Кхари Жекка Кализая Арсе

Покупка мячей для любой игры

1. Перехожу на веб-сайт
2. Выбрат из опцей мяч смотряя на картину
3. Нажимаю на кнопку «в корзину»
4. Перехожу в корзину
5. Нажимаю кнопку «перейти к оформлению»
6. Заполняю данные для покупки (карта и адрес)
7. Нажимаю кнопку оформить заказ
8. Жду мой заказ

Проблема:

При покупке мячей, никогда нет опции, которая обеспечивает что ты покупаешь самый хороший вариянт за стоимость денгей. Должен быть специяльный фильтр для тех, которые ищут не самый хороший и не самый плохой, а самый оптимальный вариант.

Режение:

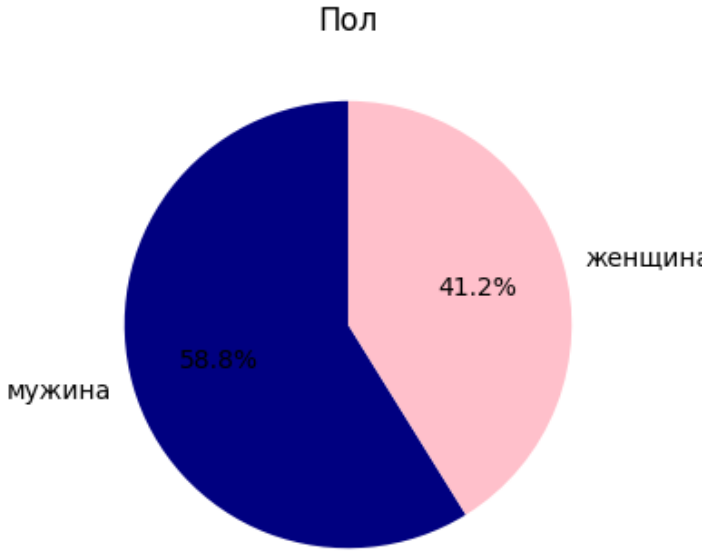
Можно добавить такой специяльный фильтр

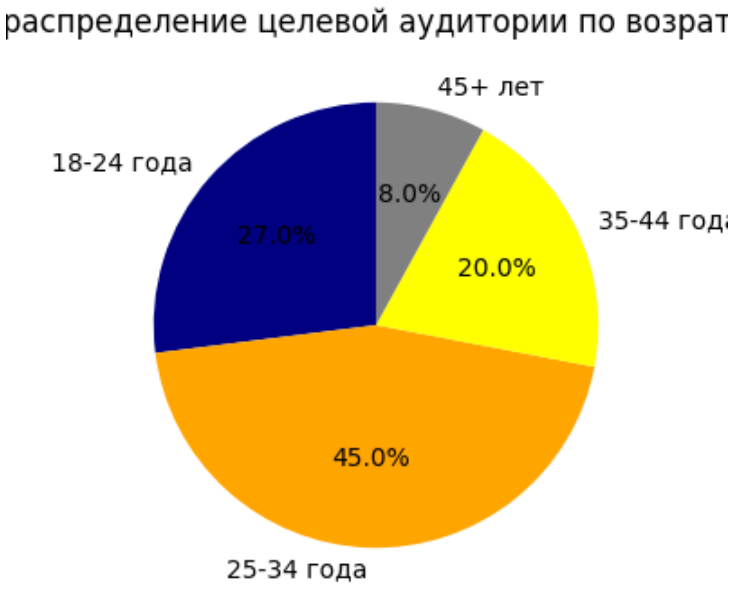
Тема ис: Онлайн-магазин для продажа мячей для любой игры

1. **Цель исследования:** Провести оценку вариантов и экономической целесообразности проекта на основе анализа рынка, потребностей целевой аудитории и существующих рыночных предложений, а также оценить потенциальную конкуренцию и риски в краткосрочной и долгосрочной перспективе.

2. **Задачи исследования:**
    a. Провести оценку рыночной потребности.
    b. Выполнить анализ онлайн-площадок, таких как Amazon, Wildberries и Ozon.
    c. Оценить затраты на инфраструктуру и программное обеспечение.
    d. Определить объем инвестиций, необходимых для реализации проекта.
3. **Целевая аудитория:**
    a. В одном исследовании в Испании была определена разница в процентах между мужчинами и женщинами, которые занимаются каким-либо видом спорта.

    b. Другие исследования определяют количество населения по возрастным группам.


Пол
женщина
41.2%
58.8%
мужина


распределение целевой аудитории по возрат
45+ лет
18-24 года
8.0%
35-44 года
27.0%
20.0%
45.0%
25-34 года

4. **4P-анализ**

PRODUCTION Вся продукция будет экспортироваться из Китая морским путем с прогнозируемым запасом на 4 месяца. В течение этого периода мы будем привозить различные бренды и качества, которые будут оцениваться и ранжироваться для обеспечения оптимальной системы фильтрации по качеству и цене.	**PRICE** Цена продукции будет определяться в процентном соотношении: от 200% для продукции наивысшего качества до 10% для продукции низшего качества.
PLACE Изначально серверы будут расположены только в России, работа будет вестись исключительно через службу доставки, начиная с Москвы, затем Санкт-Петербурга, а позже — в различных регионах России и мира, когда будет обеспечена возможность доставки и достаточная доступность серверов для удовлетворения спроса.	**PROMOTION** В связи с ростом популярности социальных сетей, наилучшим вариантом является продвижение моего интернет-магазина через платформы TikTok, Facebook и Instagram, а также привлечение стримеров и блогеров на YouTube для рекламы магазина.

5. **SWOT-анализ.**

Магазин «wildberries»

Слабые стороны **Лидер в российской электронной коммерции.** **Обширная сеть пунктов сбора.** **Большое разнообразие продуктов и брендов.**	**Сильные стороны** **Зависимость от внешних поставщиков.** **Логистические проблемы из-за быстрого расширения.**
Возможности **Международная экспансия.** **Развитие финансовых услуг.**	**Угрозы** **высокое доминирование на рынке, возможность нечестной игры**

Магазин «ozon»

Слабые стороны Признан “Амазонкой России". Устойчивый рост доходов. Диверсификация услуг (торговля + финансы).	Сильные стороны Сильная конкуренция с Wildberries. Потребность в значительной логистической инфраструктуре.
Возможности Расширение финансовых услуг. Стратегические партнерские отношения с технологическими компаниями.	Угрозы высокое доминирование на рынке, возможность нечестной игры

Магазин «sportmaster»

Слабые стороны Специалист по спортивным товарам. Физическое присутствие и присутствие в Интернете. Разнообразие брендов и продуктов.	Сильные стороны Ограниченное международное присутствие. Зависимость от российского рынка.
Возможности Расширять ассортимент предлагаемой продукции. Укрепление платформы электронной коммерции.	Угрозы прямой конкурент. простота принятия рисков и действий против своих конкурентов, большой капитал

6. **Временной период создания**

Для построения инфраструктуры, как веб-, так и физической, потребуется ориентировочно от **3 до 6 месяцев**, с учётом найма программистов и специалистов, а также заключения контрактов с облачными сервисами.

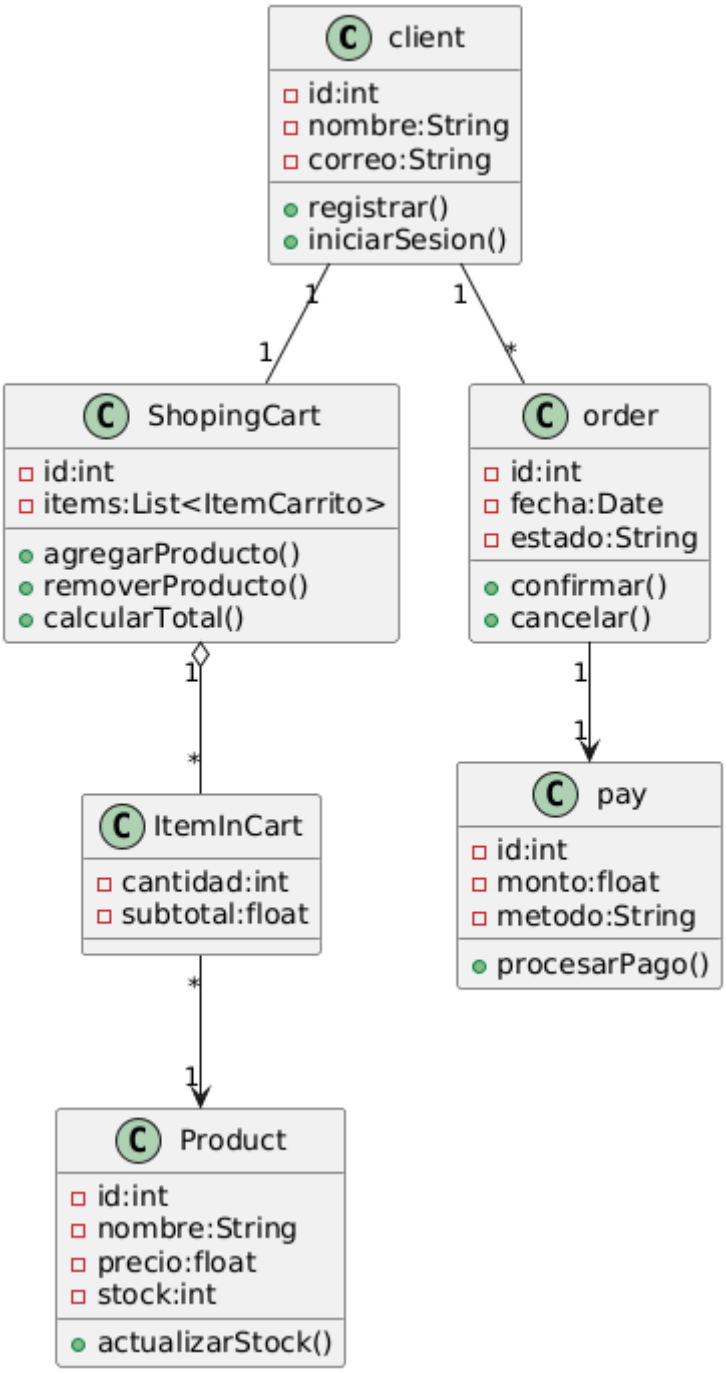
Поиск поставщиков займёт не менее **4 месяцев**, в течение которых необходимо будет привлечь переводчика, совершить поездку в Китай и

изучить рынок производителей мячей. К данному сроку следует прибавить время на заключение контрактов и производство продукции, что в совокупности формирует **минимальный срок начала деятельности около 10 месяцев**.

диаграмы следаны через PlantUML Editor

СТРУКТУРНЫЕ ДИАГРАММЫ

1. **Диаграмма классов**

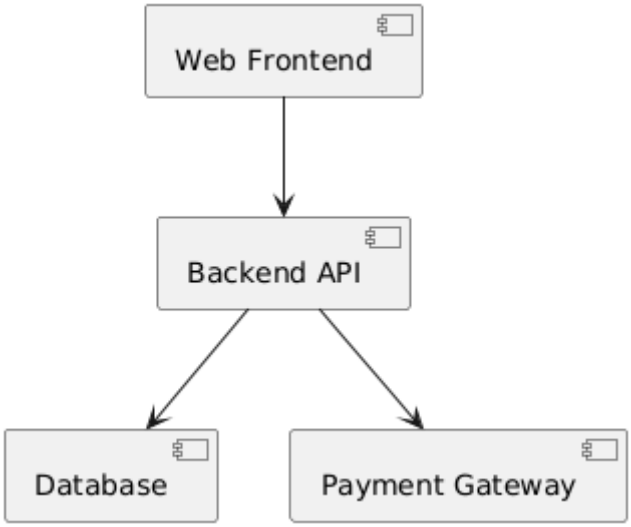


код:

```
@startuml
class client {
 -id:int
 -nombre:String
 -correo:String
 +registrar()
 +iniciarSesion()
}

class ShopingCart {
 -id:int
 -items:List<ItemCarrito>
 +agregarProducto()
 +removerProducto()
 +calcularTotal()
}

class ItemInCart {
 -cantidad:int
 -subtotal:float
}

class Product {
 -id:int
 -nombre:String
 -precio:float
 -stock:int
```

```
 +actualizarStock()
}

class order {
 -id:int
 -fecha:Date
 -estado:String
 +confirmar()
 +cancelar()
}

class pay {
 -id:int
 -monto:float
 -metodo:String
 +procesarPago()
}

client "1" -- "1" ShopingCart
ShopingCart "1" o-- "*" ItemInCart
ItemInCart "*" --> "1" Product
client "1" -- "*" order
order "1" --> "1" pay
@enduml
```

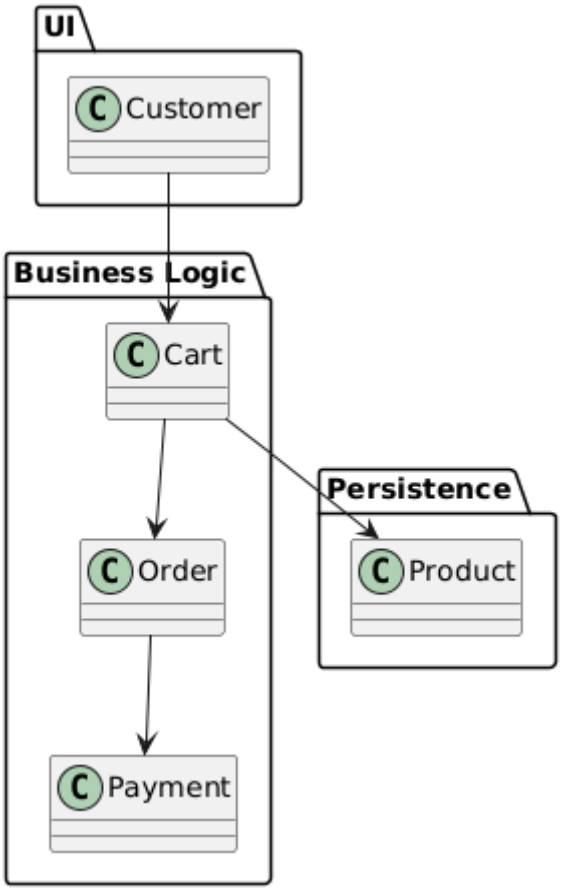
2. **Диаграмма компонентов**



код:

```
@startuml
component "Web Frontend" as FE
component "Backend API" as BE
component "Database" as DB
component "Payment Gateway" as PAY

FE --> BE
BE --> DB
BE --> PAY
@enduml
```

3. **Диаграмма пакетов**



код:

```
@startuml
package "UI" {
 class Customer
}

package "Business Logic" {
 class Cart
 class Order
 class Payment
}

package "Persistence" {
 class Product
```

}

Customer --> Cart

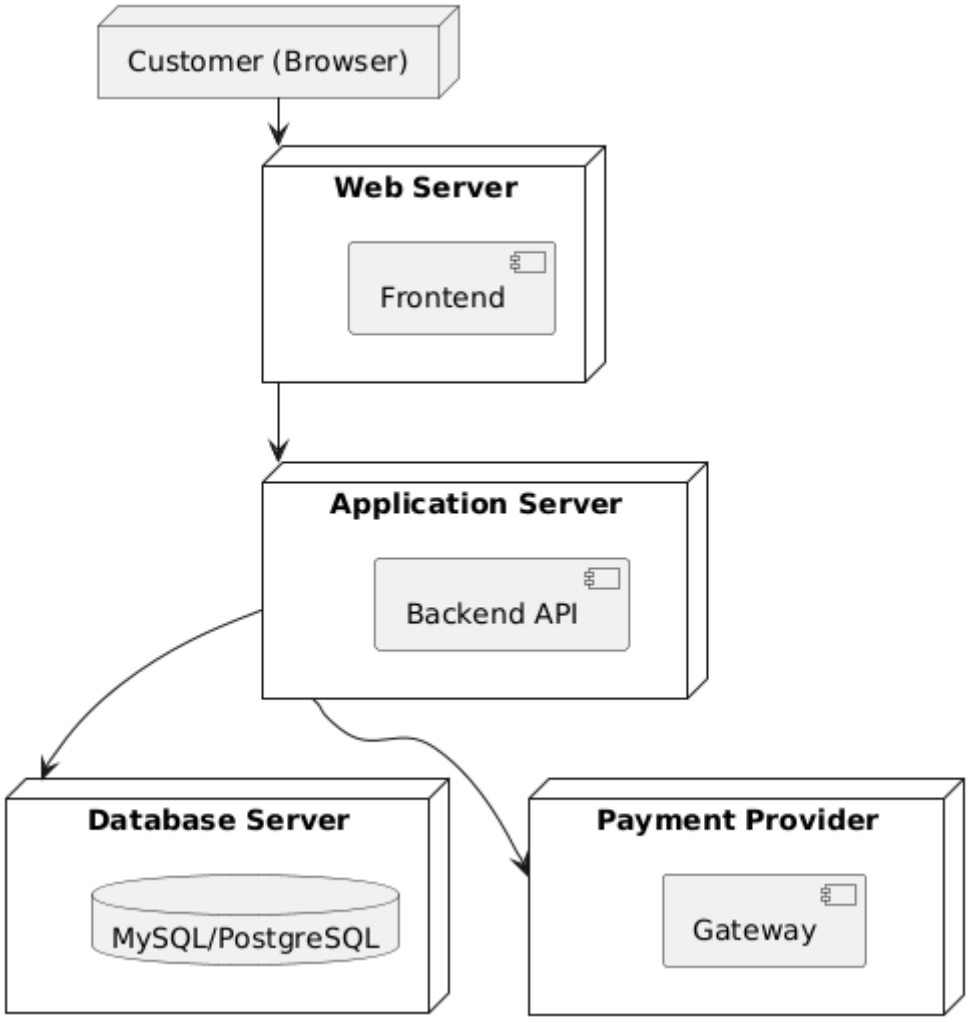
Cart --> Order

Order --> Payment

Cart --> Product

@enduml

4. **Диаграмма развертывания**

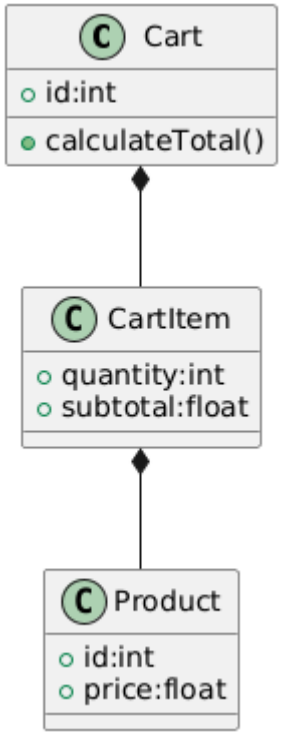


код:

@startuml

```
node "Customer (Browser)" {
}

node "Web Server" {
  component "Frontend"
}

node "Application Server" {
  component "Backend API"
}

node "Database Server" {
  database "MySQL/PostgreSQL"
}

node "Payment Provider" {
  component "Gateway"
}

"Customer (Browser)" --> "Web Server"
"Web Server" --> "Application Server"
"Application Server" --> "Database Server"
"Application Server" --> "Payment Provider"
@enduml
```

5. **Диаграмма композитной структуры**



код:

```
@startuml
class Cart {
 +id:int
 +calculateTotal()
}

class CartItem {
 +quantity:int
 +subtotal:float
}

class Product {
 +id:int
 +price:float
}
```

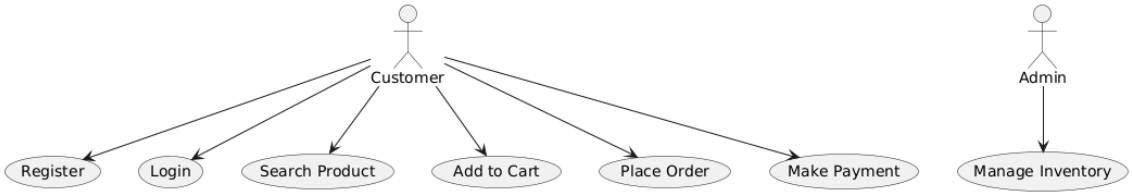
Cart *-- CartItem

CartItem *-- Product

@enduml

Поведенческие диаграммы

1. **Use Case**



код:

@startuml

actor Customer

actor Admin

usecase "Register" as UC1

usecase "Login" as UC2

usecase "Search Product" as UC3

usecase "Add to Cart" as UC4

usecase "Place Order" as UC5

usecase "Make Payment" as UC6

usecase "Manage Inventory" as UC7

Customer --> UC1

Customer --> UC2

Customer --> UC3

Customer --> UC4

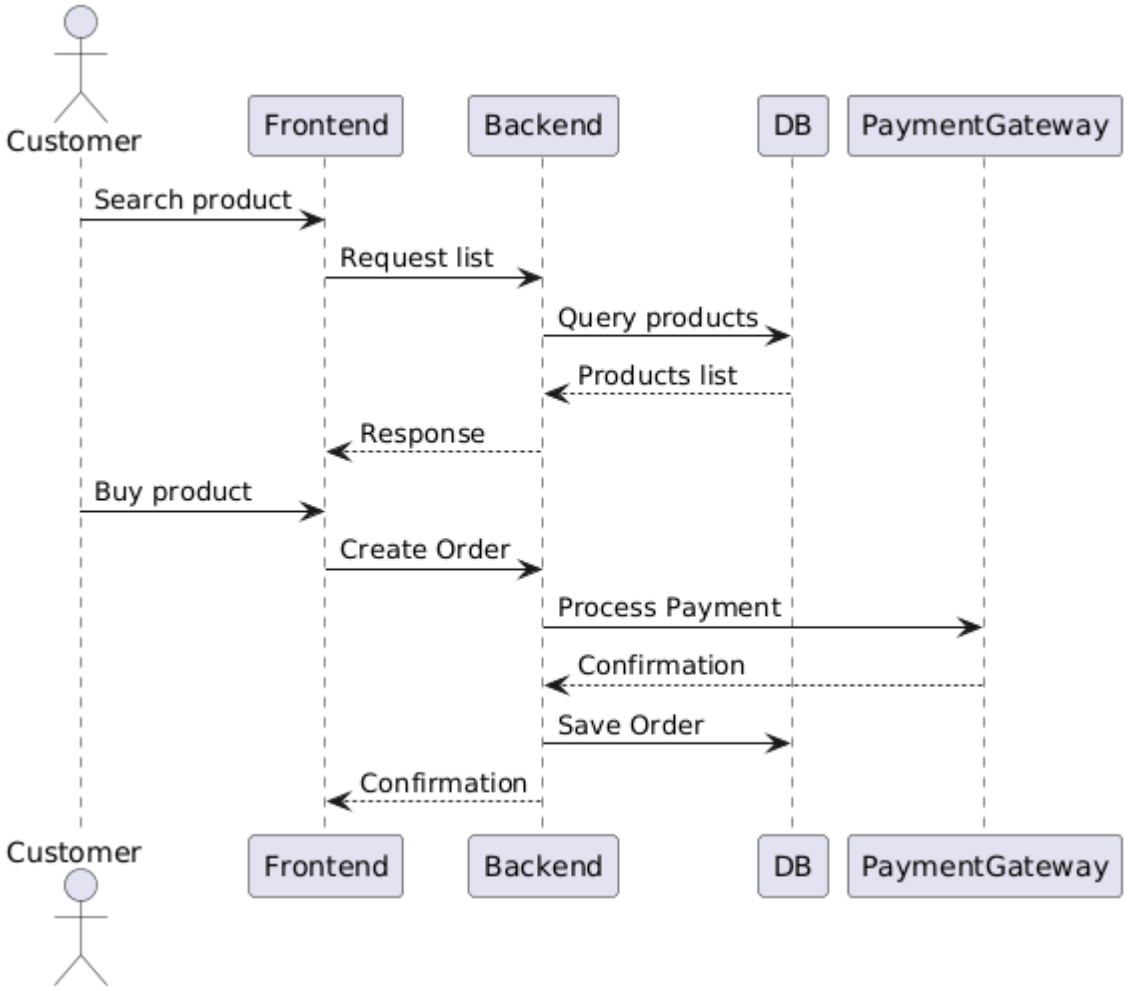
Customer --> UC5

Customer --> UC6

Admin --> UC7

@enduml

2. **Диаграмма последовательностей**



код:

@startuml

actor Customer

participant "Frontend" as FE

participant "Backend" as BE

participant "DB" as DB

participant "PaymentGateway" as PAY

Customer -> FE: Search product

FE -> BE: Request list

BE -> DB: Query products

DB --> BE: Products list

BE --> FE: Response

Customer -> FE: Buy product

FE -> BE: Create Order

BE -> PAY: Process Payment

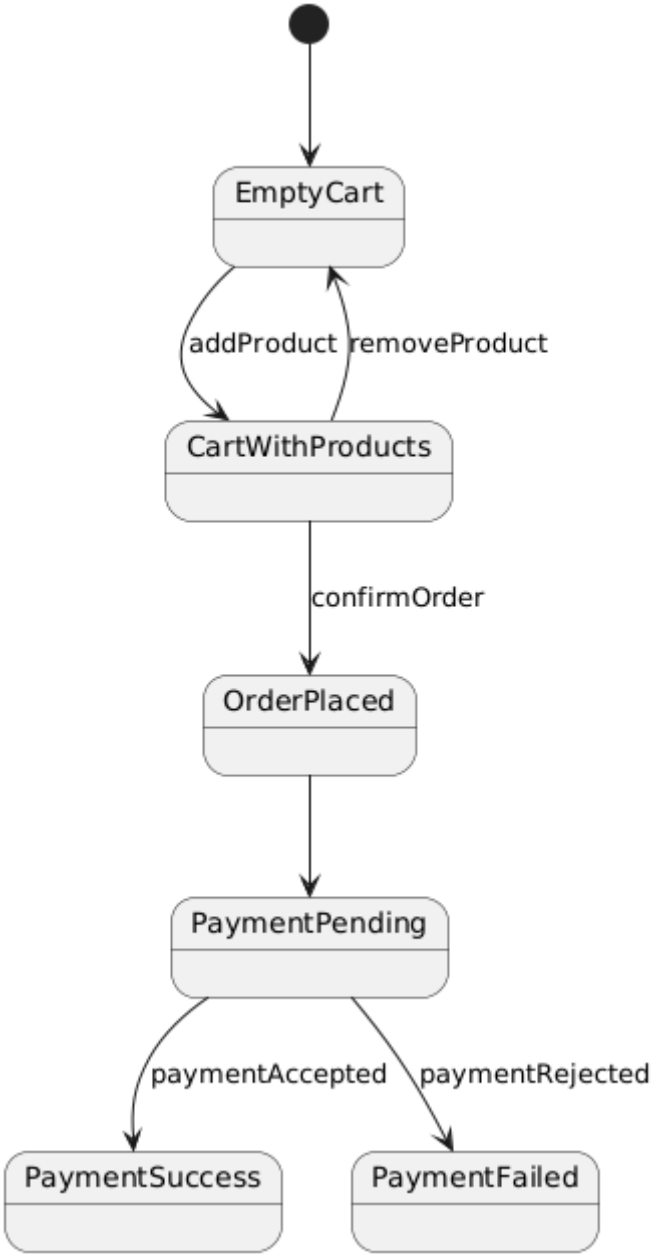
PAY --> BE: Confirmation

BE -> DB: Save Order

BE --> FE: Confirmation

@enduml

3. **диаграмма состояний**



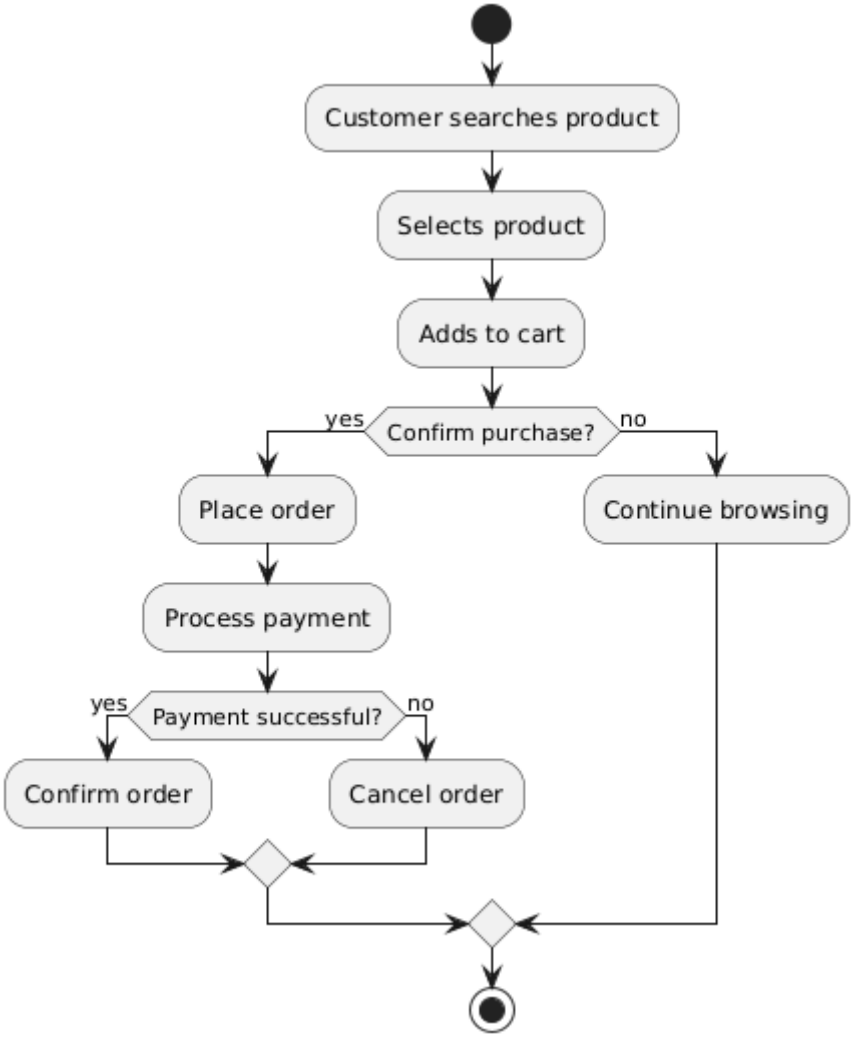
код:

```
@startuml

[*] --> EmptyCart

EmptyCart --> CartWithProducts : addProduct

CartWithProducts --> EmptyCart : removeProduct

CartWithProducts --> OrderPlaced : confirmOrder

OrderPlaced --> PaymentPending

PaymentPending --> PaymentSuccess : paymentAccepted
```

PaymentPending --> PaymentFailed : paymentRejected

@enduml

4. **диаграмма деятельности**



код:

@startuml

start

:Customer searches product;

:Selects product;

:Adds to cart;

if (Confirm purchase?) then (yes)

:Place order;

:Process payment;

if (Payment successful?) then (yes)

:Confirm order;

else (no)

:Cancel order;

endif

else (no)

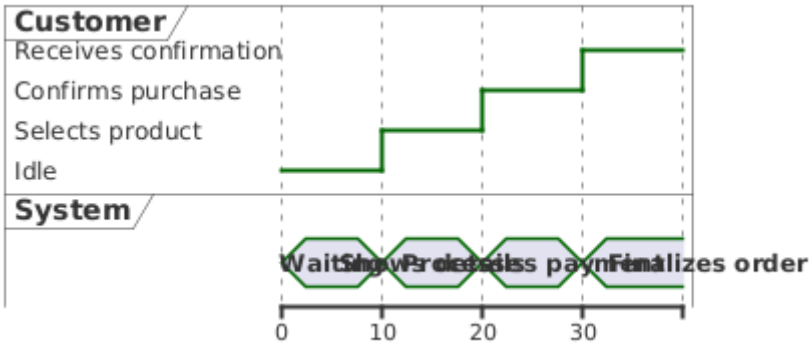
:Continue browsing;

endif

stop

@enduml

5. **временная диаграмма**



код:

@startuml

robust Customer

concise System

@0

Customer is Idle

System is Waiting

@10

Customer is "Selects product"

System is "Shows details"

@20

Customer is "Confirms purchase"

System is "Processes payment"

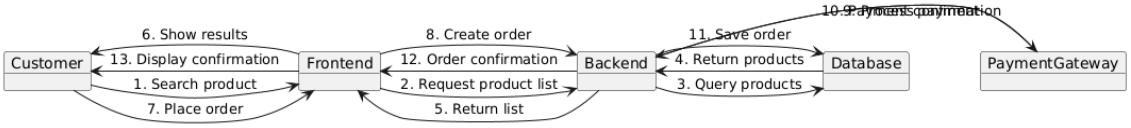
@30

Customer is "Receives confirmation"

System is "Finalizes order"

@enduml

6. **диаграмма коммуникации**



код:

@startuml

object Customer

object Frontend

object Backend

object Database

object PaymentGateway

Customer -> Frontend : 1. Search product

Frontend -> Backend : 2. Request product list

Backend -> Database : 3. Query products

Database --> Backend : 4. Return products

Backend --> Frontend : 5. Return list

Frontend --> Customer : 6. Show results

Customer -> Frontend : 7. Place order

Frontend -> Backend : 8. Create order

Backend -> PaymentGateway : 9. Process payment

PaymentGateway --> Backend : 10. Payment confirmation

Backend -> Database : 11. Save order

Backend --> Frontend : 12. Order confirmation

Frontend --> Customer : 13. Display confirmation

@enduml

МОДЕЛИ:

1. ARIS-модель

а) Организационная модель

```
@startuml
!theme plain

rectangle "Клиент" as Client
rectangle "Сайт (онлайн-магазин)" as Web
rectangle "Менеджер заказов" as Manager
rectangle "Курьер" as Courier

Client -down-> Web
Web -down-> Manager
Manager -down-> Courier
@enduml
```

b) Функциональная модель

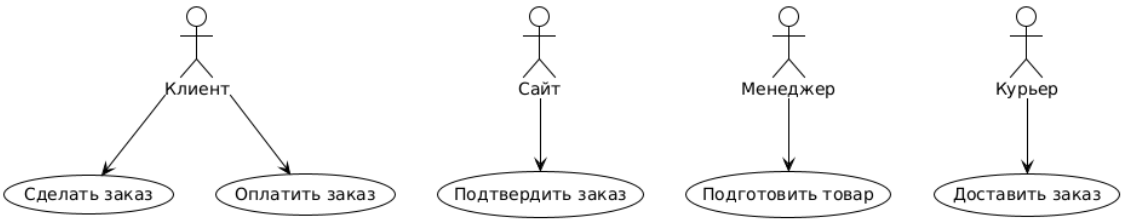
```
@startuml
!theme plain

usecase "Сделать заказ" as UC1
usecase "Оплатить заказ" as UC2
usecase "Подтвердить заказ" as UC3
usecase "Подготовить товар" as UC4
usecase "Доставить заказ" as UC5

actor "Клиент" as Client
actor "Менеджер" as Manager
actor "Курьер" as Courier
actor "Сайт" as Web

Client --> UC1
Client --> UC2
```

Web --> UC3

Manager --> UC4

Courier --> UC5

@enduml



c) Модель данных

@startuml

!theme plain

entity "Клиент" {

*id_клиента : INT

имя : TEXT

адрес : TEXT

телефон : TEXT

}

entity "Товар" {

*id_товара : INT

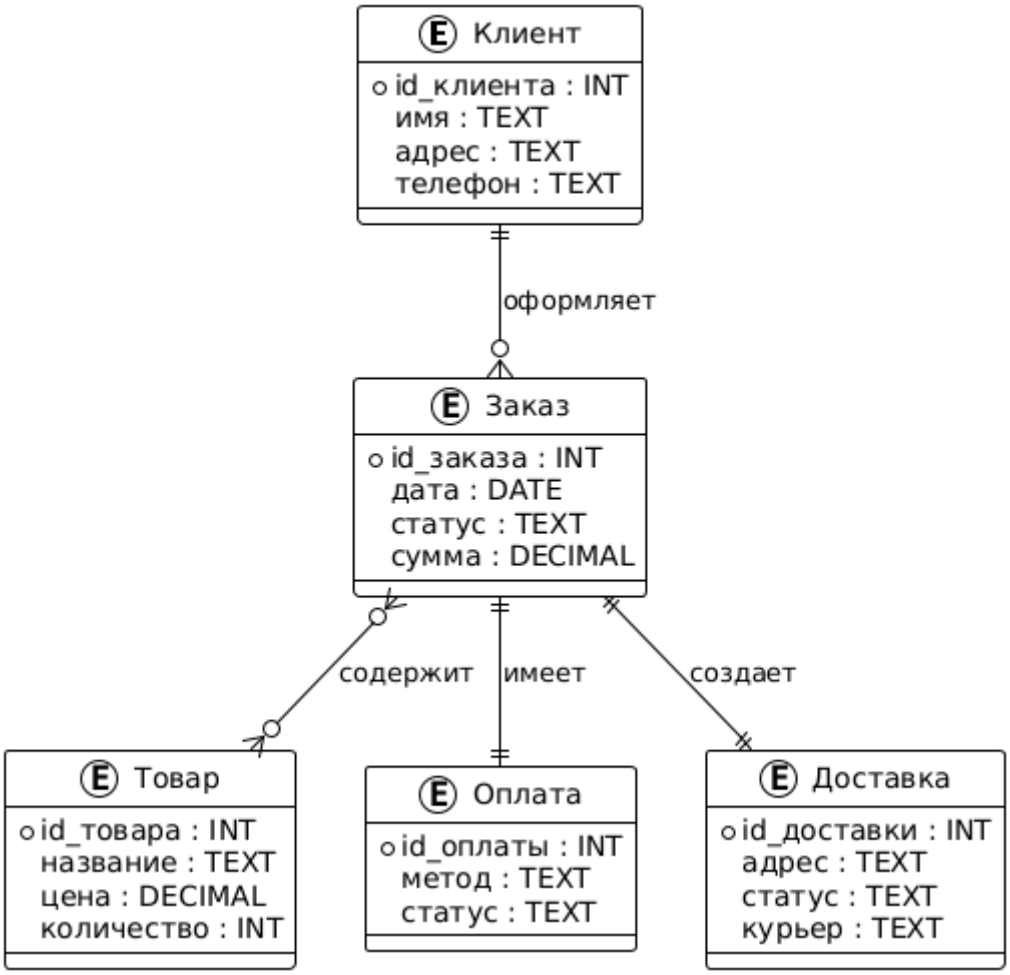
название : TEXT

цена : DECIMAL

количество : INT

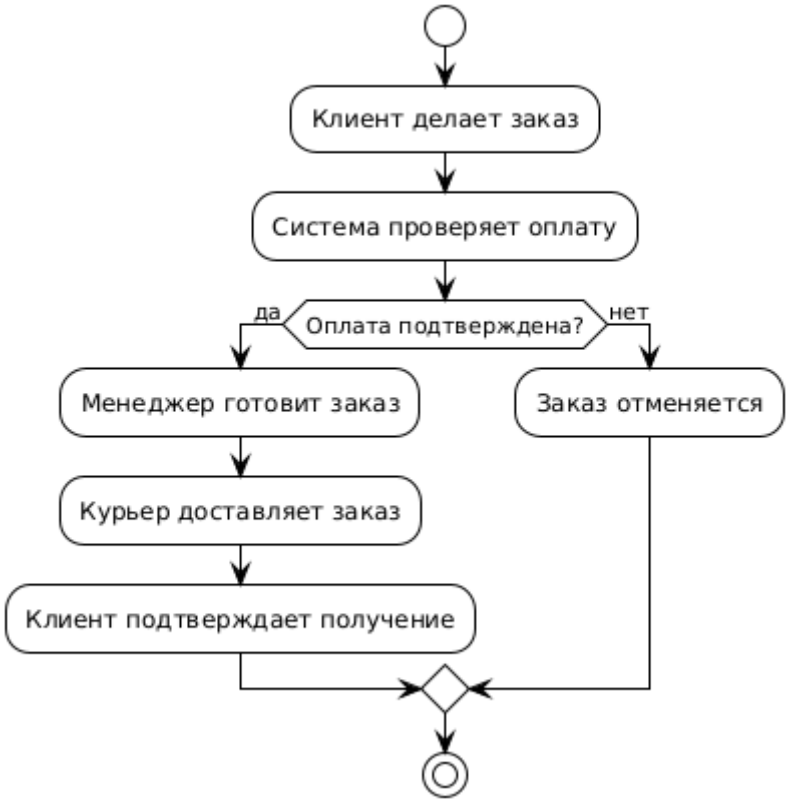
}

```
entity "Заказ" {
  *id_заказа : INT
  дата : DATE
  статус : TEXT
  сумма : DECIMAL
}

entity "Оплата" {
  *id_оплаты : INT
  метод : TEXT
  статус : TEXT
}

entity "Доставка" {
  *id_доставки : INT
  адрес : TEXT
  статус : TEXT
  курьер : TEXT
}

"Клиент" ||--o{ "Заказ" : "оформляет"
"Заказ" }o--o{ "Товар" : "содержит"
"Заказ" ||--|| "Оплата" : "имеет"
"Заказ" ||--|| "Доставка" : "создает"
@enduml
```

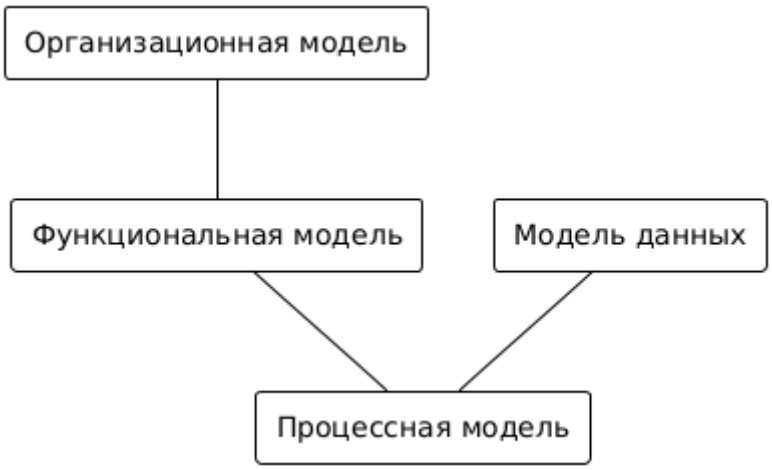

Клиент
id_клиента : INT
имя : TEXT
адрес : TEXT
телефон : TEXT
оформляет
Заказ
id_заказа : INT
дата : DATE
статус : TEXT
сумма : DECIMAL
содержит
имеет
создает
Товар
id_товара : INT
название : TEXT
цена : DECIMAL
количество : INT
Оплата
id_оплаты : INT
метод : TEXT
статус : TEXT
Доставка
id_доставки : INT
адрес : TEXT
статус : TEXT
курьер : TEXT

d) Процессная модель

```
@startuml
!theme plain
start
:Клиент делает заказ;
:Система проверяет оплату;
if (Оплата подтверждена?) then (да)
  :Менеджер готовит заказ;
  :Курьер доставляет заказ;
  :Клиент подтверждает получение;
else (нет)
  :Заказ отменяется;
endif
stop
@enduml
```



е) Диаграмма связей между представлениями



2. Информационная модель с бизнес-правилами

а) Бизнес-правила (Business Rules, текст)

@startmindmap

* Бизнес-правила

** Заказ не создается, если нет товара на складе

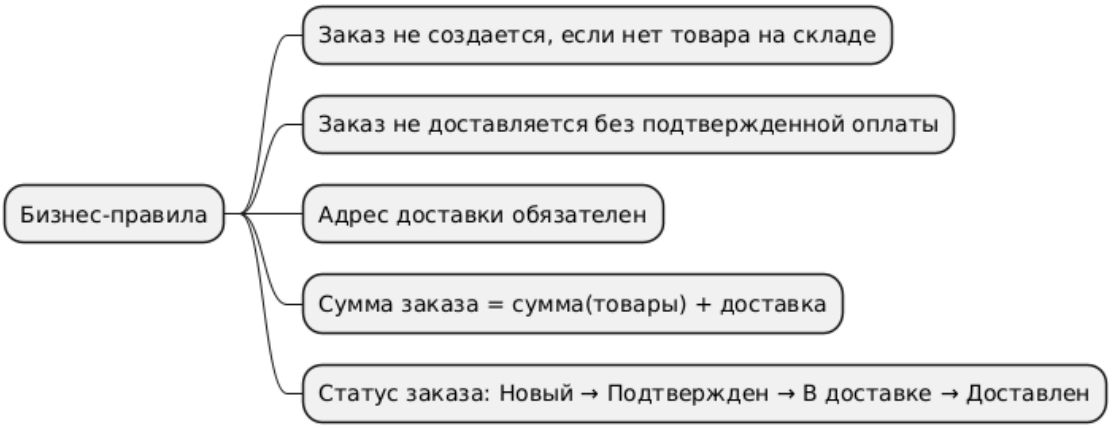
** Заказ не доставляется без подтвержденной оплаты

** Адрес доставки обязателен

** Сумма заказа = сумма(товары) + доставка

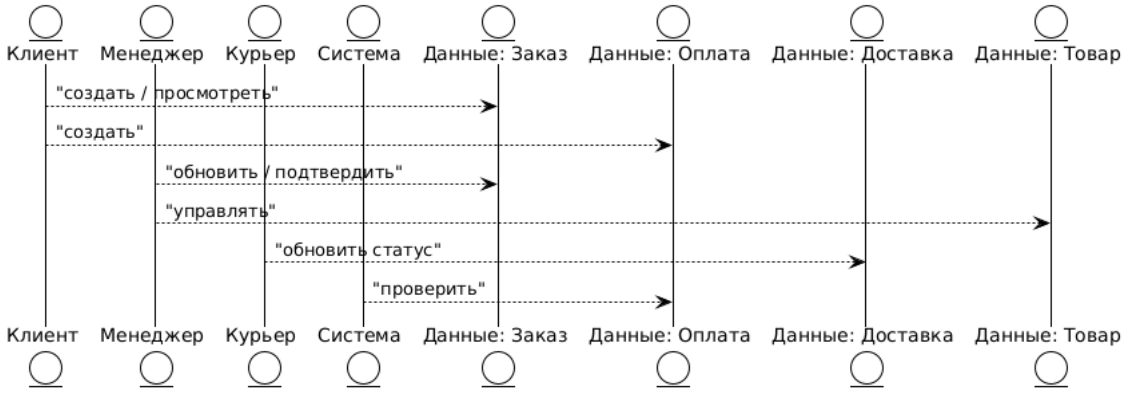
** Статус заказа: Новый → Подтвержден → В доставке → Доставлен

@endmindmap



b) Матрица доступа к данным

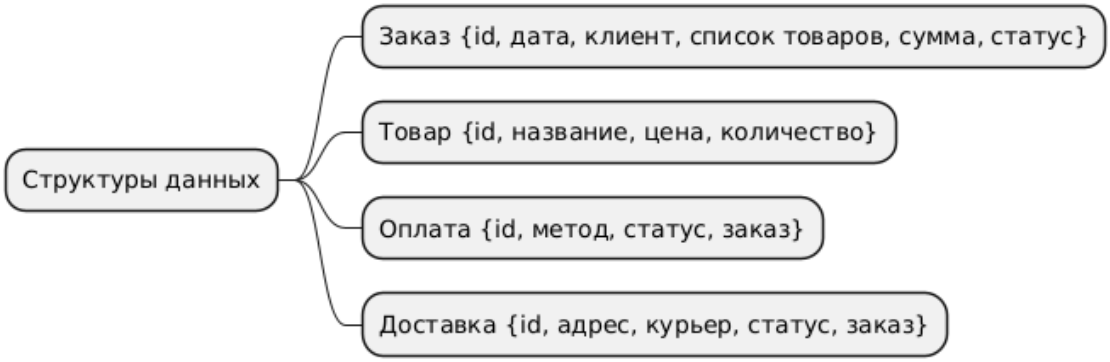
```
@startuml
!theme plain

entity "Клиент" as Client
entity "Менеджер" as Manager
entity "Курьер" as Courier
entity "Система" as System

entity "Данные: Заказ"
entity "Данные: Оплата"
entity "Данные: Доставка"
entity "Данные: Товар"

Client --> "Данные: Заказ" : "создать / просмотреть"
Client --> "Данные: Оплата" : "создать"
Manager --> "Данные: Заказ" : "обновить / подтвердить"
Manager --> "Данные: Товар" : "управлять"
Courier --> "Данные: Доставка" : "обновить статус"
System --> "Данные: Оплата" : "проверить"
@enduml
```



3. Математическая модель системы

a) Базовые структуры данных

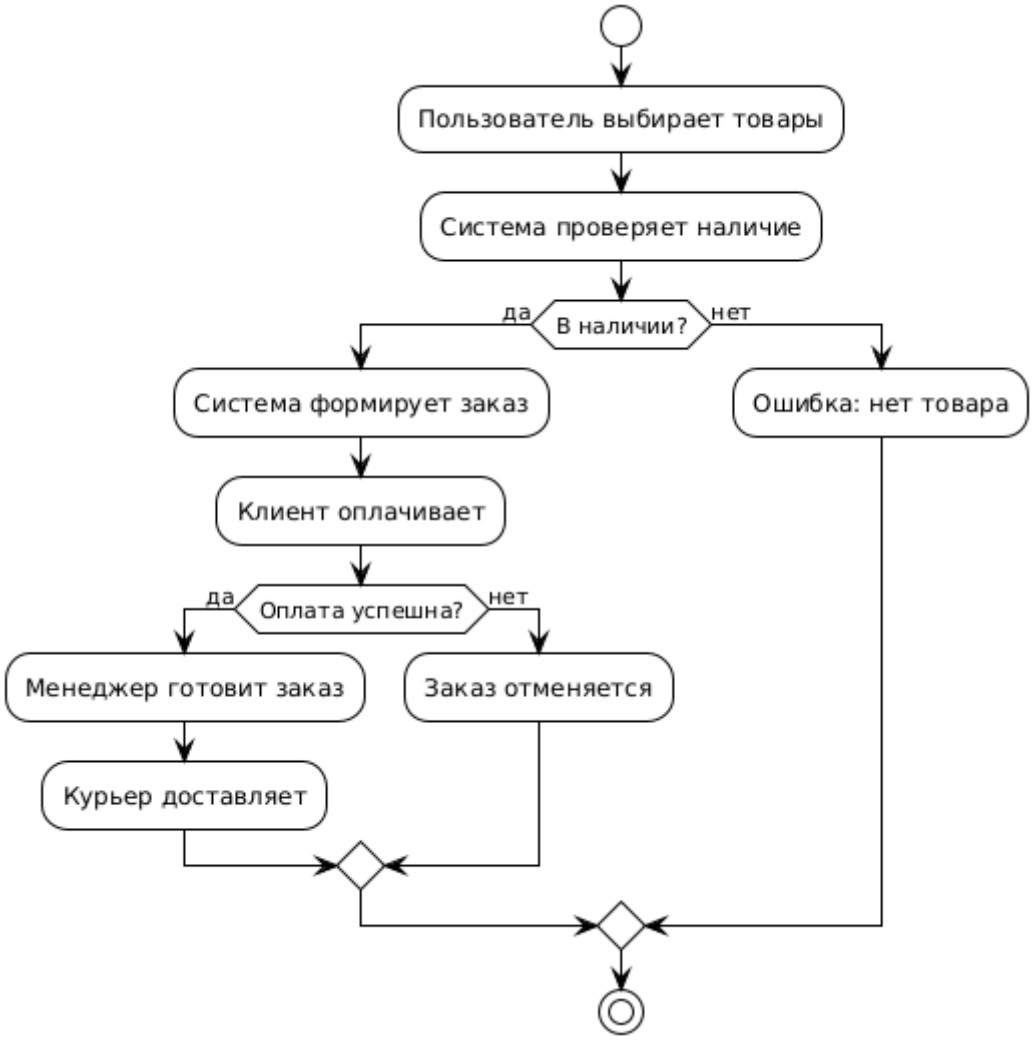
```
@startmindmap
* Структуры данных
** Заказ {id, дата, клиент, список товаров, сумма, статус}
** Товар {id, название, цена, количество}
** Оплата {id, метод, статус, заказ}
** Доставка {id, адрес, курьер, статус, заказ}
@endmindmap
```



b) Алгоритмы системы

```
@startuml
!theme plain
start
:Пользователь выбирает товары;
:Система проверяет наличие;
if (В наличии?) then (да)
  :Система формирует заказ;
  :Клиент оплачивает;
  if (Оплата успешна?) then (да)
    :Менеджер готовит заказ;
    :Курьер доставляет;
  else (нет)
```

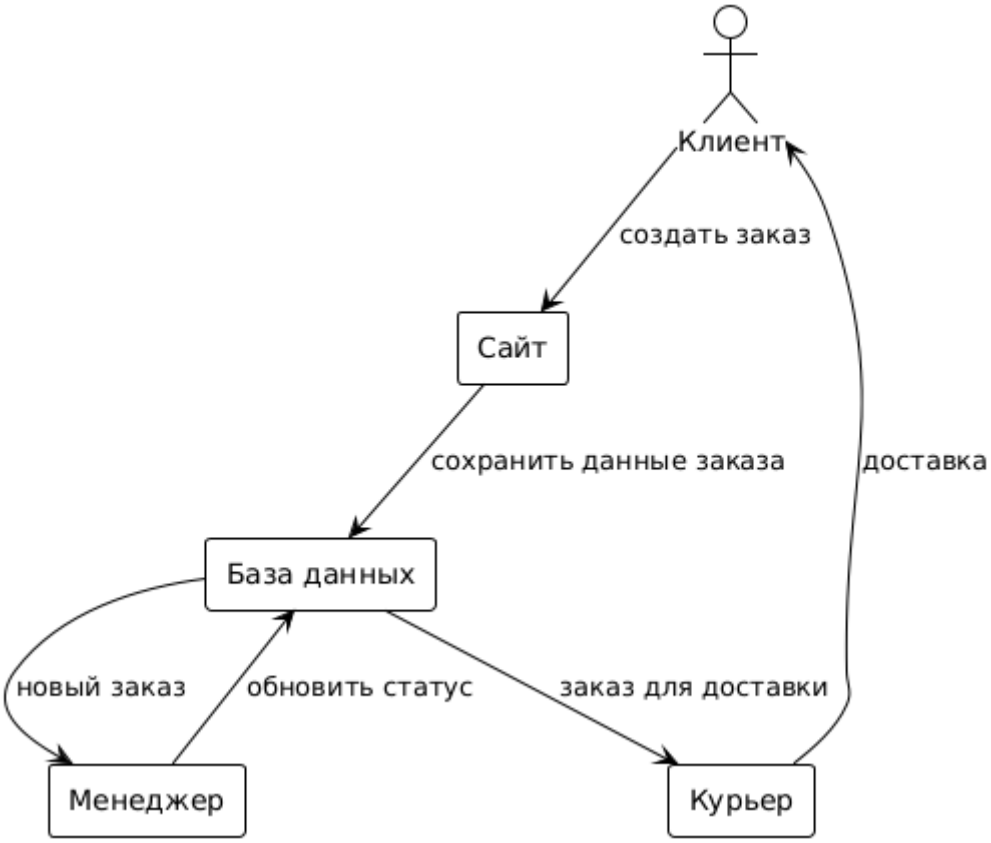
```
    :Заказ отменяется;
  endif
else (нет)
  :Ошибка: нет товара;
endif
stop
@enduml
```



c) Формальные условия и ограничения

4. Диаграмма потоков информации

```
@startuml
!theme plain

actor "Клиент" as Client
rectangle "Сайт" as Web
rectangle "База данных" as DB
rectangle "Менеджер" as Manager
rectangle "Курьер" as Courier

Client --> Web : "создать заказ"
Web --> DB : "сохранить данные заказа"
DB --> Manager : "новый заказ"
Manager --> DB : "обновить статус"
DB --> Courier : "заказ для доставки"
Courier --> Client : "доставка"
@enduml
```


Клиент
создать заказ
Сайт
сохранить данные заказа
доставка
База данных
новый заказ
обновить статус
заказ для доставки
Менеджер
Курьер