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. ___Б21-191-1з_____ 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

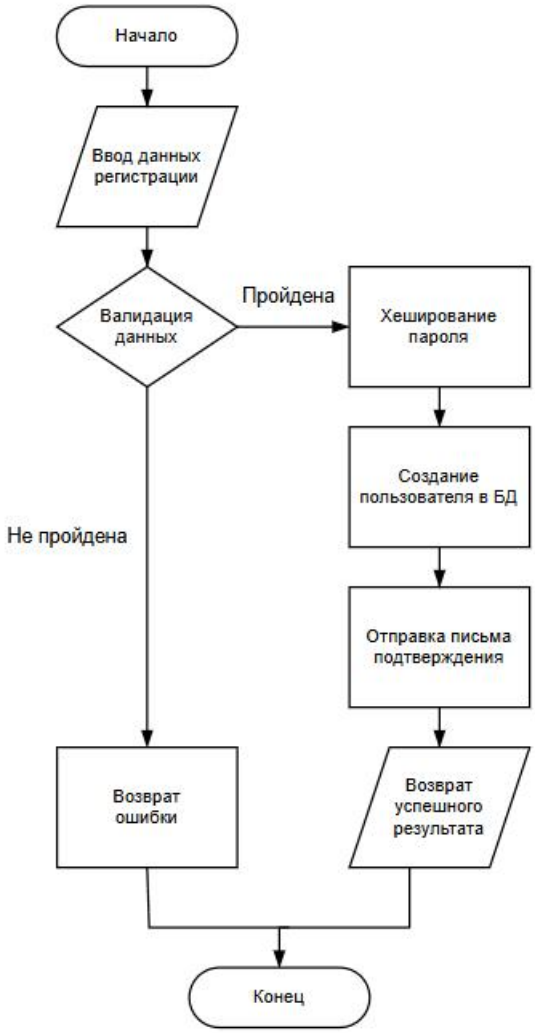


Рис. 3.1

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

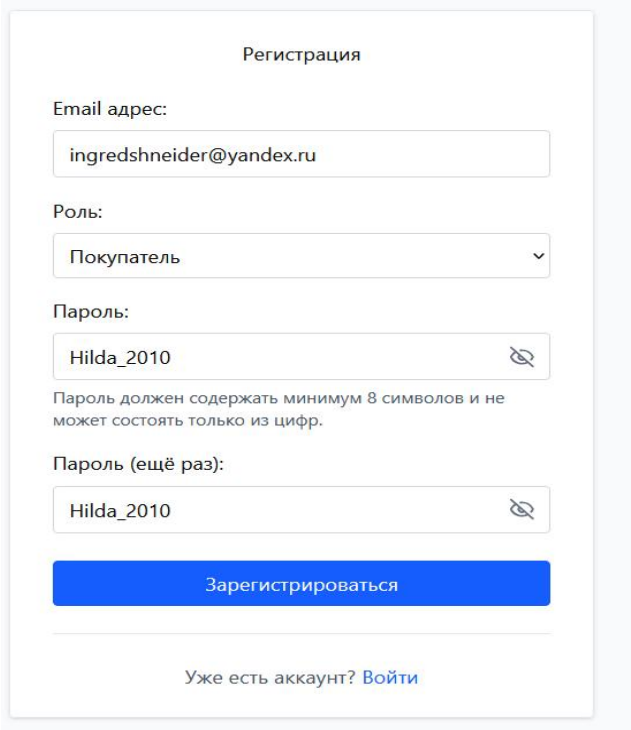


рис. 3.2

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

$TF(t,d) = f(t,d)/\Sigma f(t',d)$ // Частота термина в документе

$IDF(t) = \log(N/df(t))$ // Обратная частота документа

$TF - IDF(t,d) = TF(t,d) \times IDF(t)$ // 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

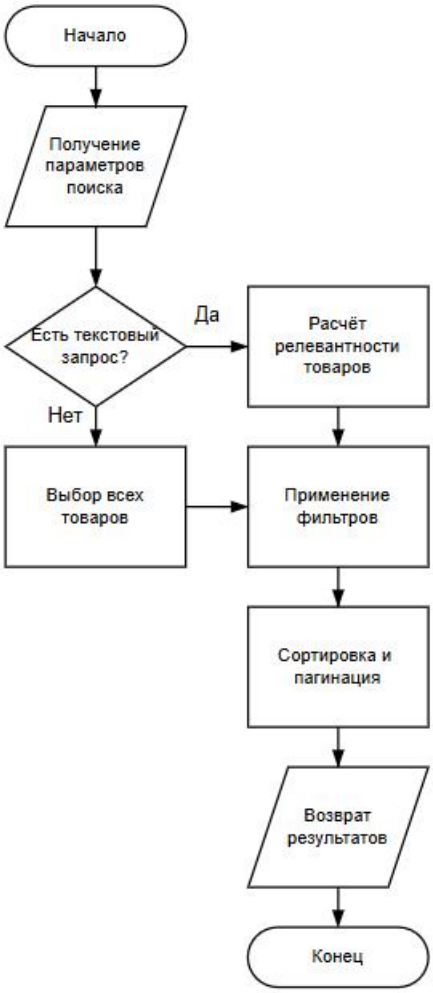


Рис. 3.3

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

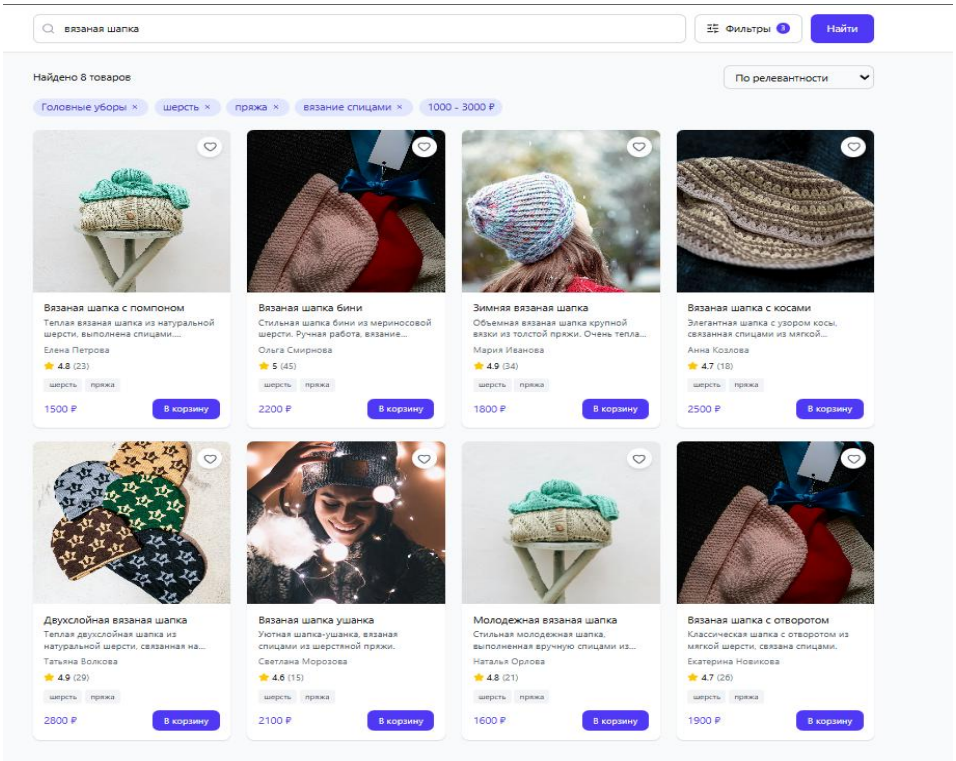


Рис. 3.4

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

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

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

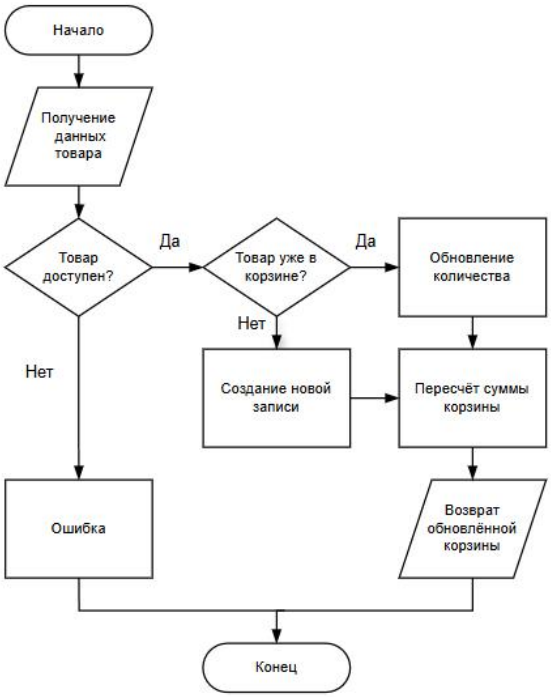


Рис. 3.5

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

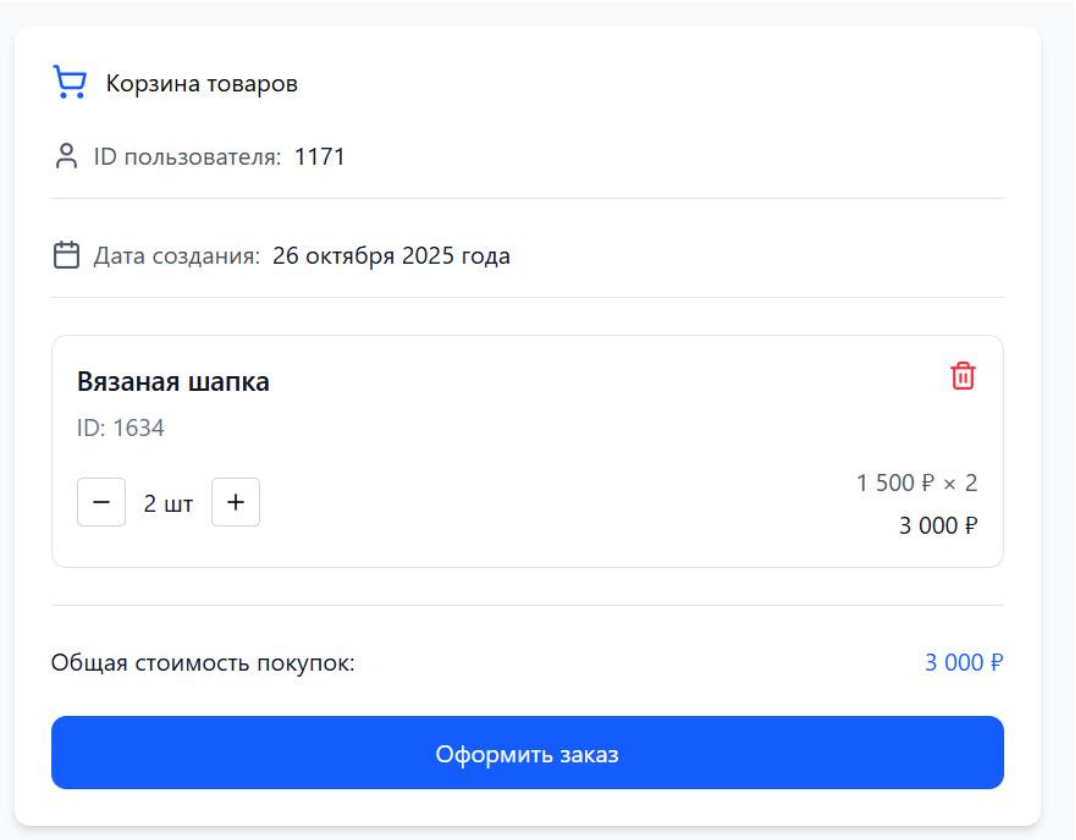


Рис. 3.6

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

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

T - общая сумма к оплате, P_i - цена i-го товара, Q_i - количество i-го товара, D - стоимость доставки, C - сумма скидки

Расчёт скидки

$$C = \begin{cases} 0.1 \times T\_base, если T\_base \geq 5000 \wedge promo = 'SUMMER10' \\ 0.05 \times T\_base, если T\_base \geq 3000 \\ 0, иначе \end{cases}$$

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

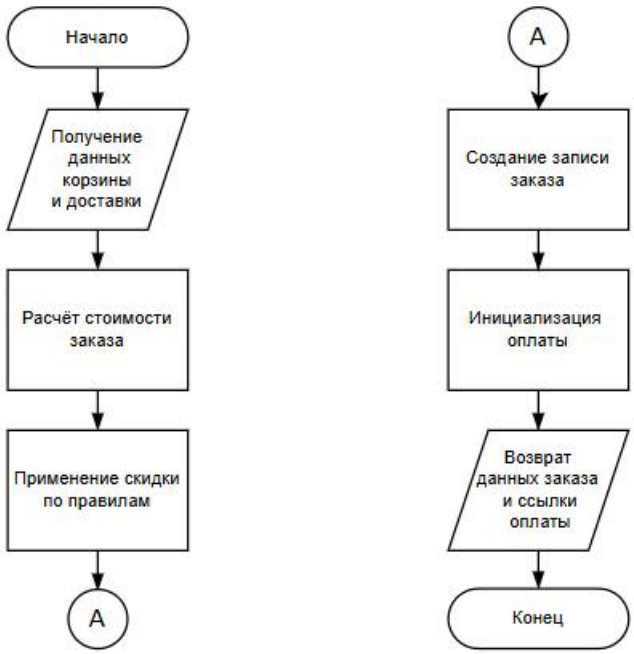


Рис. 3.7

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

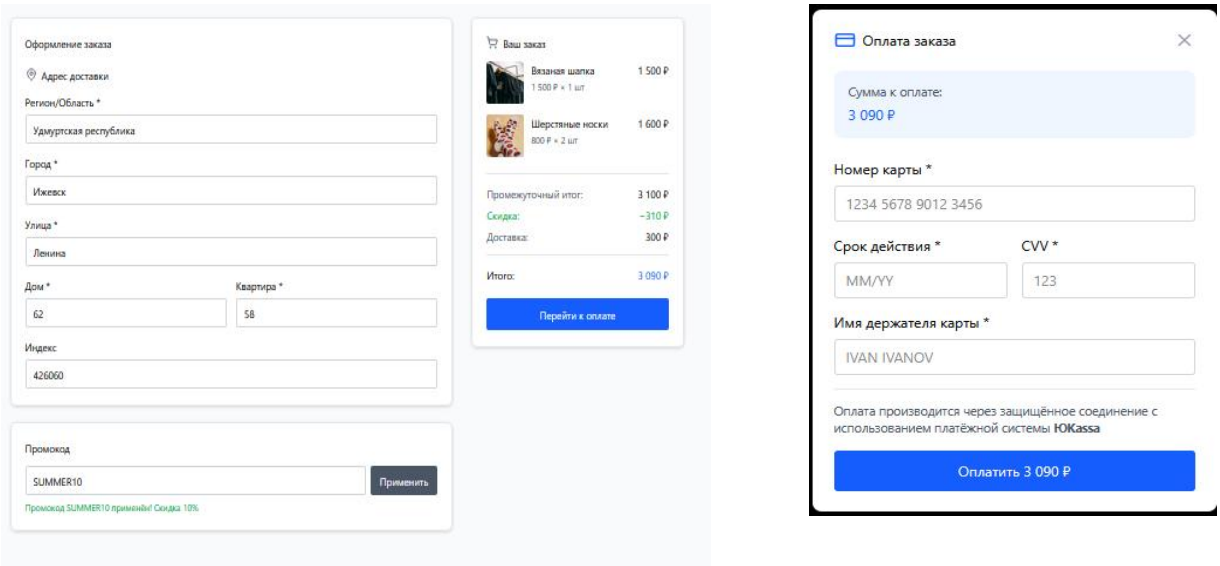
б) "Шерстяные носки": 800 руб × 2 шт

— Доставка: 300 руб

— Промокод: SUMMER10

Страница оформления заказа представлена на рис.3.8.

Экранная форма оформления заказа



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

$$C\_total = C\_base + \Sigma(C\_material) + \Sigma(C\_parameter) + C\_personalization$$

Cbase - базовая цена товара, Cmaterial - наценка за выбранные материалы, Cparametr - наценка за параметры (размер, цвет), Cpersonalization - стоимость персонализации

Расчёт срока

$$T\_total = T\_base + \Sigma(T\_material) + \Sigma(T\_parameter) + T\_personalization$$

Tbase - базовый срок изготовления в днях, Tmaterial - дополнительное время на работу с материалом, Tparameter - время на выполнение параметров, Tpersonalization - 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

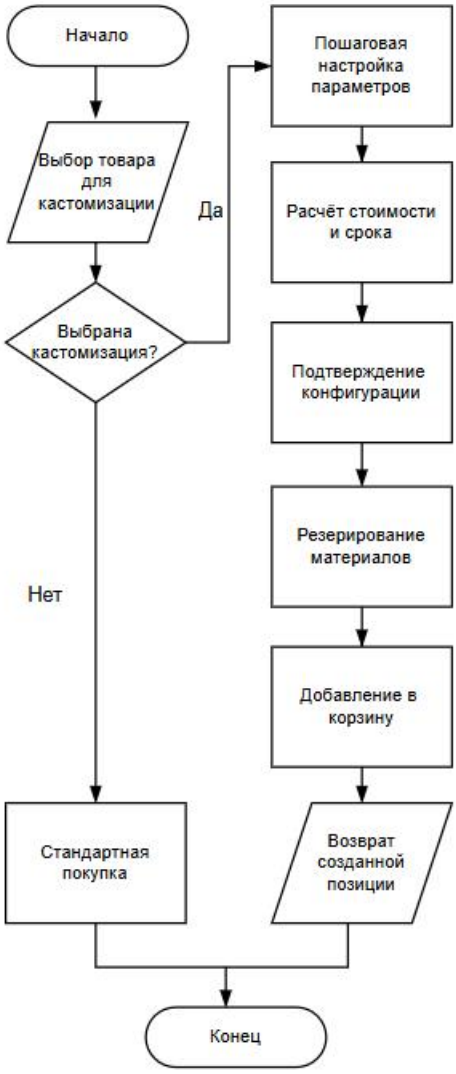


Рис. 3.9

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)

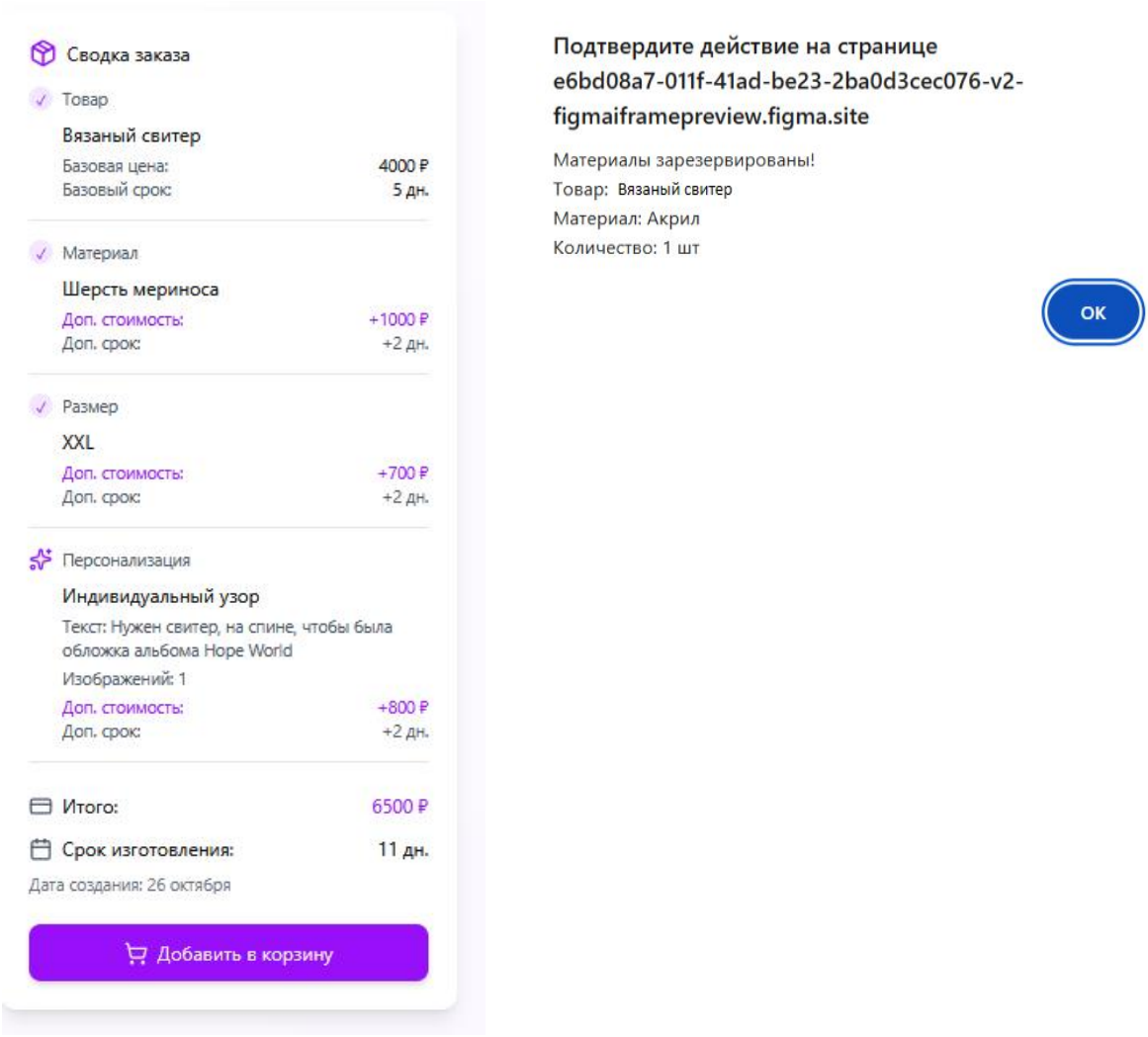
Выбор материалов: шерсть мериноса(+1000 руб, +2 дня)

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



а) 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

$$Q\_available = Q\_initial - \Sigma(N\_i \times R\_i \times (1 + waste\_factor))$$

Q_available - доступное кол-во каждого материала, Q_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

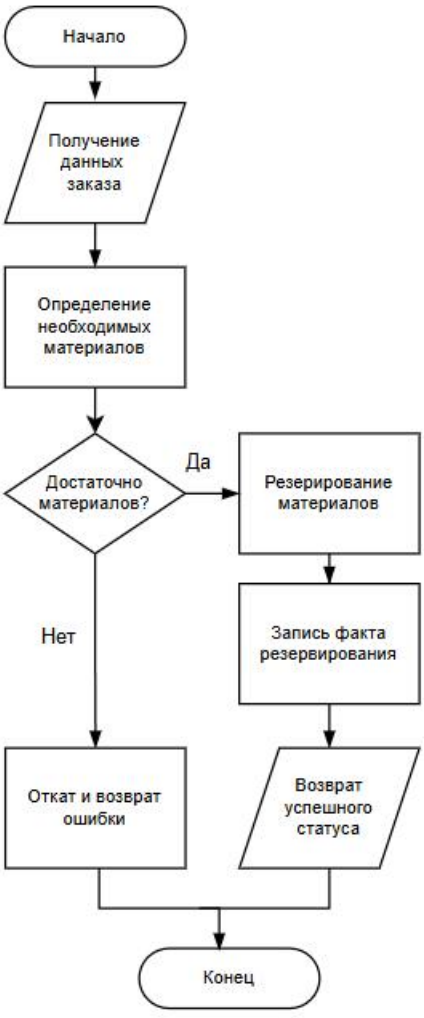


Рис. 3.11

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 мериноса» (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

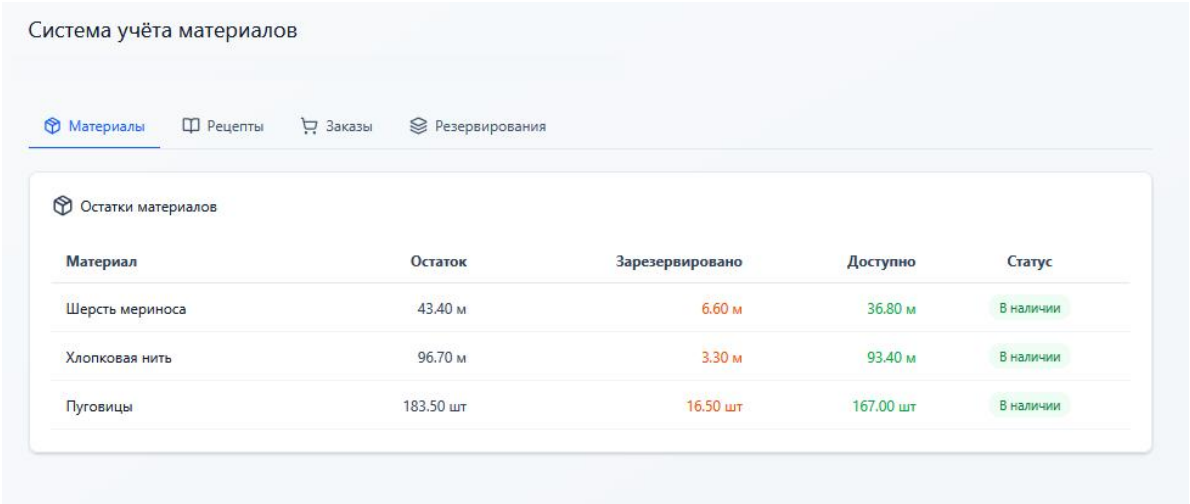


Рис. 3.12

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 = {ПРИНЯТ, СОГЛАСОВАН, В_РАБОТЕ, ГОТОВИТСЯ_К_ОТПРАВКЕ, ОТПРАВЛЕН, ДОСТАВЛЕН, ОТМЕНЁН}

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

$F = \{$ДОСТАВЛЕН, 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

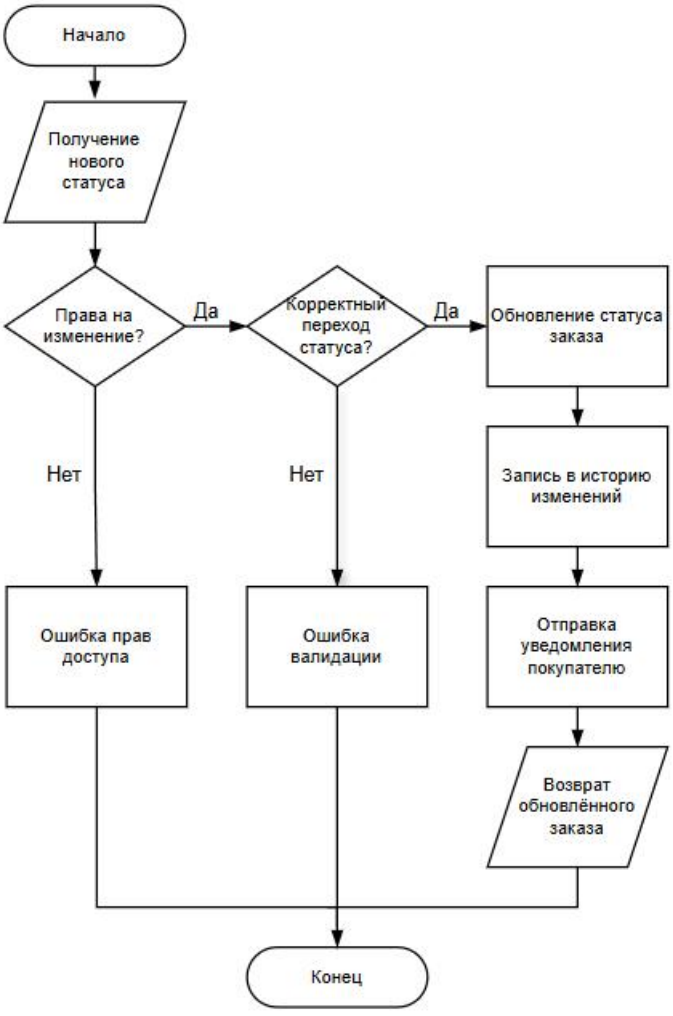


Рис. 3.13

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 являть миру красоту)»

Фотоотчёт: загружено изображение процесса

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

Экранная форма истории статусов заказа

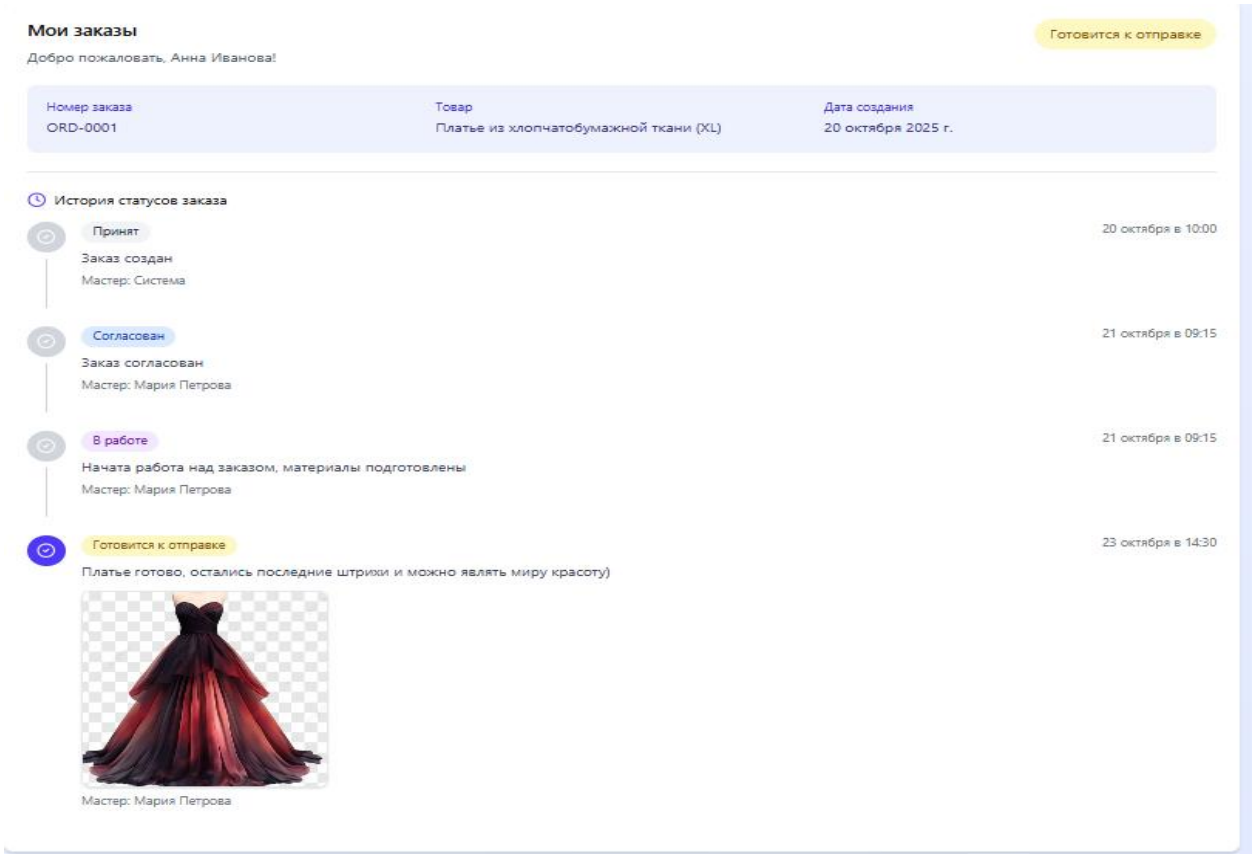


Рис. 3.14