Софтуер за складова организация

Проектна група: 10

Автори:

Марио Мариов Георгиев 471218002

Джем Шюкрю Мустафа 471218006

Боян Константинов Миланов 471218033

Теодор Антонов Любенов 471218070

Пламен Дамянов Нончев 471218051

**Въведение**

Читателите на този документ могат по-подробно да получат информация отностно складовият софтуер и по точно :

-Подробно описание къде и в какви ситуации той ще бъде полезен

-Упътване за тънкостие при работата със складовият софтуер

-Източници от които са използвани идеи

-Архитекрура на софтуера(функционалост, зависими компоненти и други)

**Участници в проекта:**

* Разработчици
* Project manager
* QA
* HR
* Фирма поръчваща софтуера

**Предназначение**

Обхват

В архитекруният проект можем по-подробно да се запознаем със складовия софтуер. В него можем да намерим диаграми, които изобразяват графично структурата и информация за работата на продукта. Тук още може да се намери информация за достъпност на програмата. Всеки служител може да намери какво ниво на достъп има. При проблем със софтуера, администраторите по-лесно ще могат да търсят и съответно да решват дадените казуси.

Актьори:

* Служители
* Администратори
* Клиенти

Използвани термини и символи:

* **QA** **–** Quality Assurance
* **HR** **–** Human resources
* **Project manager –** Ръководител на проекта

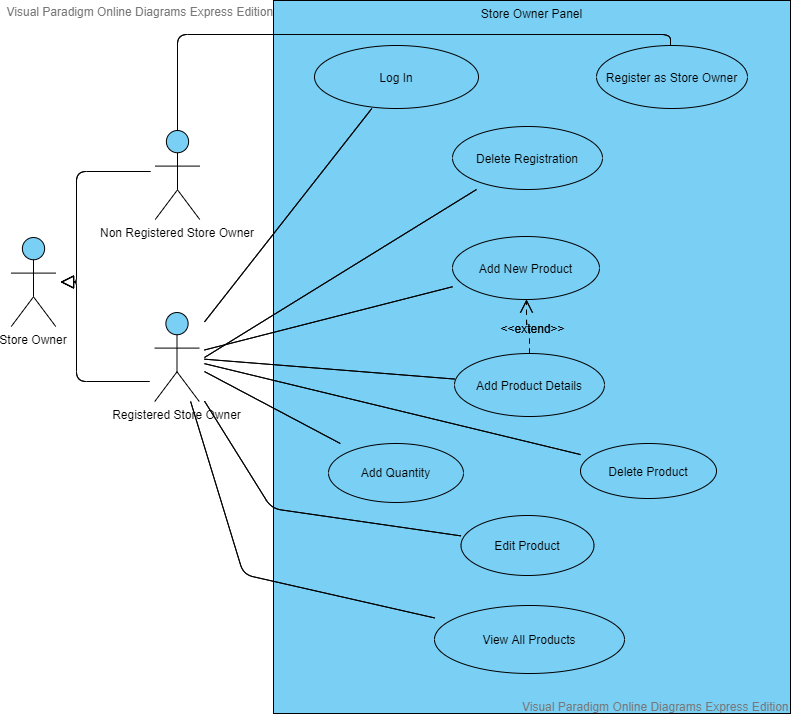
Източници:

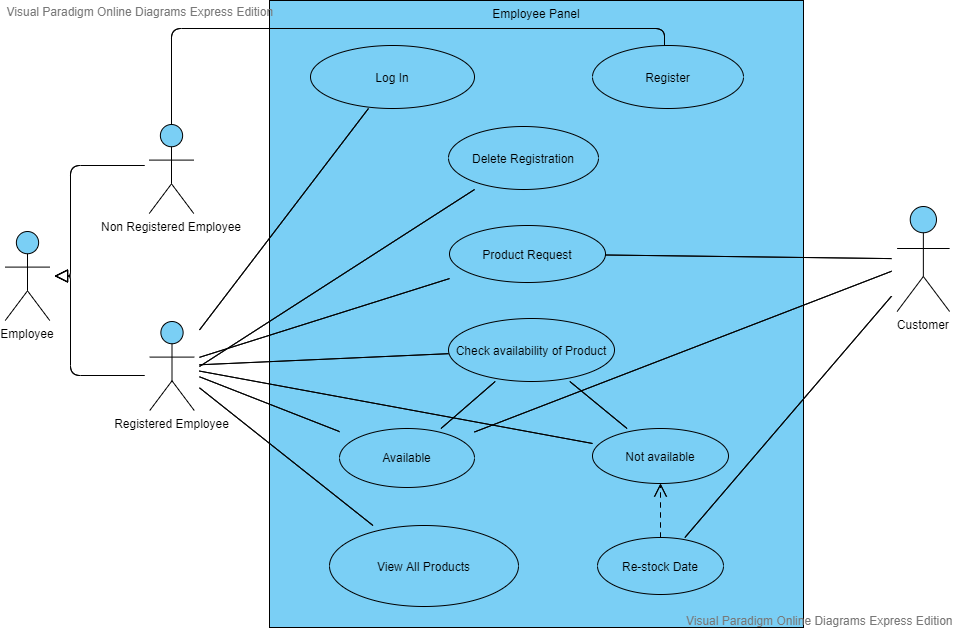
<https://www.guru99.com/mvc-tutorial.html>

https://www.quora.com/What-are-the-pros-and-cons-of-the-MVC-pattern

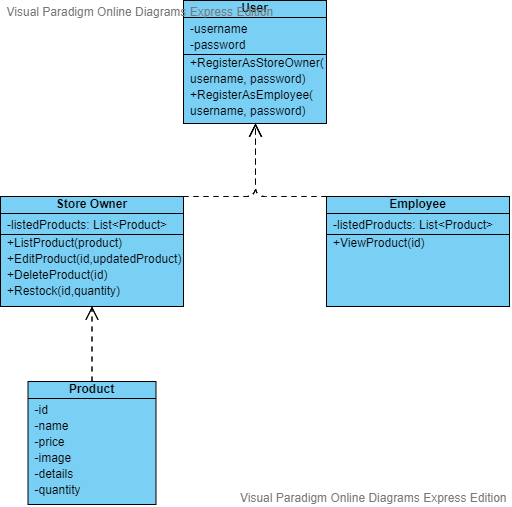
**Архитектурен обзор**

Use-case изглед

На диаграмата са демонстрирани основните задачи, които системата трябва да изпълнява за ролята “Собственик на магазин”, а именно - вписване, регистрация, изтриване на регистрация, добавяне на продукт - добавяне на детайли за продукта, редакция на съществуващ продукт, добавяне на допълнително количество/бройки на даден продукт,изтриване на продукт и достъп до списък с всички продукти



На диаграмата са демонстрирани основните задачи, които системата трябва да изпълнява за ролята “Служител”, а именно - вписване, регистрация и проверка за наличност на даден продукт.



Описание на логическия изглед на архитектурата. Включени са клас диаграми, за да се илюстрират връзките между архитектурни значимите класове. Има един базов клас User, който ще бъде наследяван от класовете Store Owner и Employee. User има съответните атрибути и методи, които ще бъдат имплементирани в класовете наследници. В наследниците са съотвените допълнителни методи, които осигуряват различни права за различните потребители. Друг важен клас е Product, който се свързва с клас Store Owner чрез метода ListProduct. Връзката между тях е правота на собственика да добавя и модифицира продуктите в склада и магазина.

**Нефункционални изисквания**

Model-View-Controller (MVC) архитектура е подходяща, както за уеб приложения така и за desktop приложения. Цената, която се плаща за имплементирането на тази архитектура е сложността ѝ, но от друга страна MVC архитектурата осигурява добра реализация за продукта и изпълнение на важни атрибути за качество. Важен за конкретната система, софтуер за складова организация, атрибут е времето за реакция (responsiveness). Други ключови атрибути за качество са използваемост (usability), мащабируемост (scalability), надеждност (reliability) и др.

* **Време за реакция (responsiveness) –** Един от най-важните атрибути за качество в системата. Тъй като функциалността на софтуера се изразява в това да обработва заявки и да извежда информация, която трябва да бъде предоставена на клиента в рамките на секунди, изпълнението на този атрибут е от голямо значение за удовлетвореността, както на клиента, така и на служителя, който ползва системата. Свързаността на трите компонента на MVC архитектурата помага за по-лесна комуникация между модулите и намаляване на времето на реакция. За удобство, системата трябва да връща резултат до максимум 3 секунди. Комуникация между model, view и controller спомага затова, защото и 3те са свързани едно с друго по триъгълната форма.
* **Използваемост (usability) –** Друг важен атрибут за качество, който спомага за удовлетвореността на служителя за работа със софтуера. Този атрибут ще спести разходи за обучение на персонала за работа със системата. Ще спести време, както на целия колектив, така и на отделните служители, на които им се налага да работят с него. MVC архитектурата е подходяща за доброто реализиране на използваемостта, защото е отделен цял модул за интерфейса предложен на потребителя. В случая View частта е основната част, която ще даде предпоставки за лесно, удобно и приятно използване на софтуера. При отделение на този модул е лесно да се фокусира вниманието върху конкретни изисквания към потребителския интефейс и в случаи на проблеми те да бъдат коригирани.
* **Мащабируемост (scalability) –** Мащабируемостта осигурява системата да не понижава своята производителност при увеличаване на натоварването в системата. При работа с данни, за което е предназначен софтуерът, е много вероятно натоварването да стане по-голямо. Причина може да бъде увеличаването на потребителите работещи със системата или пък да се уголеми обемът на данните. В случая вторият вариант е по-вероятен. Добавяне на нови стоки, нова информация за стоките или пък дори отваряне на нови клонове магазини или складове може да допринесе за увеличаването на натоварването. При MVC архитектурата е лесно да се внедрят нови модули от същия тип, като по този начин се прави хоризонтална мащабируемост (scale out). Друг възможен метод е вертикална мащабируемост (scale up), която се осигурява чрез добавяне на ресурси към съществуващи сървъри. Разделението на модулите в MVC прави добавянето на нови такива или ресурси по-лесно и по този начин се осъществява по-лесно мащабируемостта.
* **Надеждност (reliability) –** Надеждността е важен фактор за добрата работа на системата, защото тя е предназначена за отговор на заявките на клиентите. При работа с клиенти е много важно да може да им се отговори бързо и точно. За да можем да определим надеждността на системата трябва да се направят различни тестове, което води до следващия важен атрибут за качество:
* **Възможност за проверка (testability) –** Този атрибут е свързан с надеждността, която е от значение за работата на системата. MVC моделът е подходящ за изпълнение на възможността за проверка, защото компонентите могат да бъдат тествани отделни от потребителя като по този начин не се причинява неудобство за ползвателите на системата и не забавя процесите във фирмата.
* **Възможност за поддръжка (maintainability) –** Когато един софтуерен продукт е завършен следва най-важната част от живота му – поддръжката. Този атрибут е от голямо значение за успеха на системата, защото от него зависи отстраняването на грешки без да се създава неудобство на потребителя, както и добавянето на нови функции. Благодарение на модулната структура на MVC архитектурата възможността за поддръжка е лесно осъществима. Освен поддръжката, голямата гъвкавост помага и за прилагане на бъдещи подобрения.