Архитектурен проект засофтуер, подпомагащ успешното управление на различни видове заведения

Изготвили:

Проектна група 15

Анита Валентинова Караиванова 471218056

Бончо Георгиев Бонев 471218053

Георги Донков Донков 471218026

Йоана Станиславова Ангелова 471218016

Дата: 27.10.2020г.

**Съдържание**

**1. Въведение**

**2. Предназначение**

**2.1. Обхват**

**3. Архитектурен обзор**

**3.1. Use-case изглед**

**3.2. Логически изглед**

**3.3. Процесен изглед**

**3.4. Изглед на данните**

**4. Нефункционални изисквания**

**4.1. Достъпност**

**4.2. Разширяемост**

**4.3. Производителност**

**4.4. Сигурност**

**4.5. Възможност за тестване**

**4.6. Интероперабилност**

**4.7. Използваемост**

**1. Въведение**

   Даденият документ представя архитектурата на софтуера, изготвян от проектна група 15. Направен е архитектурен обзор, който включва основни диаграми, акцентиращи и детайлизиращи различни видове дейности, които ще се извършват чрез нашия дофтуер. Диаграмите, които са използвани в документа, са Use-case UML диаграма, Sequence UML диаграма, State UML диаграма, Class UML диаграма и Entity-Relationship UML диаграма. Освен архитектурен обзор, са представени и нефункционалните изисквания, които ще трябва да бъдат спазени при изграждането на нашия софтуер, за да постигнем нужните цели. Акцентът е върху достъпност, разширяемост, производителност, сигурност, възможност за тестване, интероперабилност и използваемост.

**2. Предназначение**

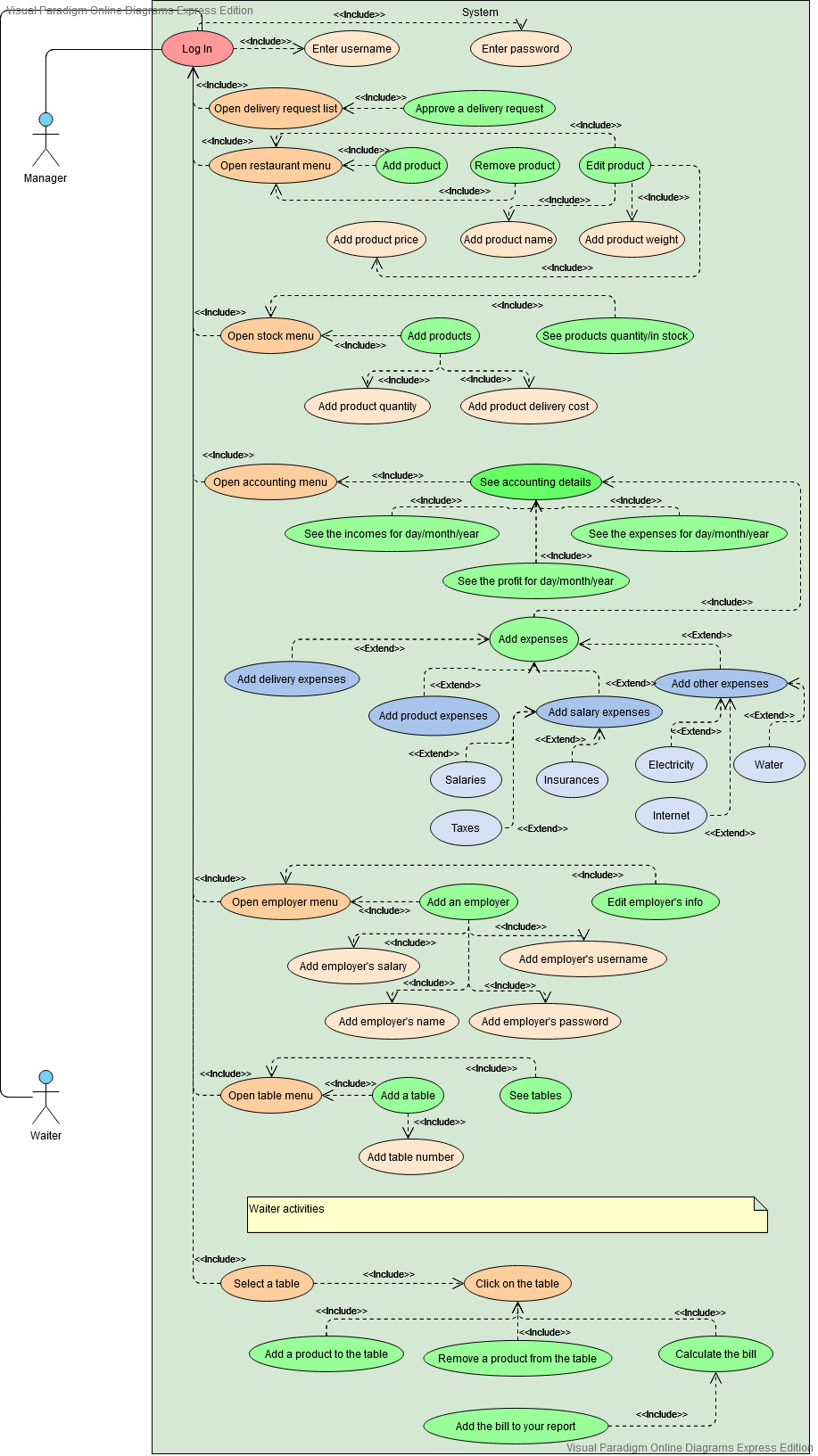
**2.1. Обхват**

Настоящият документ обхваща етапа на разработване на софтуерната архитектура и описва подробно нейните спецификации, функционалности и изисквания.

**3. Архитектурен обзор**

**3.1.       Use-case изглед** – представя функционалните възможност на софтуера (основни сценарии) от гледна точка на ползвателите (актьори).

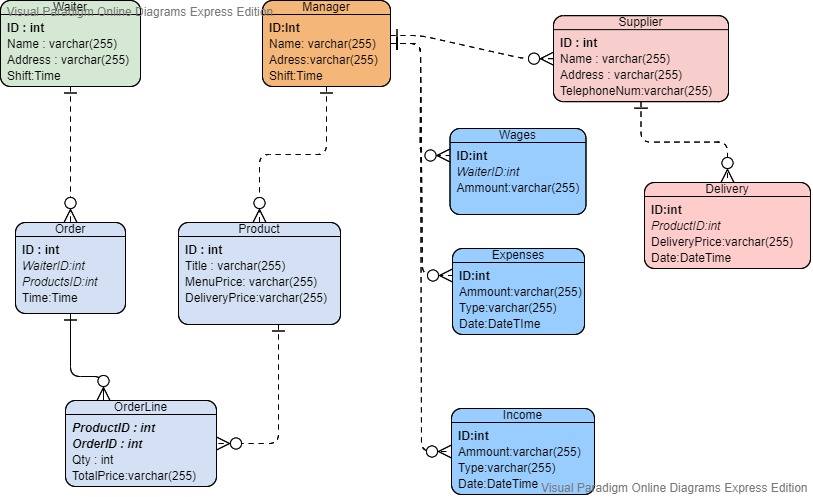
*(Георги Донков)*



Дадената Use-case диаграма показва основните дейности, който ще могат да се извършват от мениджъра на заведението и сервитьорите. Дейностите в софтуера ще могат да се извършват след като всеки от актьорите влезе в системата със своите име и парола. Мениджърът ще може да отваря списъка с поръчки и съответно да одобрява такива(ако има чакащи); да отваря менюто на ресторанта, като вътре в него ще може да добавя продукти, добавяйки техните характеристики, които са описани в редактирането на продуктите, да премахва продукти, да редактира продукти, редактирайки цената, името и теглото на съответния продукт; да отваря менюто за складовата наличност, където ще може да вижда кои продукти са налични и каква е тяхната бройка и да добавя различни продукти в наличност след извършена от доставчика доставка, заедно с тяхната бройка; да отваря счетоводното меню, където ще може да вижда счетоводни детайли, а именно приходите за ден/месец/година, разходите за ден/месец/година, печалбата за ден/месец година, ще има възможността да добавя разходите на заведението, които ще са доставните разходи на всеки продукт, разходите за доставка на всеки продукт, разходите за заплати, включващи заплатите, данъците и осигуровките, разходите за ток, вода и интернет; да отваря менюто за служители, в което ще има възможността да добавя служители, добавяйки тяхната заплата, име, име за влизане в системата(username) и парола (добавяйки служител в менюто за служители всъщност мениджърът ще създава акаунт на добавения служител, чиято информация ще се добавя в базата от данни); да отваря менюто за маси, в което ще може да добавя различни маси, добавяйки техния номер, и да вижда добавените до момента маси. Сервитьорите на заведението ще могат да селектират различните маси, добавени от мениджъра, като натискат върху тях, и съответно да добавят продукти към селектираната маса, да премахват продукти от селектираната маса, да изкарват сметката на селектираната маса(изчисляването на сметката ще се извършва от софтуера) и да добавят сметката към отчета си.

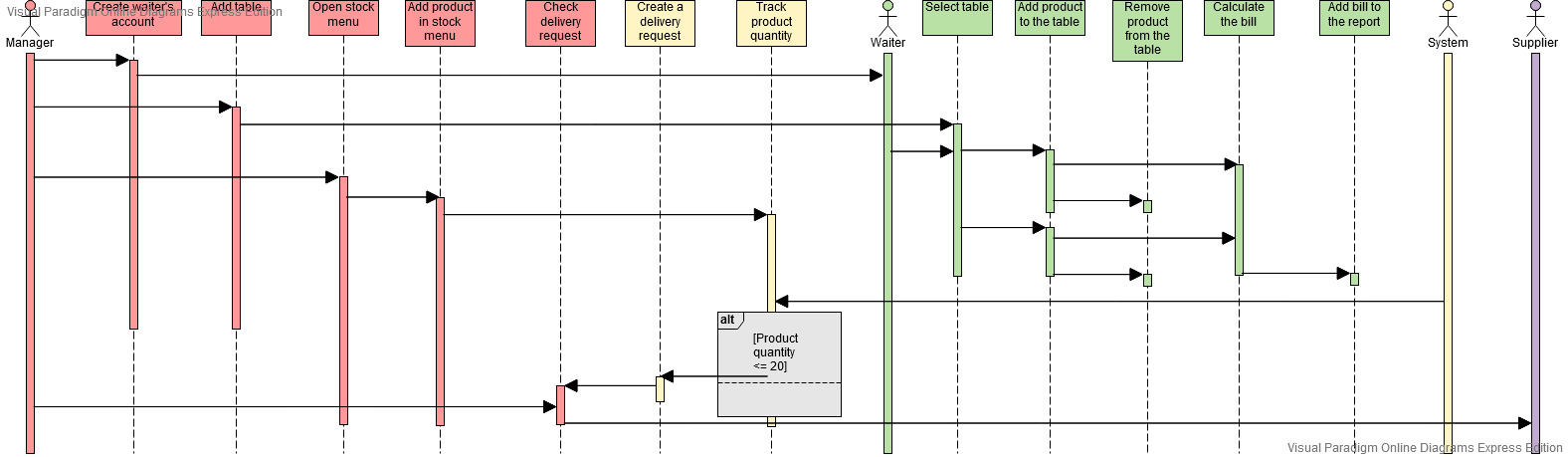
**3.2.       Логически изглед** – софтуерът се представя като композиция от модули, пакети и класове.

*(Бончо Бонев)*

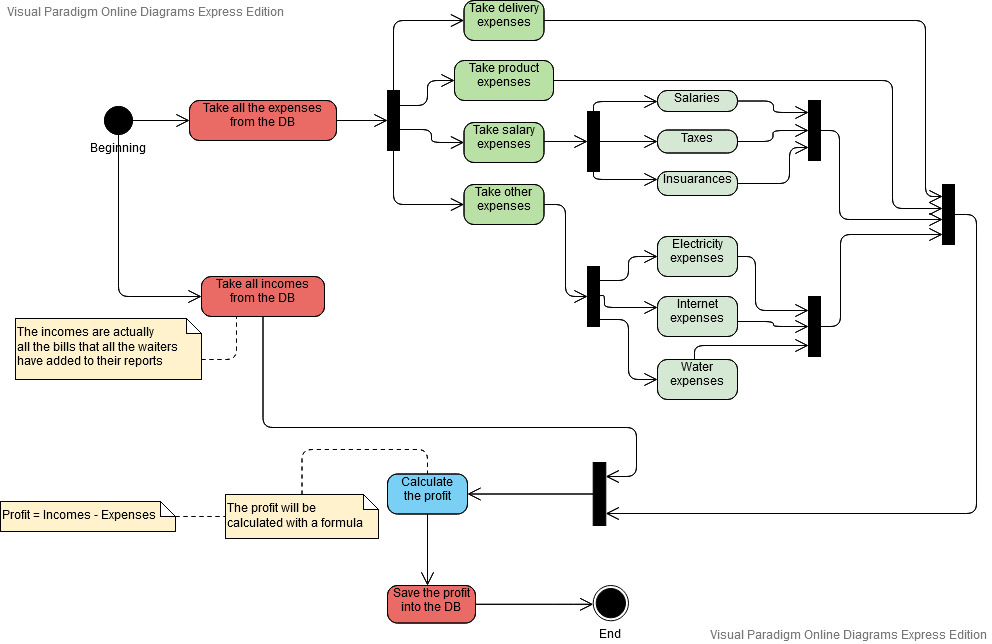
**

**3.3.       Процесен изглед** – представя алгоритмите, реализиращи архитектурно значимите функции.

*(Георги Донков)*



Дадената Sequence диаграма акцентира и детайлизира няколко основни дейности. Първата дейност е създаването на сервитьорски акаунт от мениджъра. Сервитьорът може да извършва дейностите си чак след като му бъде създаден съответният акаунт. Втората дейност е създаването на маса от мениджъра. След като бъде създадена маса, сервитьорите ще мофат да добавят множество продукти към нея, да махат продукти от нея, да изкарват сметката и да добавят сметката към отчета си. Третата дейност е отварянето на менюто за складовата наличност, в което ще може да добави продукт, който вече е наличен в склада, заедно с неговата бройка, която ще се следи от софтуера. Ако бройката на даден продукт стане равна на двадедесет или по малка от двадесет, то софтуерът автоматично ще изпраща заявка за доставка към мениджъра, която заявка ще съдържа информация за това, че количеството на обектният продукт е достигнало съответната бройка. Мениджърът от своя страна ще може да преглежда всички заявки за доставки и да ги одобрява, като при одобряването заявката автоматично се препраща към доставчика на продукта, за да може той да го изпрати към склада. Одобряването на заявка от мениджъра е с превантивна цел в случаите в които даден продукт е на изчерпване, но мениджърът няма да желае той да се достави на момента в склада.



Дадената State диаграма акцентира и детайлизира алгоритъма, чрез който софтуерът ще пресмята счетоводството на заведенията. Първоначално софтуерът ще взима от базата данни всички разходи, добавени до момента от мениджъра, които ще са:

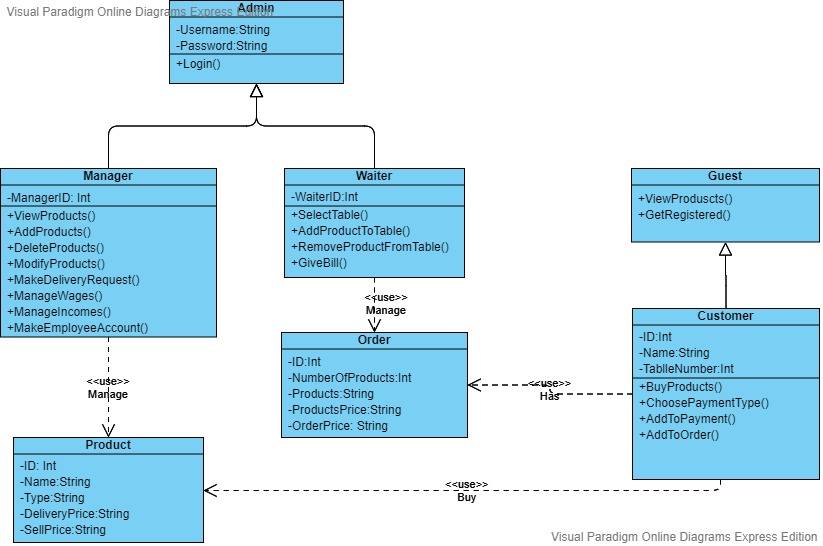
* Разходи за доставка
* Доставни разходи на всеки продукт
* Разходи за заплати
  + Разходи за заплатите на съответните служители
  + Разходи за данъци
  + Разходи за осигуровки
* Други разходи
  + Разходи за електричество
  + Разходи за интернет
  + Разходи за вода

След това софтуерът ще взима всички приходи от базата данни. Приходите всъщност са стойностите на всяка сметка, която всеки сервтиьор е добавил към своя отчет. След като софтуерът е взел приходите и разходите от базата данни, то той ще може да пресмята чистата печалба по формулата:

След изчисляването на печалбата софтуерът ще добавя стойността й в базата данни.

**3.4.       Изглед на данните** – представя модел на обработваните данни, включващ логическите връзки между тях.

*(Бончо Бонев)*

**

Тази диаграма илюстрира основните класове в системата. Има един основен клас Admin, който има потребителско име и парола, чрез който влиза в системата. Наследява се от два класа, Manager и Waiter. Класът Manager може да преразглежда продуктите, да добавя нови,да ги променя или да ги трие от системата. Другата важна операция, която прави е поръчките за нови продукти и да създава акаунти на служителите. Waiter класът има за цел обслужването на клиентите. Той избира маса и изпълнява поръчките на масите, който е избрал, като накрая носи сметката. Класът Product е с атрибути за име, тип ,номер и цена за поръчка и продажба. Главната цел на класа Manager е да управлява продуктите в склада и тези в менюто. В Order се представят номера поръчката, общата сума на продуктите, който са поръчани, техният брой и цена.

**4. Нефункционални изисквания** *(Йоана Ангелова)*

**4.1. Достъпност**

За да се осигури лесно диагностициране и отстраняване на проблеми се предвижда софтуерният продукт да бъде разделен на модули – модул, който отговаря за текущите сметки, модул за отчетите, модул, съдържащ менюта и модул, отговарящ за доставките. От своя страна това осигурява лесно и бързо откриване на проблема и неговото отстраняване без опасност да се наруши работата на друг модул.

**4.2. Разширяемост**

Тъй като е възможно в бъдеще да е нужно системата да изпълнява и допълнителни функционалности в зависимост от нуждите на заведението, тя трябва да е така проектирана, че да предостави възможност те да бъдат добавени без особено затруднение от страна на разработчиците. Предвидено е тя да бъде разделена на относително автономни модули, които обменят данни помежду си, като по този начин добавянето на нов модул в бъдеще не би представлявал трудност.

**4.3. Производителност**

За да се постигне нужната производителност е необходимо на първо място да се осигури нужният хардуер – лаптоп или персонален компютър с процесор с тактова честота 3.2 GHz и 4 GB оперативна памет биха били достатъчни. Основен приоритет е заявките към софтуера да се изпълняват за 1-2 секунди. За да се избегне претоварване на компютърната система и евентуална загуба на данните при използване на предложения софтуерен продукт, се препоръчва той да не се използва паралелно с други натоварващи системата компютърни програми. Нужно е и да се осигури постоянна високоскоростна интернет връзка.

**4.4. Сигурност**

Сигурността на данните зависи както от потребителите, така и от самата система. В системата е заложена йерархична структура на нивата на достъп, която кореспондира с длъжността на потребителите, съответно с най-високо ниво на достъп са мениджърите, а с по-ниско бармани и сервитьори. По този начин се ограничава достъпа на неоторизирани служители до дадена функционалност на системата. Съответните нива на достъп се получават при регистрация на потребителя (със съответни име и парола) в системата и се ползват при последващ вход. Добра практика от страна на потребителите е да използват парола, състояща се от много на брой и различни символи. Системата от своя страна може да повиши нивото на сигурност като осигури хеширане на данните при запазването им в базата данни.

**4.5. Възможност за тестване**

По време на програмирането на всеки модул е предвидено всяка една негова функционалност да бъде тествана. След разработването на всички модули и тяхното е обединяване се тества цялостното поведение на обединения софтуерен продукт, като това включва и връзката с базата данни.

**4.6. Интероперабилност**

Предвидено е отделните модули на системата да обменят данни помежду си, като съответният тип на данните е предварително определен в зависимост от заявките. Например от модула, отговарящ за текущите поръчки, данните (сума на поръчката) да бъдат прехвърлени към модула, отговарящ за отчетите (натрупване на поръчките от деня в дневния отчет). Освен вътрешна комуникация между компонентите си, системата обменя данни с базата данни, като отново те са синхронизирани по тип.

**4.7. Използваемост**

Основно изискване към системата е тя да бъде лесна за използване от потребителите. За да се постигне това е нужно създаденият дизайн да консистентен, тоест различните компоненти на системата да следват един и същ модел. Също така е важно дизайнът да е минималистичен – да се избегне излишно детайлизиране като допълнителни бутони, етикети, надписи и т.н., но да се не пропускат всички нужни компоненти. Не на последно място дизайнът трябва да е интуитивен и да не отнема повече от час време за обучение и добре изглеждащ визуално, за да бъде приятен за използване от потребителя.