

**МИНОБРНАУКИ РОССИИ**  
**Федеральное государственное бюджетное образовательное**  
**учреждение высшего образования**  
**«Ижевский государственный технический университет имени М.Т. Калашникова»**  
**(ФГБОУ ВО «ИжГТУ имени М.Т. Калашникова»)**

Факультет Информационные технологии  
Кафедра Программное обеспечение  
Направление подготовки 09.03.04 Программная инженерия

**ОТЧЕТ ПО ПРАКТИКЕ**

Производственная практика. Преддипломная практика

---

Выполнил  
студент гр. Б21-191-13

И.А.Шнейдер

Дата сдачи отчета: «  » 2025г.

Дата аттестации : «  » 2025г.

Оценка \_\_\_\_\_

Руководитель практики  
от ФГБОУ ВО  
«ИЖГТУ имени М.Т. Калашникова» \_\_\_\_\_ / К.С. Чернышев  
старший преподаватель кафедры ПО

Заведующий кафедрой \_\_\_\_\_ /М.В. Леонов

д.э.н., зав. кафедрой ПО

### **3. РЕАЛИЗАЦИЯ ПРЕЦЕДЕНТОВ ИНФОРМАЦИОННОЙ СИСТЕМЫ УПРАВЛЕНИЯ МАСТЕРСКОЙ АВТОРСКИХ ИЗДЕЛИЙ**

В данном разделе описываются ключевые прецеденты, реализованные в интернет-магазине рукодельных изделий, включая управление учётной записью, поиск и фильтрацию товаров, управление корзиной, оформление и оплату заказа, а также специализированные функции для hand-made: конструктор индивидуальных заказов, двухуровневый учёт материалов и детализированное отслеживание этапов заказа.

Каждый из прецедентов включает в себя математические постановки, алгоритмы, классы и контрольные примеры, что позволяет чётко понять логику работы системы и её функциональные возможности. Реализация этих прецедентов обеспечивает пользователям интуитивно понятный и безопасный доступ к уникальным товарам ручной работы, а мастерам — эффективный инструмент для управления бизнес-процессами.

#### **3.1. Реализация прецедента «Управление учётной записью»**

Данный прецедент обеспечивает регистрацию, аутентификацию и управление профилем пользователя, включая подтверждение email, хеширование паролей и генерацию токенов для безопасного доступа.

##### **3.1.1. Реализация алгоритма**

Входные данные: email, пароль, роль (покупатель/мастер).

Выходные данные: при успешной регистрации создаётся запись в таблице User, отправляется письмо подтверждения, после подтверждения пользователь может войти в систему. Сессия управляется через Django Sessions.

Алгоритм регистрации пользователя представлен на рис.3.1

## Схема алгоритма регистрации

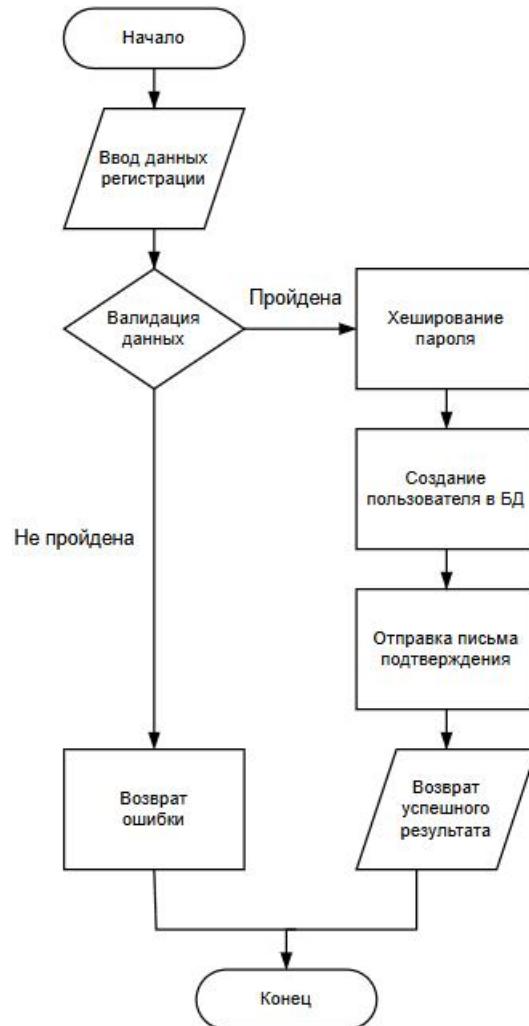


Рис. 3.1

### 3.1.2. Реализация классов

Класс User реализует модель пользователя системы.

Поля класса:

- id: Integer — уникальный идентификатор (Primary Key)
- email: String — электронная почта, уникальная
- password\_hash: String — хешированный пароль
- role: String — роль ('buyer', 'master', 'admin')
- created\_at: DateTime — дата регистрации

Основные методы:

- register(email, password, role) - создание нового пользователя
- authenticate(password) - проверка пароля и аутентификация
- confirm\_email() - подтверждение email адреса

### 3.1.3. Описание контрольного примера

Входные данные:

- email: ingredshneider@yandex.ru
- Пароль: Hilda\_2010
- Роль: buyer

Форма регистрации представлена на рисунке 3.2

Экранная форма регистрации

The screenshot shows a registration form titled "Регистрация". It includes fields for "Email адрес:" (Email address) containing "ingredshneider@yandex.ru", "Роль:" (Role) set to "Покупатель" (Buyer), "Пароль:" (Password) containing "Hilda\_2010", and "Пароль (ещё раз):" (Password again) also containing "Hilda\_2010". A note below the first password field states: "Пароль должен содержать минимум 8 символов и не может состоять только из цифр." (The password must contain at least 8 characters and cannot consist only of digits.) A blue "Зарегистрироваться" (Register) button is at the bottom, and a link "Уже есть аккаунт? Войти" (Already have an account? Log in) is at the bottom right.

рис. 3.2

### 3.2. Реализация прецедента «Поиск и фильтрация товаров»

Данный прецедент позволяет пользователям находить товары через текстовый поиск с использованием TF-IDF, а также применять фильтры по категориям, цене, материалам и другим параметрам.

### 3.2.1. Математическая постановка

Алгоритм TF-IDF для текстового поиска

$$TF(t, d) = f(t, d) / \sum f(t', d) \quad // \text{Частота термина в документе}$$

$$IDF(t) = \log(N / df(t)) \quad // \text{Обратная частота документа}$$

$$TF - IDF(t, d) = TF(t, d) \times IDF(t) \quad // \text{Взвешенная оценка}$$

Расчёт релевантности с весовыми коэффициентами

$$R = 0.6 \times TFIDF(name) + 0.3 \times TFIDF(description) + 0.1 \times TFIDF(category)$$

### 3.2.2. Реализация алгоритма

Входные данные: поисковый запрос (от 3 символов), фильтры (категория, цена, материалы).

Выходные данные: отсортированный список товаров с пагинацией, соответствующих запросу и фильтрам.

Алгоритм поиска товаров представлен на рисунке 3.3.

Схема алгоритма поиска товаров

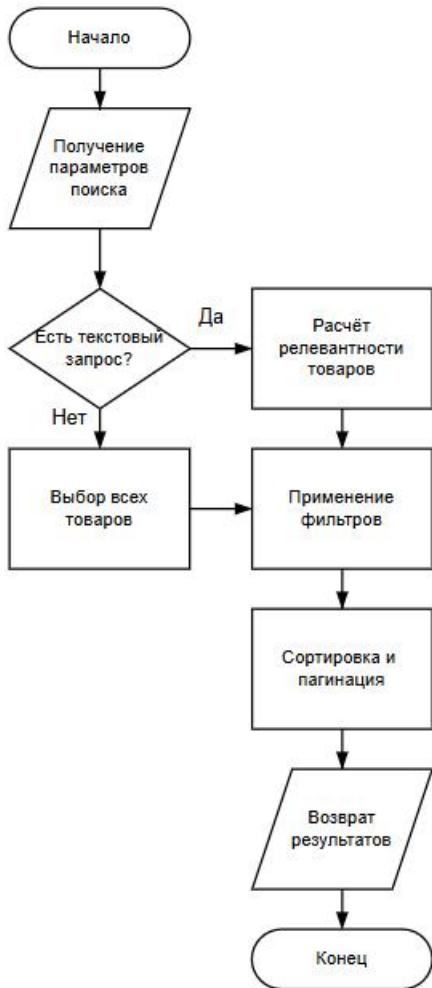


Рис. 3.3

### 3.2.3. Реализация классов

Класс `Product` описывает товар ручной работы.

Поля класса:

- `id: Integer` — идентификатор товара
- `name: String` — название товара
- `description: Text` — подробное описание
- `price: Decimal` — цена товара
- `category_id: Integer` — ссылка на категорию
- `stock_quantity: Integer` — количество на складе

Основные методы:

- `search_by_query(query)` — поиск по текстовому запросу
- `filter_by_category(category_id)` — фильтрация по категории

- filter\_by\_price(min\_price, max\_price) — фильтрация по цене

### 3.2.4. Описание контрольного примера

Входные данные:

1. Поисковый запрос: "вязаная шапка"

2. Фильтры

Категория: "Головные уборы"

Цена: от 1000 до 3000 руб

Материалы: ["шерсть", "пряжа"]

Атрибуты: {"technique": "вязание спицами"}

Поиск товара представлен на рисунке 3.4.

Экранная форма поиска товаров

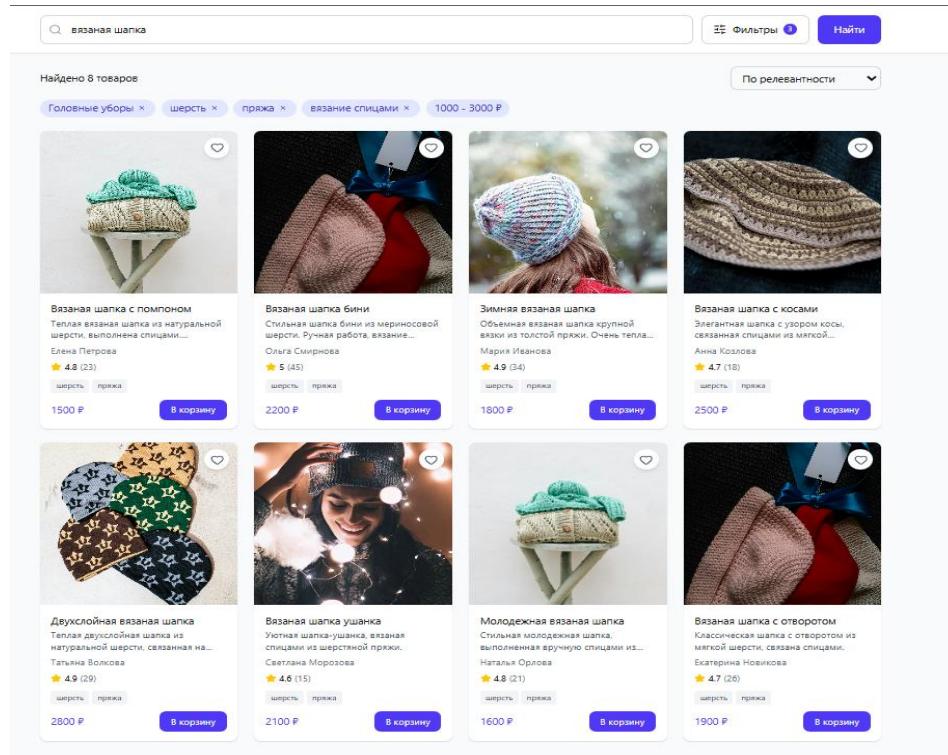


Рис. 3.4

### 3.3. Реализация прецедента «Управление корзиной»

Данный прецедент отвечает за добавление, удаление и изменение количества товаров в корзине, а также за расчёт общей суммы с учётом доступности товаров на складе.

### 3.3.1. Математическая постановка

Расчёт общей суммы корзины

$$total = \sum(item.quantity \times item.product.price)$$

Проверка доступности товара

$$available = (product.stock \geq quantity) \wedge (product.status = 'active')$$

### 3.3.2. Реализация алгоритма

Входные данные: ID пользователя, ID товара, количество.

Выходные данные: обновлённая корзина с пересчитанной суммой или сообщение об ошибке (недостаточно товара, товар неактивен).

Алгоритм добавления в корзину представлен на рис.3.5.

Схема добавления в корзину

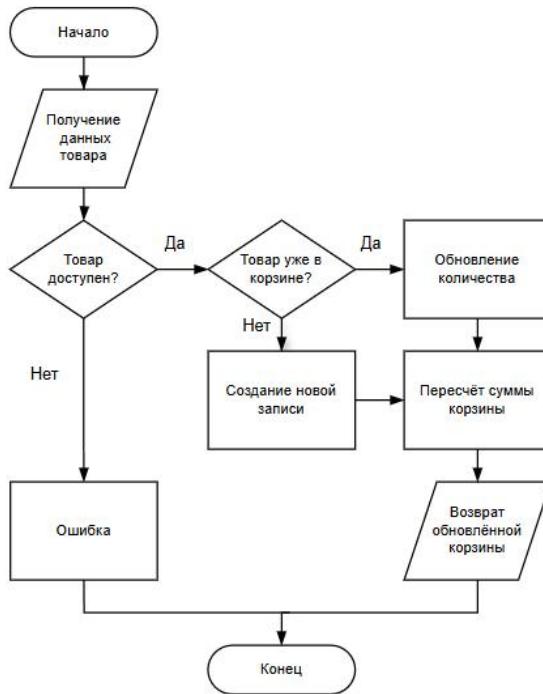


Рис. 3.5

### 3.3.3. Реализация классов

1. Класс Cart представляет корзину покупок пользователя.

Поля класса:

- id: Integer — идентификатор корзины
- user\_id: Integer — ссылка на пользователя
- created\_at: DateTime — дата создания

Основные методы:

add\_item(product\_id, quantity) - добавление товара в корзину  
remove\_item(product\_id) - удаление товара из корзины  
calculate\_total() - расчёт общей суммы корзины  
clear() - очистка корзины

2. Класс CartItem представляет корзину покупок пользователя

Поля класса:

id: Integer — идентификатор позиции  
cart\_id: Integer — ссылка на корзину  
product\_id: Integer — ссылка на товар  
quantity: Integer — количество товара

Основные методы:

update\_quantity(new\_quantity) - изменение количества  
get\_subtotal() - расчёт стоимости позиции

### 3.3.4. Описание контрольного примера

Входные данные

Пользователь: ID 1171

Товар: "Вязаная шапка" (ID 1634, цена: 1500 руб, остаток: 10 шт)

Количество: 2 шт

Работа с корзиной представлена на рис. 3.6.

Экранная форма работы с корзиной

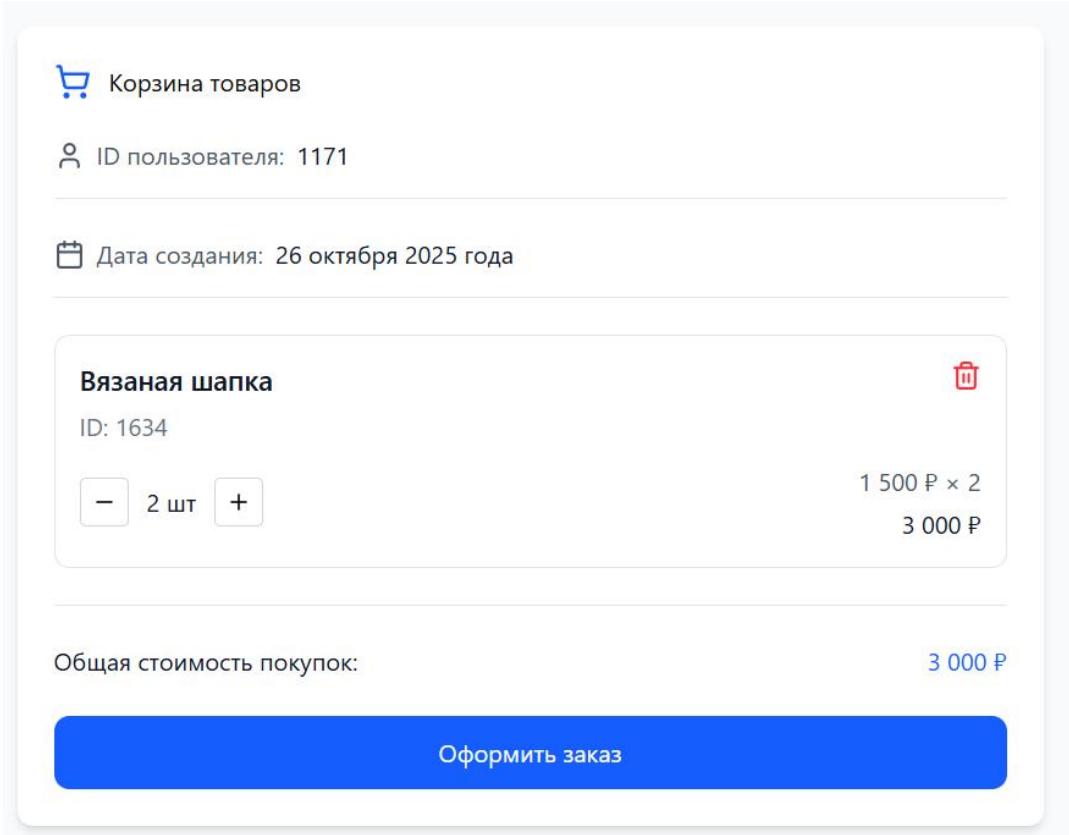


Рис. 3.6

### 3.4. Реализация прецедента «Оформление и оплата заказа»

Данный прецедент включает процесс формирования заказа, расчёт стоимости с учётом скидок и доставки, интеграцию с платёжной системой и изменение статуса заказа после оплаты.

#### 3.4.1. Математическая постановка

Расчёт общей стоимости заказа

$$T = \sum(P_i \times Q_i) + D - C$$

T - общая сумма к оплате, P<sub>i</sub> - цена i-го товара, Q<sub>i</sub> - количество i-го товара, D - стоимость доставки, C - сумма скидки

Расчёт скидки

$$\begin{aligned} C = & \{ 0.1 \times T_{base}, \text{ если } T_{base} \geq 5000 \wedge \text{promo} = 'SUMMER10' \\ & 0.05 \times T_{base}, \text{ если } T_{base} \geq 3000 \\ & 0, \text{ иначе} \} \end{aligned}$$

### 3.4.2. Реализация алгоритма

Входные данные: данные корзины, адрес доставки, промокод.

Выходные данные: созданный заказ со статусом «ожидает оплаты», ссылка для оплаты через ЮKassa.

Алгоритм оформления заказа представлен на рис.3.7.

Схема алгоритма оформления заказа

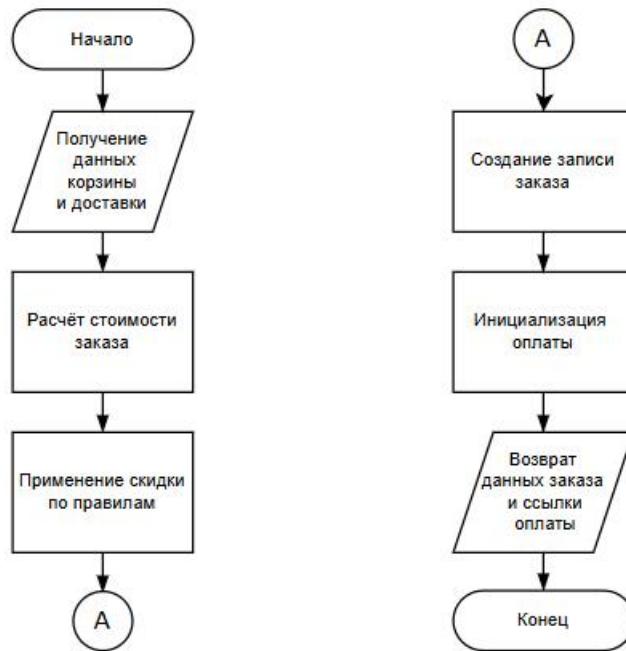


Рис. 3.7

### 3.4.3. Реализация классов

Класс Order представляет заказ покупателя.

Поля класса:

- id: Integer — идентификатор заказа
- user\_id: Integer — ссылка на покупателя
- total\_amount: Decimal — общая сумма заказа
- status: String — статус ('pending', 'paid', 'processing', 'shipped')
- delivery\_address: String — адрес доставки

Основные методы:

- `create_from_cart(cart)` — создание заказа из корзины
- `calculate_total()` — расчёт итоговой стоимости
- `process_payment()` — обработка оплаты через ЮKassa

### 3.4.4. Описание контрольного примера

Входные данные:

- Товары в корзине:
  - а) "Вязаная шапка": 1500 руб × 1 шт
  - б) "Шерстяные носки": 800 руб × 2 шт
- Доставка: 300 руб
- Промокод: SUMMER10

Страница оформления заказа представлена на рис.3.8.

Экранная форма оформления заказа

The screenshot illustrates the order placement process across three windows:

- Step 1: Delivery Address Entry** (Left Window)

  - Form fields include: Region/Oblast (Удмуртская Республика), City (Ижевск), Street (Ленина), House (62), Apartment (58), and Zip Code (426060).
  - A Promocode input field contains "SUMMER10".
  - A "Применить" (Apply) button is visible next to the promocode input.

- Step 2: Order Summary** (Middle Window)

Ваш заказ	
	Вязаная шапка 1 500 ₽ × 1 шт
	Шерстяные носки 800 ₽ × 2 шт
Промежуточный итог: <b>3 100 ₽</b>	
<i>Соцда:</i> -310 ₽	
Доставка: 300 ₽	
Итого: <b>3 090 ₽</b>	

A blue "Перейти к оплате" (Go to payment) button is at the bottom.

- Step 3: Payment Window** (Right Window)

Оплата заказа	
Сумма к оплате: <b>3 090 ₽</b>	
Номер карты *	
1234 5678 9012 3456	
Срок действия *	CVV *
MM/YY	123
Имя держателя карты *	
IVAN IVANOV	
Оплата производится через защищённое соединение с использованием платежной системы ЮKassa	
<b>Оплатить 3 090 ₽</b>	

а) указание адреса доставки и промокода  
при наличии

б) окно с оплатой заказа

Рис. 3.8

### 3.5. Реализация прецедента «Конструктор индивидуальных заказов»

Данный прецедент позволяет покупателю создавать уникальные изделия ручной работы через интерактивный конструктор с автоматическим расчётом стоимости и срока изготовления.

### 3.5.1. Математическая постановка

#### Расчёт стоимости

$$C_{\text{total}} = C_{\text{base}} + \sum(C_{\text{material}}) + \sum(C_{\text{parameter}}) + C_{\text{personalization}}$$

$C_{\text{base}}$  - базовая цена товара,  $C_{\text{material}}$  - наценка за выбранные материалы,  $C_{\text{parameter}}$  - наценка за параметры (размер, цвет),  $C_{\text{personalization}}$  - стоимость персонализации

#### Расчёт срока

$$T_{\text{total}} = T_{\text{base}} + \sum(T_{\text{material}}) + \sum(T_{\text{parameter}}) + T_{\text{personalization}}$$

$T_{\text{base}}$  - базовый срок изготовления в днях,  $T_{\text{material}}$  - дополнительное время на работу с материалом,  $T_{\text{parameter}}$  - время на выполнение параметров,  $T_{\text{personalization}}$  - время на персонализацию

### 3.5.2. Реализация алгоритма

Входные данные: выбор товара-шаблона, настройки материалов, параметров, персонализации.

Выходные данные: расчёт стоимости и срока, резервирование материалов, добавление в корзину.

Алгоритм работы конструктора представлен на рис.3.9.

Схема алгоритма работы конструктора

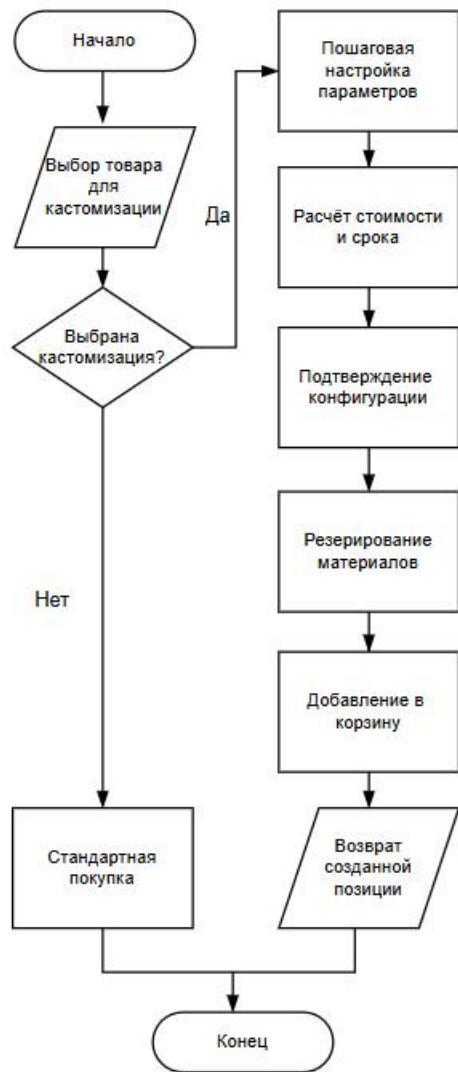


Рис. 3.9

### 3.5.3. Реализация классов

1. Класс `ProductTemplate` представляет шаблон для кастомизации товара.

Поля класса:

- `id: Integer` — идентификатор шаблона
- `product_id : Integer` — ссылка на базовый товар
- `name: String` — название шаблона
- `configuration : JSON` — структура параметров
- `is_active : Boolean` — активен ли шаблон

Основные методы:

- calculate\_price(selections) — расчёт стоимости по выбору
  - validate\_selections(selections) — валидация выбора пользователя
2. Класс CustomOrderSpecification представляет шаблон для индивидуального заказа.

Поля класса:

- id: Integer — идентификатор спецификации
- order\_item\_id: Integer — ссылка на элемент заказа
- user\_id: Integer — ссылка на пользователя
- configuration: JSON — выбор пользователя
- total\_price: Decimal — итоговая цена
- production\_days: Integer — срок изготовления

Основные методы

- get\_materials\_required() — получение необходимых материалов

### 3.5.4. Описание контрольного примера

Входные данные:

Шаблон товара: «Вязаный свитер» (базовая цена — 4000 руб, срок — 5 дней)

Выбор материалов: шерсть мериноса(+1000 руб, +2 дня)

Параметры: размер XXL(+700 руб, +2 дня)

Персонализация: индивидуальный узор (+800 руб, +2 дня)

Пример создания индивидуального заказа представлен на рис.3.10.

Экранная форма оформления заказа

The screenshot displays two overlapping windows. The main window on the left is a 'Summary of the order' (Сводка заказа) for a 'Knitted sweater' (Вязаный свитер). It lists the base price (Базовая цена) as 4000 ₽ and the delivery period (Базовый срок) as 5 days. Below this, it shows a material addition (Материал) for 'Merino wool' (Шерсть мериносова) with an additional cost of +1000 ₽ and a delivery period of +2 days. Further down, it lists a size addition (Размер) for 'XXL' with an additional cost of +700 ₽ and a delivery period of +2 days. Personalization (Персонализация) is shown with an additional cost of +800 ₽ and a delivery period of +2 days. At the bottom, the total (Итого) is 6500 ₽ and the delivery period (Срок изготовления) is 11 days. A purple button at the bottom right says 'Add to cart' (Добавить в корзину).

Подтвердите действие на странице  
e6bd08a7-011f-41ad-be23-2ba0d3cec076-v2-  
figmaframepreview.figma.site

Материалы зарезервированы!  
Товар: Вязаный свитер  
Материал: Акрил  
Количество: 1 шт

OK

a) Сводка создания  
индивидуального заказа

б) Резервирование материалов в  
системе

Рис. 3.10

**3.6. Реализация прецедента «Двухуровневый учёт материалов и остатков»**  
Данный прецедент автоматизирует учёт сырья и его связь с готовыми изделиями, включая резервирование и списание материалов при заказах.

### 3.6.1. Математическая постановка

Расчёт доступного количества материала

$$Q_{\text{available}} = Q_{\text{initial}} - \sum (N_i \times R_i \times (1 + \text{waste\_factor}))$$

$Q_{\text{available}}$  - доступное кол-во каждого материала,  $Q_{\text{initial}}$  - начальный остаток,  $N_i$  - кол-во произведённых/проданных единиц товара  $i$ ,  $R_i$  - норма

расхода материала на единицу товара, `waste_factor` — коэффициент отходов (обычно 0.1)

### 3.6.2. Реализация алгоритма

Входные данные: заказ с товарами, рецепты материалов.

Выходные данные: резервирование материалов, обновление остатков, запись о резервировании.

Алгоритм резервирования материалов представлен на рисунке 3.11

Схема алгоритма резервирования материалов

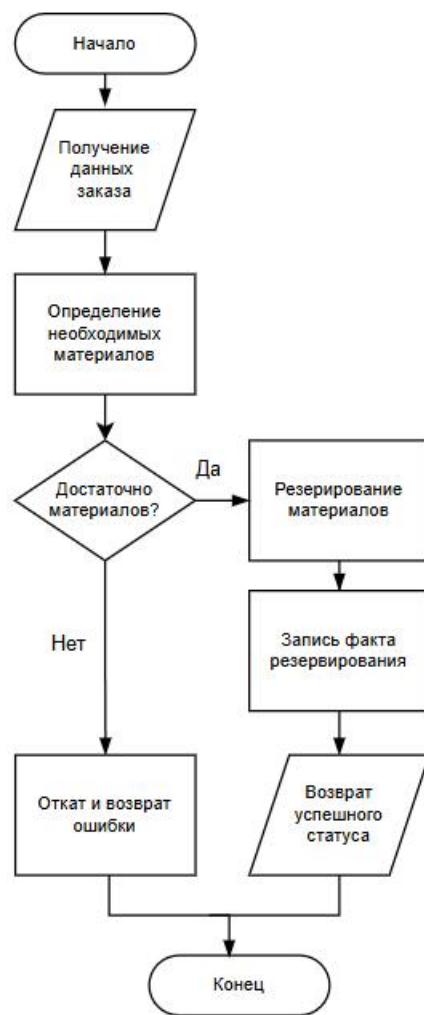


Рис. 3.11

### 3.6.3. Реализация классов

1. Класс `Material` описывает сырьё для производства товаров.

Поля класса:

- id: Integer — идентификатор материала
- name: String — название материала
- current\_quantity: Decimal — текущее количество
- unit: String — единица измерения
- min\_quantity: Decimal — минимальный запас

Основные методы:

- reserve(quantity, order\_id) — резервирование материала
- check\_availability(required\_quantity) — проверка достаточности
- consume(quantity, order\_id) — списание материала

2. Класс MaterialRecipe описывает рецепт расхода материала на товар.

Основные поля

- id: Integer — идентификатор рецепта
- product\_id: Integer — ссылка на товар
- material\_id: Integer — ссылка на материал
- consumption\_rate: Decimal — норма расхода

Основные методы :

- calculate\_required(quantity) — расчёт требуемого количества

3. Класс MaterialReservation фиксирует факт резервирования материала.

Основные поля

- id: Integer — идентификатор
- material\_id: Integer — ссылка на материал
- order\_id: Integer — ссылка на заказ
- quantity: Decimal — количество
- reserved\_at: DateTime — время резервирования

Основные методы:

- consume() — списание
- release() — освобождение резерва

### 3.6.4. Описание контрольного примера

Входные данные:

- Материал: «Шерсть меринаса» (остаток: 50 м, норма расхода: 2 м на свитер)
- Заказ: 3 свитера
- Коэффициент отходов: 0.1

Двухуровневый учёт материалов и остатков представлена на рис.3.12

Экранная форма учёта и резервирования материалов

The screenshot shows a software application window titled "Система учёта материалов". At the top, there are four navigation tabs: "Материалы" (Materials), "Рецепты" (Recipes), "Заказы" (Orders), and "Резервирования" (Reservations). Below the tabs, a sub-menu titled "Остатки материалов" (Material Inventories) is displayed. A table lists three materials with their current status:

Материал	Остаток	Зарезервировано	Доступно	Статус
Шерсть меринаса	43.40 м	6.60 м	36.80 м	В наличии
Хлопковая нить	96.70 м	3.30 м	93.40 м	В наличии
Пуговицы	183.50 шт	16.50 шт	167.00 шт	В наличии

Рис. 3.12

### 3.7. Реализация прецедента « Детализированное отслеживание этапов заказа»

Данный прецедент реализует систему отслеживания статусов заказа через конечный автомат, ведение истории изменений и уведомление покупателей о смене этапов выполнения.

#### 3.7.1. Математическая постановка

Детерминированный конечный автомат (ДКА)

$Q = \{\text{ПРИНЯТ}, \text{СОГЛАСОВАН}, \text{В\_РАБОТЕ}, \text{ГОТОВИТСЯ\_К\_ОТПРАВКЕ}, \text{ОТПРАВЛЕН}, \text{ДОСТАВЛЕН}, \text{ОТМЕНЁН}\}$

$\delta: Q \times \Sigma \rightarrow Q$  // функция переходов

$F = \{\text{ДОСТАВЛЕН}, \text{ОТМЕНЁН}\}$  // конечные состояния

### 3.7.2. Реализация алгоритма

Входные данные: заказ, новый статус, комментарий, фотоотчёт.

Выходные данные: обновление статуса заказа, запись в историю, отправка уведомления.

Алгоритм изменения статуса заказа представлен на рисунке 3.13.

Схема алгоритма резервирования материалов

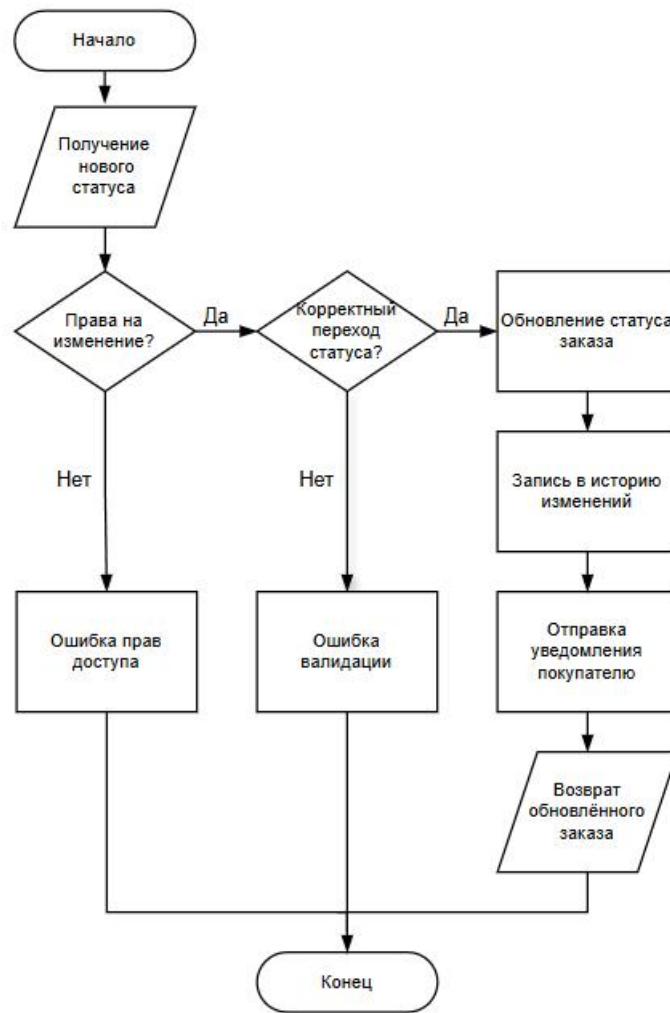


Рис. 3.13

### 3.7.3. Реализация классов

1. Класс Order расширен для управления статусами.  
Поля класса:

- id: Integer — идентификатор заказа
- user\_id: Integer — ссылка на покупателя
- status: String — текущий статус
- delivery\_address: String — адрес доставки

Основные методы:

- update\_status(new\_status, user\_id, comment, photo) — изменение статуса
- get\_status\_history() — получение истории статусов
- notify\_customer() — отправка уведомления

2. Класс OrderStatusHistory представляет историю изменений статуса заказа.

Поля класса:

- id: Integer — идентификатор записи
- order\_id: Integer — ссылка на заказ
- status: String — новый статус
- stage\_detail: Text — детализация этапа
- changed\_at: DateTime — время изменения

Основные методы:

- create\_entry(order\_id, status, details) — создание записи
- get\_timeline(order\_id) — получение истории заказа

### 3.7.4. Описание контрольного примера

Входные данные:

Заказ ID: 0001

Новый статус: ГОТОВИТСЯ\_К\_ОТПРАВКЕ

Комментарий мастера: «Платье готово, остались последние штрихи и можно явить миру красоту»

Фотоотчёт: загружено изображение процесса

История статусов в личном кабинете покупателя представлена на рис.3.14.

## Экранная форма истории статусов заказа

**Мои заказы**

Добро пожаловать, Анна Иванова!

Готовится к отправке

Номер заказа ORD-0001	Товар Платье из хлопчатобумажной ткани (XL)	Дата создания 20 октября 2025 г.
--------------------------	--	-------------------------------------

**История статусов заказа**

20 октября в 10:00  
Принят  
Заказ создан  
Мастер: Система

21 октября в 09:15  
Согласован  
Заказ согласован  
Мастер: Мария Петрова

21 октября в 09:15  
В работе  
Начата работа над заказом, материалы подготовлены  
Мастер: Мария Петрова

23 октября в 14:30  
Готовится к отправке  
Платье готово, остались последние штрихи и можно явить миру красоту!  
  
Мастер: Мария Петрова

Рис. 3.14