
  - description: texy  
  - price: decimal  
  - quantity: int  
  - added date: datetime  
  - added by: int  
  - added by name: varchar

Тук базата данни на продуктите ще има специална характеристика това продуктът към коя категория спада и специален номер на

продукта(баркод) по който да се осъществява търсенето и самата система да има по добро разпределение и достъпност до информацията с продуктите. Продуктите си имат другите характерни черти като количество и цена. С тези данни ще може да се проследява склада в системата и самите транзакции.

Следващите важни данни ще се отнасят до доставчиците и купувачите, които ще се въвеждат в системата.

* Доставчик или купувач

-id: int

- type: varchar

- name: varchar

- email: varchar

- contact: varchar

- address: text

- added date: datetime  
- added by: int  
- added by name: varchar

Тук важните неща ще са свързани с типа, който ще определя дали добавеният човек в системата е доставчик или купувач. Други важни характеристики ще са неговото описание с име, email, контакт и адрес. И това кога, как и от кого е добавен човека в системата

Важна част от целия софтуер ще са и данните за самите транзакции, кога, как, по какъв начин е направена дадена продажба и нейното проследяване.

* Транзакция  
  - id: int  
  - type: varchar  
  - dealer or customer: int  
  - description: text  
  - total price: decimal  
  - transaction date: datetime  
  - tax: decimal  
  - discount: decimal  
  - paid amount: decimal  
  - return amount: decimal  
  - added by: int  
  - added by name: varchar

Тук имаме базата данни на транзакциите, която е важно да се свърже с базата данни на категориите, продуктите, потребителите и доставчиците или купувачите, за да се знае кой е извършил дадена транзакциия и какво и от кого е продадено или закупено. Важните определящи характеристики за тази база данни са цената, таксите, отстъпките, платеното и рестото които ще дават повече информация за транзакцията. Също така дата и описанието на това кога и какво е купено или продадено. Имаме и връзка с базата на доставчиците или купувачите за това кой участва в транзакцията.

Друга важна част е и проследяването на данните за логистиката, кога, от кого е доставена дадена стока.

* Логистична операция  
  - id: int  
  - employee: varchar  
  - first name: varchar  
  - last name: varchar  
  - address: text  
  - contact: varchar  
  - date: datetime  
  - description: text  
  - price: decimal  
  - added date: datetime  
  - added by: int  
  - added by name: varchar

Тук имаме базата данни на логистични операции, която е важно да се свърже с базата данни на категориите, продуктите и потребителите за да се знае кой е извършил дадена доставка и какво е продадено. Основни характеристики са описанието на служителя, адреса до където е доставено, контакт на получателя и много важно дата, с която после може да селектираме различните доставки в различните дни. Имаме и описание на това какво е доставено/продадено.

Софтуерът предвижда да предлага още функционалности свързани с данните като проследяване на най-продаван и най-непродаван продукт, пращане на заявки за поръчване при определен брой липса на продукти ако даденият служител не забележи, предлагане на отчети и история. Тези данни ще се събират и обработват в самата база данни и ще се предоставят на служителите в разбираем вид.

**Изглед на внедряването**

Представя хардуерните ресурси (потребителски работни станции, мобилни устройства и др.), върху които се изпълняват описаните процеси, както и начина на комуникация между тях.

Софтуерът е изцяло информационен и компютърно ориентиран, затова техническите изисквания към даденият обект, който ще използва софтуерът ще бъде наличието на добра информационна мрежа дали било то локална или интернет, наличието на компютри или лаптопи за по голямо удобство като с времето софтуерът може да се прехвърли и на таблети. Важно е и наличието на добро захранване и най-важното компютърна грамотност при служителите.

**Нефункционални изисквания описание**

Атрибутите за качество на софтуера са показатели, които описват предвиденото поведение на системата в средата, за която е предназначена. Софтуерната архитектура влияе дълбоко върху качествата, а от своя страна атрибутите за качество влият на софтуерната архитектура.

Имаме два типа изисквания функционални и нефункционални, като функционалните изисквания ги разгледахме в предишната документация. Те определят функцията на дадена система и нейните компоненти, тоест точна спецификация на найното поведение и това което тя се предполага да върши. Use case диаграмите определят точно това как всеки един потребител и всяка една роля, какво трябва да извършват и взаймнодействието между тях. Тези функционални изисквания се допълват от нефункционалните изисквания, които по някакъв начин налагат ограничения върху разработката и проектирането на системата на абстрактно ниво.

Нефункционалните изисквания определят критерии, които се използва за оценка на цялостната работа на дадена система, а не за конкретна нейна функция. Започваме с описание на атрибутите за качество и обосновка на това как изградената от нас архитектура осигорява адекватна реализация на тези нефункционални изисквания.

**Productivity/Производителност:** Производителността е един от най-важните атрибути в една система, той определя по какъв начин работи системата и с каква скорост тя изпълнява своите задачи. Колкото по добра е производителността толкова по ефективна е работата и удовлетвореността на разработчиците и потребителите на системата. По високата производителност предразполага за изпълнение на повече заявки и функции и забързване на работния процес. Разработването на софтуера по добър и последователен начин с добри практики и стабилн архитектурни шаблони ще доведе до високата производителност. В нашето приложение производителността е постигната с помоща на добра йерархия от класове, с добра комуникация, кохезия, наследственост и добре структурирани връзки, както между приложението и базата данни така и в базата данни.

**Еfficiency/Ефективност:** Ефективността е важна за системата и за това какво количество работа може да се извършва за минимално време. Ефективността е може би един от основните атрибути, които показват колко добър е всъщност един софтуер, защото трябва с минимални изисквания и усилия да се извършва голямо количество работа за кратко време. Ефективността в нашият софтуер ще се постигне благодарение на добрият графичен интерфейс, който ще бъде лесен за разбиране и работа, ще улеснява и ще предлага множество от функции на администраторите и потребителите. Те ще изпълняват своите задължения безпроблемно и бързо и ще подобрят работния процес на даденото място. Диалогът между софтуера и потребителя ще се осъществява гледко и лесно с реализацията на всички тези изисквания софтуерът ще бъде добър и ефективен за използване.

**Security/Сигурност:** Сигурноста е атрибут, който е свързан със степента на защита на данните в софтуера от неправомерен достъп до тях, осигоряване на индефикация на потебителя. В нашата система с use case диаграмите е показано, че има индефикация на потебителите: authorization, identify, проверка. В class диаграмите е показно че потребителя винаги влиза в неговият акаунт винаги чрез потребителксо име и парола, възможно е да се добави още една защита с цел по висока сигурност като това е напълно реализируемо, защото системата предразболага за мащабируемост и добавяне на нови функции. В нашата система id номерите също се грижат за сигурността като се следи кой, кога и как прави своите покупки, плащания и всякакви действия във вид на транзакции и проследяване на историята. По този начин има списък от действия на направеното от различните служители или администратори чрез техните id номера. Администратора има права да разглежда тези действия и при нужда дадено действие може да се проследи и провери с цел сигурност. Всичко това ще помогне за по добра сигурност на софтуера.

**Reliability/Надежност:** Надежността е важна, защото тя отговаря за безпроблемната работа на софтуера за определен период от време в определена среда. Тя е много важен фактор, за да може нашите потребители да са удовлетворени и да използват приложението по най-удобният за тях начин без възникване на грешки. Надеждността в нашият софтуер ще бъде постигнат благодарение на точната и ясна работа която трябва да извършва администратора или служителя. Ще бъдат предотвратени потребителските грешки, които могат да се доспуснат и поради невъзможността от допускане на на тези грешки системата ще работи без проблем по всяко едно време. При правилно реализиране на софтуера с добри практики и добро разпределение, той ще бъде постояннен в работата си с минимална възможност от прекратяване или спиране. В системата средното време за отказ ще е малко, средното време за ремонт също ще е малко тъй като приложението ще може бързо, лесно и безпроблемно да се ремонтира с такава лесна за разбиране архитектура и средното време между отказите тогава когато системата ще работи без проблем ще е дълго. Така ще се постигне надеждността в системата.

**Usability/Използваемост:** Използваемостта е възможността софтуера да се използва от потребителя лесно, бързо, удобно и ефективно. Колкото по добър и ориентирам към потребителя е графичният дизайн на софтуера толкова по лесно и удобно ще се използва той. Използваемостта е важен фактор и за ресурсите, които ще се отделят във връзка със софтуера като време и обучние на служителите за правилната работа с него. В нашата система сме разработили изцяло ориентиран към потребителя интерфейс, бърз и изпълнен с множество функции улесняващи работата на служителя. Разработили сме няколко модула според човека използващ системата(администратор или служител), за да може да се разпределя удобно работата и всеки да упражнява само своите задължения. Разработили сме и модул за влизане улесняващ потребителите и подсигуряващ сигурността и самостоятелната им работа.

**Testability/Възможност за проверка:** Този атрибут е основната отговорност за добрият софтуее, той трябва да може да бъде тестван преди пускането му на пазара и по време на работния процес. Добрата реализация на софтуера играе и роля намирането на проблеми ако има такива, колкото по лесно и просто е дефинирана и проектирана толкова по лесно ще се намери проблема. В нашата система както и много други имаме възможност на задълбочено тестване преди пускането официално на системата. След кат отя бъде стартирана в даден обект, вече адмнистратора ще отговаря за правилната работа и подържането на системата от гледан точка на действия на служителите и правилното изпълнение на функциите. Системата напълно предразполага за тези действия и предотвратяването на ненужни проблеми и спирания.

**Extensibility/Разширяемост:** Разширяемостта е атрибута, който определя дали дадената система има възможност за добавяне на допълнителни участници, функции и проложения, дали системата има бъдеще и дали може да се обогатява. Разширяемостта е важно качество на един софтуер, тъй като изискванията към системата непрекъснато растат, съсздават се множество нови софтуери и приложения, технологиите и информационните системи се развиват скоростно, за това софтуера трябва да бъде в крак с тенденциите, да предлага актуални функции и възможности и дори да предоставя нови на потребителите си. В нашето приложение сме оставили възможност за добавяне на много нови роли(участници), които да взаймодействат със системата по различни начини. Софтуерът предлага възможност за добавянето и разширяването както на своите функции така и на функциите на участниците в системата. Самата идея за този софтуер предразполага за по нататъчно разширение с онлайн системи, cloud системи, връзка между различни сектори, кореспонденция и комуникация и иползване на една обща голяма система. По този начин софтуерът може много да се разрасне и разшири.