**SuperDuperMarktApp**

Projekt-Dokumentation

Bearbeitet von: Christopher Ridder

Letzte Aktualisierung: 05.12.23

Inhalt

[Anforderungen. 3](#_Toc152609826)

[Annahmen 3](#_Toc152609827)

# Anforderungen

* Die App soll Regalpflegenden die Erfassung von eingeräumten Produkten ermöglichen, sodass jederzeit der Regalbestand eingesehen werden kann.
* Die App soll Regalpflegende dabei unterstützen, die Produkte mit den korrekten Preisen auszuzeichnen, indem produktspezifische und tagesaktuelle Preise berechnet und abgerufen werden können.
* Die App soll Regalpflegende dabei unterstützen, Produkte bei Erreichen ihres Verfallsdatums aus den Regalen zu nehmen, sodass sichergestellt ist, dass keine abgelaufene Ware in den Regalen liegt.
* Die App soll Regalpflegende dabei unterstützen, Produkte bei Unterschreitung eines gegebenen produktspezifischen Qualitätsniveaus aus den Regalen zu nehmen, sodass sichergestellt ist, dass nur Produkte der vorgegebenen Qualität in den Regalen liegen.
* Die App soll Regalpflegenden auf Abruf eine Ansicht bieten, die zu allen Produkten neben dem Namen auch tagesaktuelle Angaben zum Bestand, der Qualität, dem Preis und die Haltbarkeit enthält, wobei zu entsorgende Produkte speziell markiert werden.
* Die App soll die Aufnahme weiterer Produkte ermöglichen, ohne dass dabei der ‚zentrale Steuerungscode‘ geändert werden muss, sodass eine Aufnahme im laufenden Betrieb möglich ist.
* Die App soll das Einlesen von Produkten via csv ermöglichen, sodass eine beliebige Menge an Produkten auf einmal erfasst werden kann.
* Die App soll das Einlesen von Produkten via SQL ermöglichen, sodass eine beliebige Menge an Produkten auf einmal erfasst werden kann.

# Annahmen

* Es existiert keine Lagerhaltung. Produkte werden direkt in die Regale geräumt. Konsequenz: keine Unterscheidung verschiedener Bestände, also kein Lagerbestand, Regalbestand oder Gesamtbestand als Summe, sondern nur den einen Bestand des Sortiments.
* Im Betrachtungszeitraum finden Lieferungen von Produkten statt. Konsequenz: der Bestand kann nicht nur kleiner werden, sondern auch wieder anwachsen.

# Verwendete Pattern

* **Factory Method** [eigene Implementierung]**:** Verschiedene Produkte können mit einer zentralen Erzeuger-Methode generiert werden. Diese wird von speziellen Erzeuger-Klassen überschrieben, die jeweils den spezifischen Produkttyp zurückgeben. Dies ermöglicht das flexible Hinzufügen weiterer Produkt-Typen ohne Änderungen am bestehenden Code.
* **Singleton** [Spring-Framework]**:** Erzeugung einer zentralen Instanz einer Klasse, die an verschiedenen Stellen wiederverwendet wird. Keine eigene Implementierung des Patterns, sondern Nutzung der Spring-Implementierung, also Springs IoC-Container und Dependency Injection z.B. via Autowring oder Constructor Injection von Beans.
* **Builder** [Lombok @Builder Annotation]**:** Die Konstruktion eines Objektswird an einen Builder übergeben. Mit Hilfe des Builders kann das Object sukzessive konstruiert werden. Es ist möglich nur die Eigenschaften zu definieren, die erforderlich sind. Eine Definition von verschiedenen Constructors mit unterschiedlichen Parametern entfällt bei Verwendung des Builder patterns. Nutzung des Patterns über Lomboks @Builder-Annotation.