**Документоориентированная база данных (MongoDB):**

***Product service:*** *Коллекция: Products*  
{

"\_id": "ObjectId",

"subcategory\_id": "string",

"name": "string",

"image": "string",

"price": {

"city\_id\_1": "number",

"city\_id\_2": "number"

},

"inventory": {

"warehouse\_id\_1": "number",

"warehouse\_id\_2": "number"

},

"additional\_fields": {

"field1": "value1",

"field2": "value2"

}

}

Коллекция: Warehouses  
{

"\_id": "ObjectId",

“city\_id”: “number”

}

Коллекция: Subcategories  
{

"\_id": "ObjectId",

"category\_id": "string",

"name": "string",

"image": "string"

}

*Коллекция: Categories*{

"\_id": "ObjectId",

"group\_id": "string",

"name": "string",

"image": "string"

}

*Коллекция: Groups*  
{

"\_id": "ObjectId",

"name": "string",

"image": "string"

}

**Реляционная база данных (PostgreSQL)**

***User service:****Таблица: Users*  
Users (

id SERIAL PRIMARY KEY,

email VARCHAR(255) NOT NULL,

password VARCHAR(255) NOT NULL,

name VARCHAR(255),

city\_id INT,

registration\_date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);  
*Таблица: Cities*  
Cities (

id SERIAL PRIMARY KEY,

name VARCHAR(255) NOT NULL

);

***Order service:*** *Таблица: Orders*  
Orders (

id SERIAL PRIMARY KEY,

user\_id INT REFERENCES Users(id),

total\_price DECIMAL(10, 2),

status VARCHAR(50),

creation\_date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

*Таблица: OrderItems*  
OrderItems (

id SERIAL PRIMARY KEY,

order\_id INT REFERENCES Orders(id),

product\_id VARCHAR(255),

quantity INT,  
);

*Таблица: CartItems*  
CartItems (

id SERIAL PRIMARY KEY,

user\_id INT NOT NULL REFERENCES Users(id),

product\_id VARCHAR(255) NOT NULL,

quantity INT NOT NULL,

added\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

);

Такое разделение позволяет использовать сильные стороны каждой базы данных. MongoDB предоставляет гибкость в работе с товарами, где данные могут сильно варьироваться. PostgreSQL обеспечивает целостность и надежность данных для пользователей и заказов, где важны транзакции и строгие связи.

**Плюсы:**

1. **Гибкость структуры данных товаров:** MongoDB позволяет легко хранить и изменять структуру данных с различными полями для разных типов товаров.
2. **Четкое разделение обязанностей:** Реляционная база данных (PostgreSQL) будет использоваться для строгих структурированных данных, таких как пользователи и заказы, где важны транзакции и целостность данных.

**Минусы:**

1. **Сложность интеграции:** Необходимо управлять связями между данными, хранящимися в разных типах баз данных, что может усложнить разработку и поддержку.
2. **Повышенные затраты на администрирование:** Требуется поддержка и настройка двух различных систем баз данных.

Для хранения и управления медиаконтентом буду использовать облачное хранилище с CDN например, Amazon S3 и Amazon CloudFront, что обеспечивает масштабируемость, надежность и быструю доставку медиаконтента пользователям.