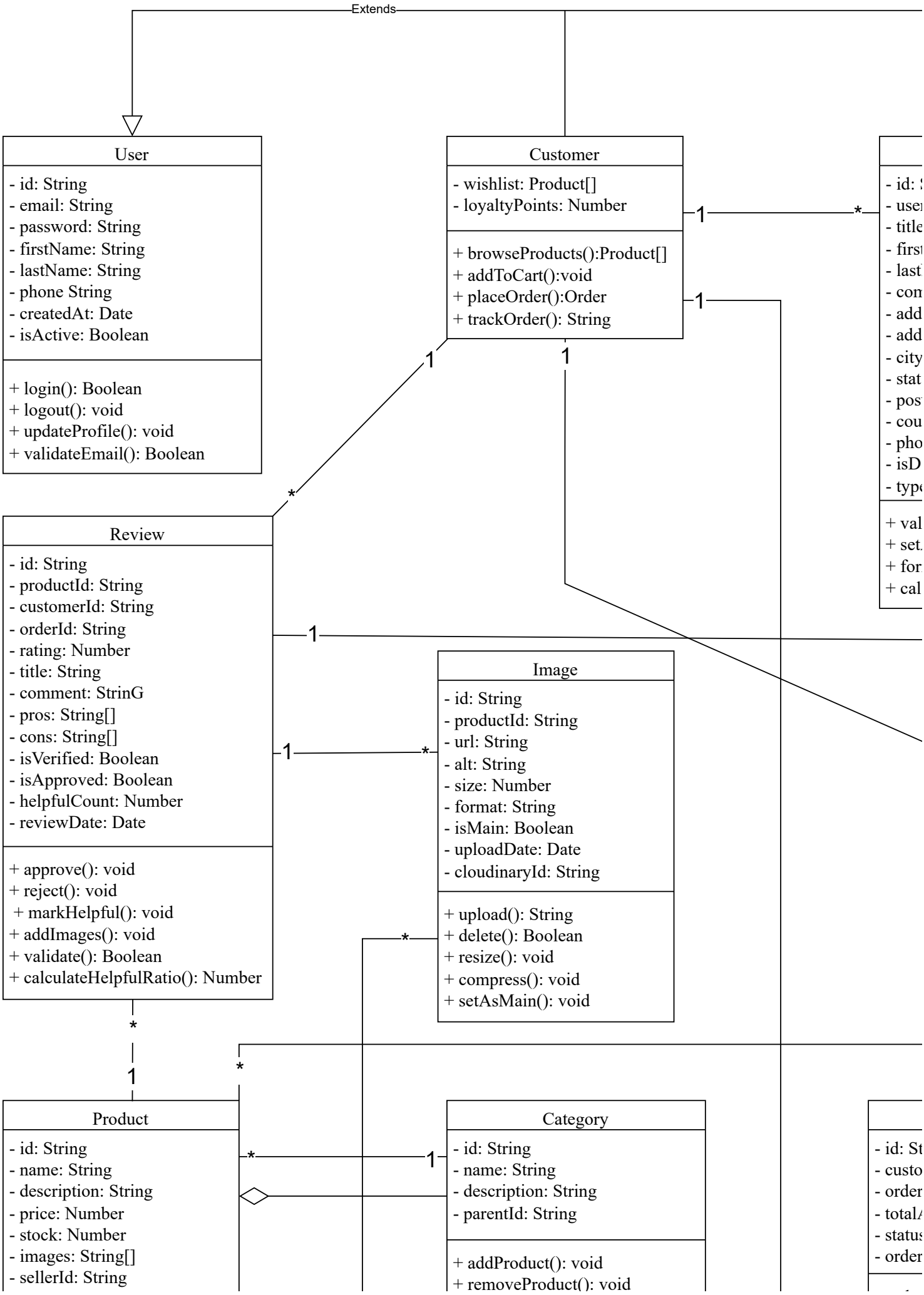
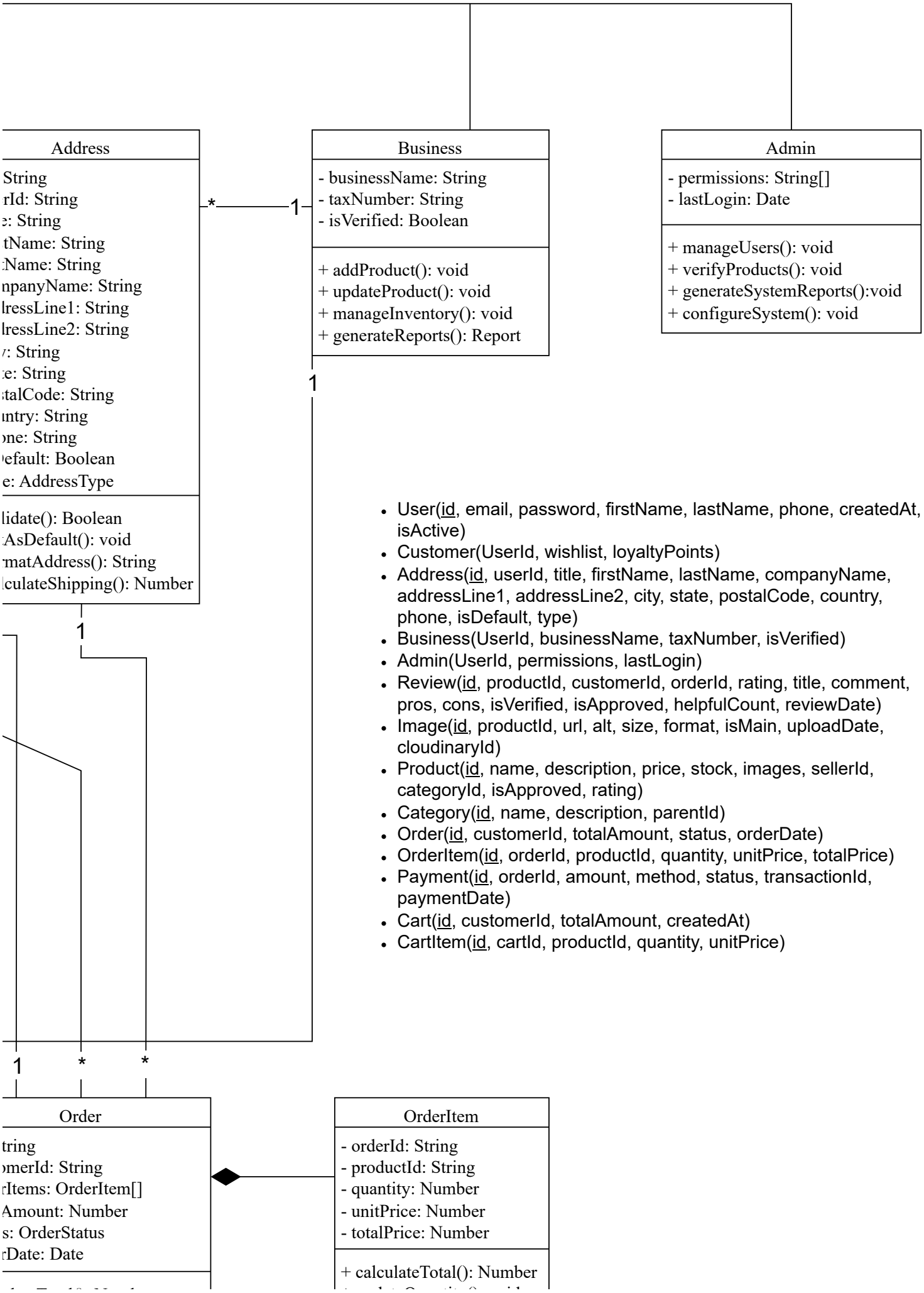
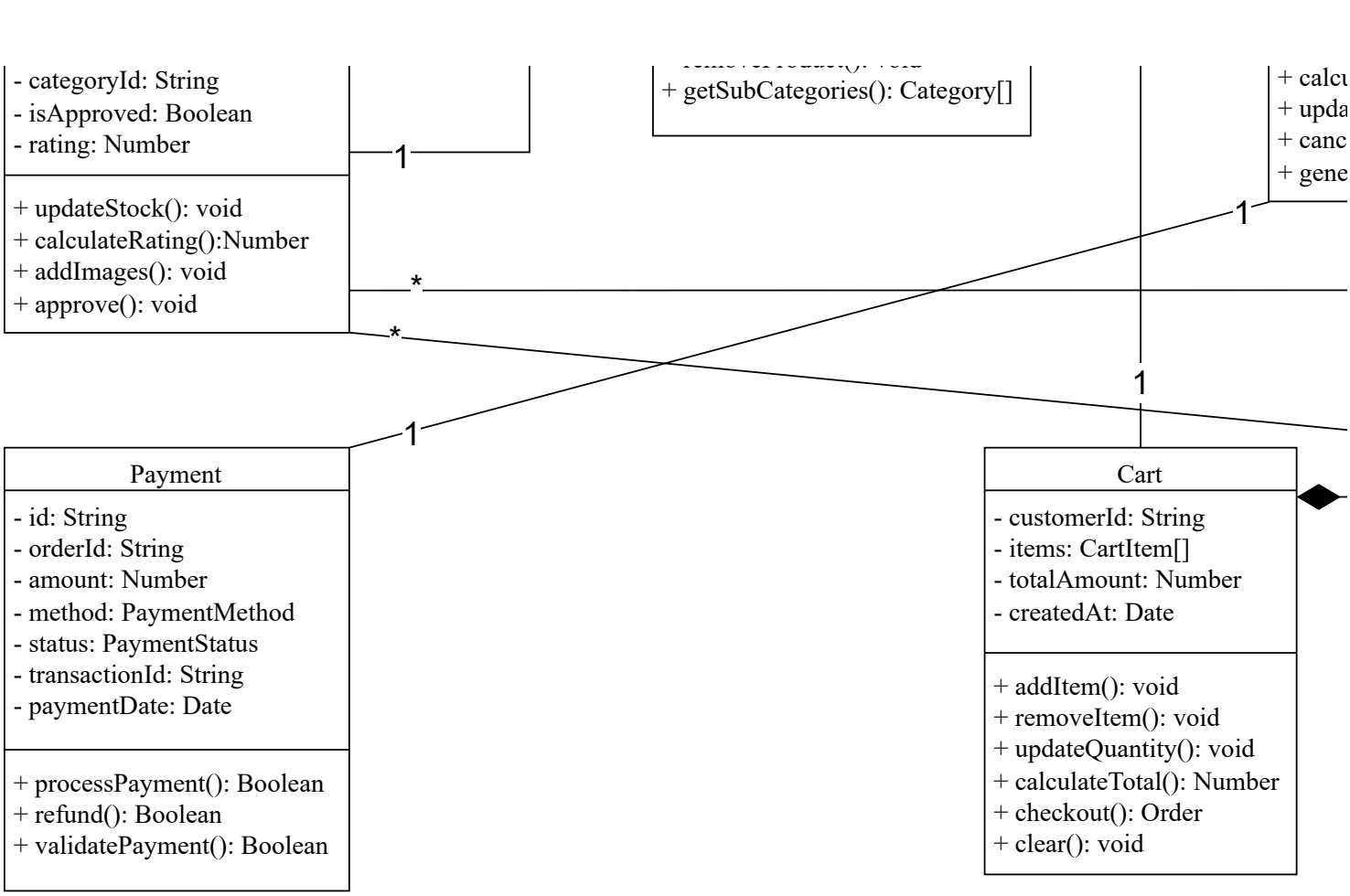
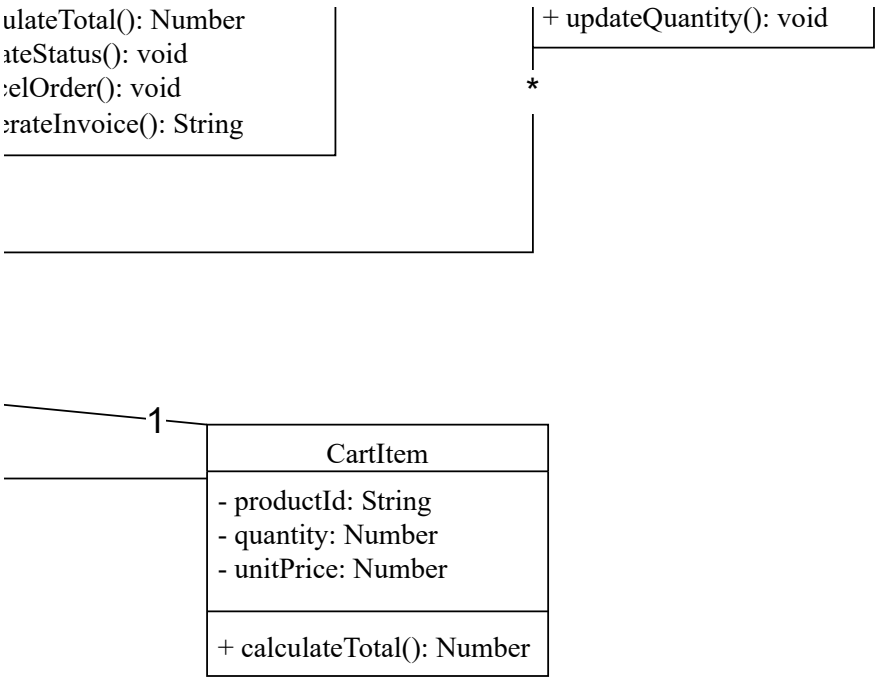

Extends
User
- id: String
- email: String
- password: String
- firstName: String
- lastName: String
- phone String
- createdAt: Date
- isActive: Boolean
+ login(): Boolean
+ logout(): void
+ updateProfile(): void
+ validateEmail(): Boolean
Customer
- wishlist: Product[]
- loyaltyPoints: Number
+ browseProducts():Product[]
+ addToCart():void
+ placeOrder():Order
+ trackOrder(): String
Review
- id: String
- productId: String
- customerId: String
- orderId: String
- rating: Number
- title: String
- comment: StrinG
- pros: String[]
- cons: String[]
- isVerified: Boolean
- isApproved: Boolean
- helpfulCount: Number
- reviewDate: Date
+ approve(): void
+ reject(): void
+ markHelpful(): void
+ addImages(): void
+ validate(): Boolean
+ calculateHelpfulRatio(): Number
Image
- id: String
- productId: String
- url: String
- alt: String
- size: Number
- format: String
- isMain: Boolean
- uploadDate: Date
- cloudinaryId: String
+ upload(): String
+ delete(): Boolean
+ resize(): void
+ compress(): void
+ setAsMain(): void
Product
- id: String
- name: String
- description: String
- price: Number
- stock: Number
- images: String[]
- sellerId: String
Category
- id: String
- name: String
- description: String
- parentId: String
+ addProduct(): void
+ removeProduct(): void
1
*



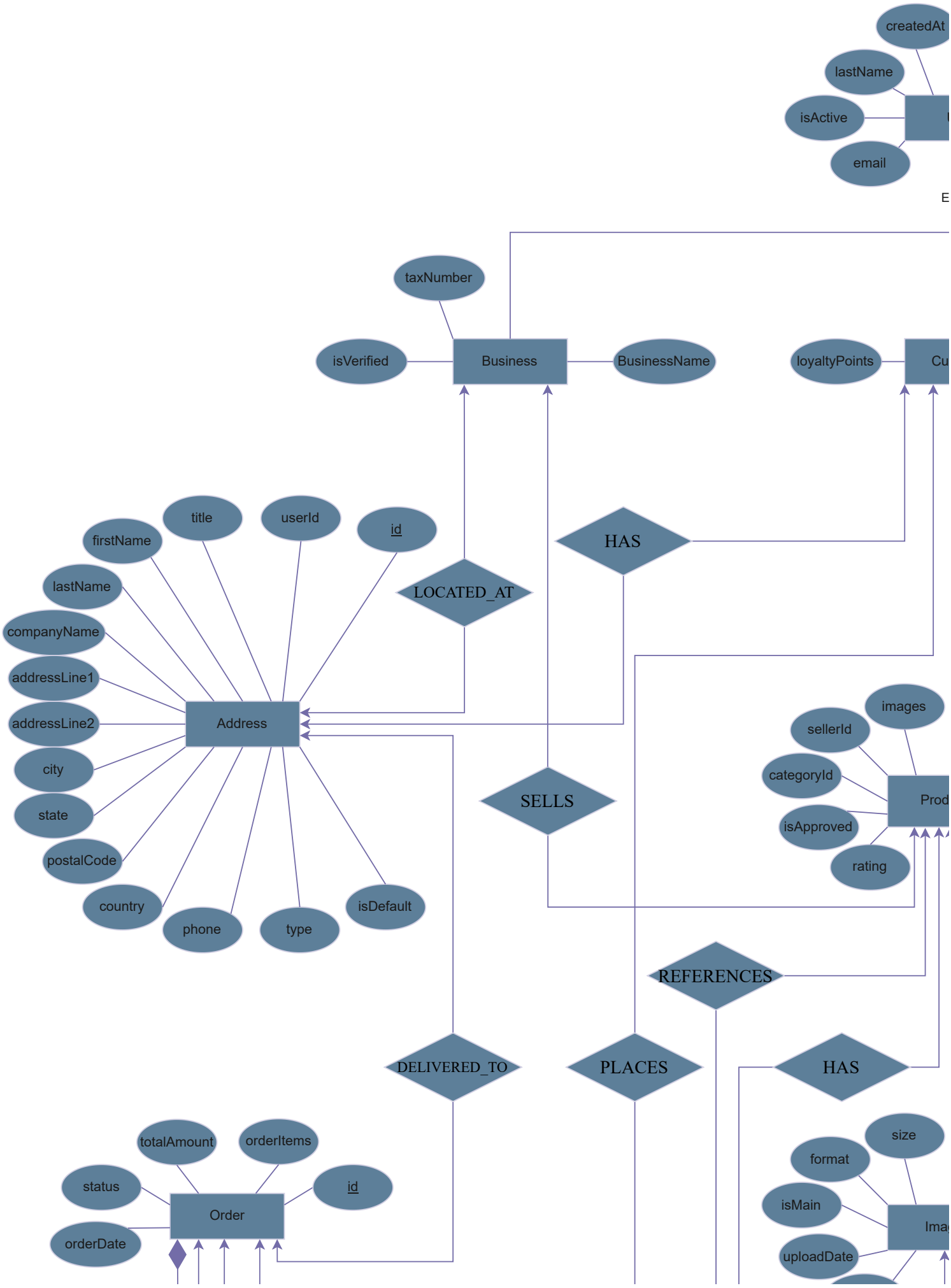
- User(id, email, password, firstName, lastName, phone, createdAt, isActive)
- Customer(UserId, wishlist, loyaltyPoints)
- Address(id, userId, title, firstName, lastName, companyName, addressLine1, addressLine2, city, state, postalCode, country, phone, isDefault, type)
- Business(UserId, businessName, taxNumber, isVerified)
- Admin(UserId, permissions, lastLogin)
- Review(id, productId, customerId, orderId, rating, title, comment, pros, cons, isVerified, isApproved, helpfulCount, reviewDate)
- Image(id, productId, url, alt, size, format, isMain, uploadDate, cloudinaryId)
- Product(id, name, description, price, stock, images, sellerId, categoryId, isApproved, rating)
- Category(id, name, description, parentId)
- Order(id, customerId, totalAmount, status, orderDate)
- OrderItem(id, orderId, productId, quantity, unitPrice, totalPrice)
- Payment(id, orderId, amount, method, status, transactionId, paymentDate)
- Cart(id, customerId, totalAmount, createdAt)
- CartItem(id, cartId, productId, quantity, unitPrice)


- categoryId: String
- isApproved: Boolean
- rating: Number
+ updateStock(): void
+ calculateRating():Number
+ addImages(): void
+ approve(): void
+ getSubCategories(): Category[]
+ calcu
+ upda
+ canc
+ gene
1
1
*
*
1
1
Payment
- id: String
- orderId: String
- amount: Number
- method: PaymentMethod
- status: PaymentStatus
- transactionId: String
- paymentDate: Date
+ processPayment(): Boolean
+ refund(): Boolean
+ validatePayment(): Boolean
Cart
- customerId: String
- items: CartItem[]
- totalAmount: Number
- createdAt: Date
+ addItem(): void
+ removeItem(): void
+ updateQuantity(): void
+ calculateTotal(): Number
+ checkout(): Order
+ clear(): void

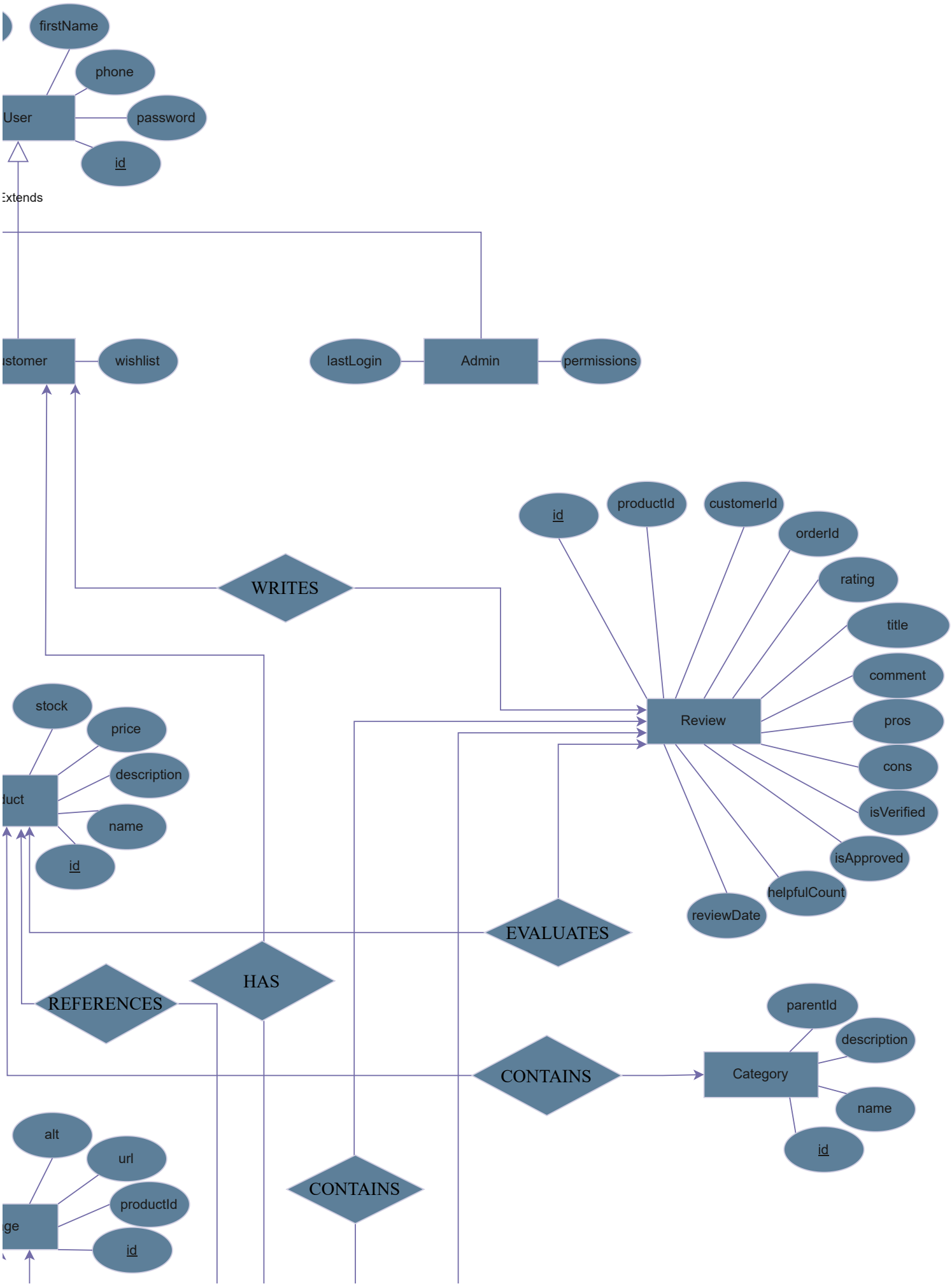

ulateTotal(): Number
ateStatus(): void
elOrder(): void
erateInvoice(): String
+ updateQuantity(): void
*

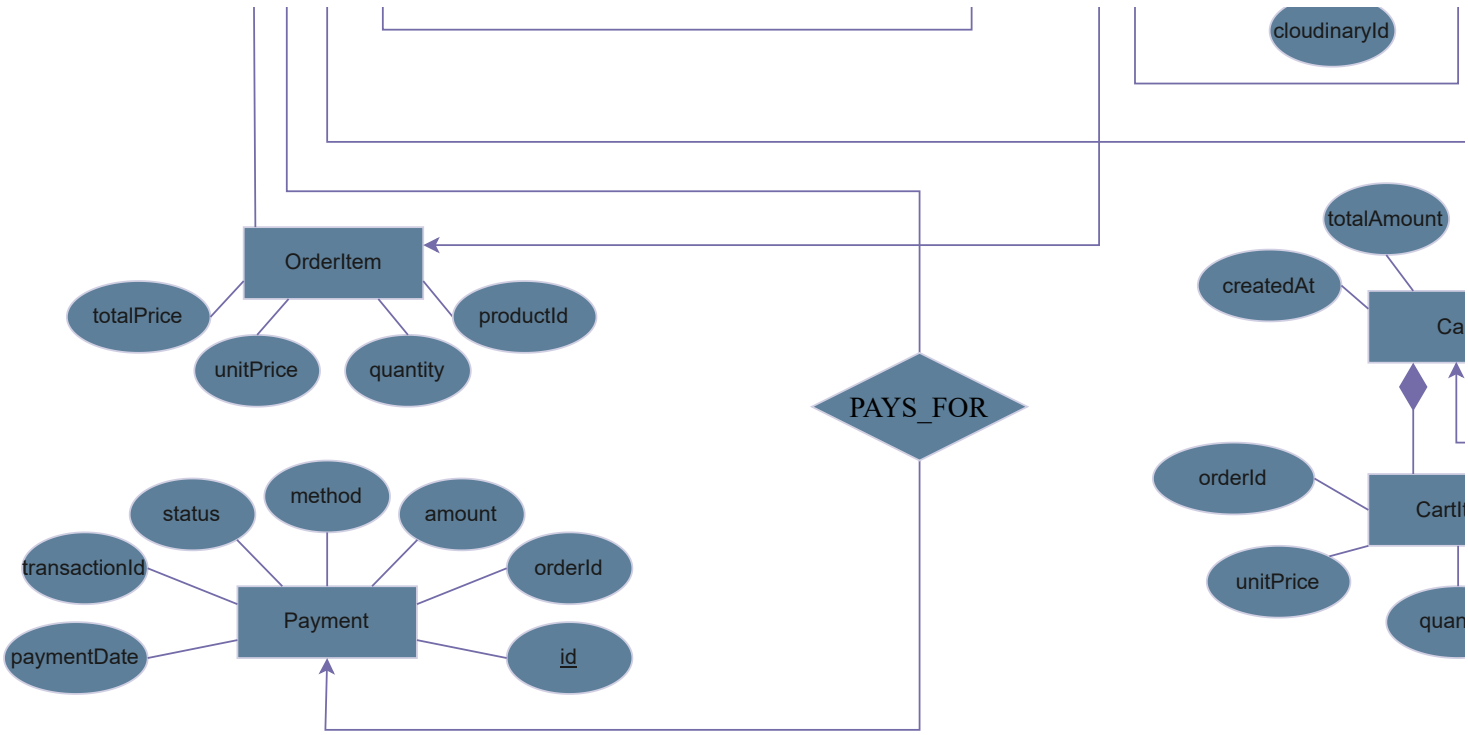
1
CartItem
- productId: String
- quantity: Number
- unitPrice: Number
+ calculateTotal(): Number

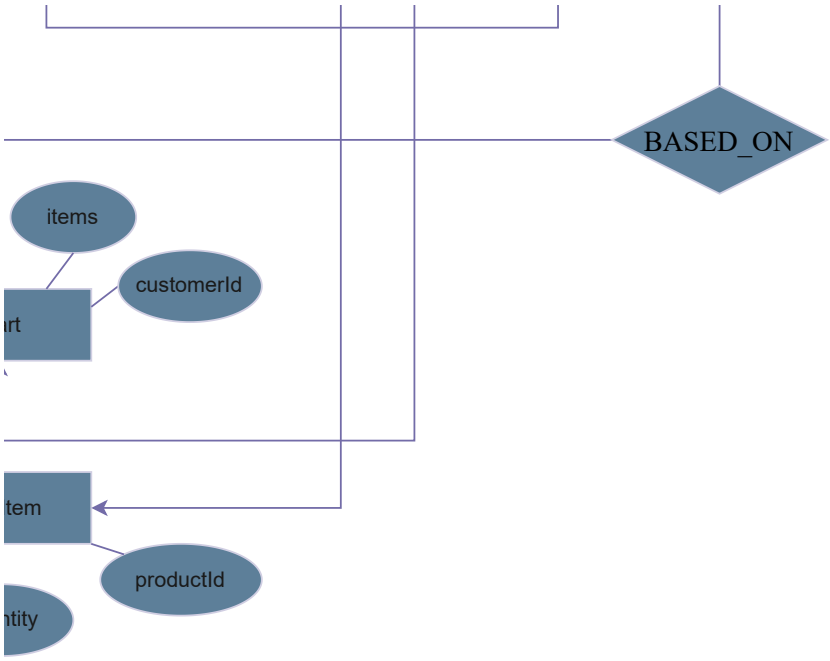
ERD (Entity Re



elationship Diagram)




cloudinaryId
totalAmount
createdAt
orderId
unitPrice
OrderItem
totalPrice
productId
unitPrice
quantity
PAYS_FOR
status
method
amount
transactionId
orderId
Payment
paymentDate
id


BASED_ON
items
customerId
productId

Reem:
Frontend: Next.js 14 + React + TypeScript + Tailwind CSS
Backend: Node.js + Express.js + Next.js API Routes
Veritabanı: MYSQL + Redis (cache)
Authentication: Payload CMS + JWT
Ödeme: Stripe + Webhooks +PAYTHOR
Hosting: Vercel + MYSQL

Mert:
Backend: Node.js + Express
Frontend: React
Database: PostgreSQL + Redis
Mimari: Mikroservis + Katmanlı

Başlıklar:

1. Kullanılacak diller / teknolojiler / framework / platform
2. Veri tabanı teknolojisi ve mimarisi
3. Nesnelerin mimarisi (ERD)
4. İş akış şemaları

Backend: Node.js + Express.js + Next.js API Route
Frontend: Next.js 14 + React + TypeScript + Tailwind C
Veritabanı: PostgreSQL + Redis (cache) + Cloudinary
Authentication: Payload CMS + JWT
Ödeme: Stripe + Webhooks + PAYTHOR
Hosting: Vercel

- **1. Kullanılan Diller / Teknolojiler / Framework / Platform**

  Büyük pazar yerleri, genellikle **çok dilli ve çoklu teknoloji** yaklaşımlarını benimserler.

  - **Backend (Arka Uç):**
    - **Java (Spring Boot ile):** Kurumsal seviye uygulamalar, mikroservisler ve yüksek performans gerektiren sistemler için çok yaygın kulla
    - **Kotlin:** Java Sanal Makinesi (JVM) üzerinde çalışan, daha modern ve geliştirici dostu bir dildir. Özellikle Android uygulama geliştirmec
    - **Go (Golang):** Yüksek eşzamanlılık (concurrency) ve performans gerektiren ağ servisleri, API ağ geçitleri, altyapı araçları ve veri akışı
    - **Python:** Veri analizi, makine öğrenimi, yapay zeka modelleri, otomasyon scriptleri ve bazı API'ler için tercih edilir.
    - **.NET (C#):** Bazı büyük şirketler hala Microsoft ekosistemindeki .NET framework'ü kullanabilir.
  - **Frontend (Ön Uç):**
    - **JavaScript / TypeScript:** Web uygulamalarının temelidir.
    - **React, Angular, Vue.js:** Modern tek sayfa uygulama (SPA) veya sunucu tarafında işlenen (SSR) web arayüzleri oluşturmak için kullar
    - **Native Mobil Geliştirme:**
      - **Kotlin / Java:** Android uygulamaları için.
      - **Swift / Objective-C:** iOS uygulamaları için.
  - **DevOps ve Altyapı:**
    - **Docker:** Uygulamaları konteynerleştirmek için.
    - **Kubernetes:** Konteynerli uygulamaları orkestre etmek ve ölçeklendirmek için.
    - **Bulut Sağlayıcıları:** Genellikle çoklu bulut stratejisi izlenir. Örneğin, **AliCloud (Alibaba Cloud)** Trendyol'un kullandığı bilinen bir bulut
    - **CI/CD Araçları:** (Continuous Integration/Continuous Deployment) Jenkins, GitLab CI/CD, GitHub Actions vb.
    - **İzleme ve Günlük Kayıt:** Prometheus, Grafana, ELK Stack (Elasticsearch, Logstash, Kibana), Splunk.
    - **CDN (İçerik Dağıtım Ağı):** Statik içeriklerin (resimler, videolar) hızlı yüklenmesi için. **Cloudflare** gibi hizmetler güvenlik ve performans

s
CSS

ınılır. Spring Boot, Java tabanlı mikroservis geliştirmeyi kolaylaştırır.

de ve bazı backend servislerinde tercih edilebilir. Trendyol'un kendi açık kaynak kodlu Kotlin kütüphaneleri de mevcut.

ı işleme (data streaming) için kullanılır. Trendyol'un bazı altyapı araçlarında Go kullandığı biliniyor.

ınılan popüler JavaScript kütüphaneleri/framework'leri. Trendyol'un **Baklava** adında kendi **React Native** tabanlı bir tasarım sistemi bulunuyor.

t sağlayıcısıdır. Diğer büyük oyuncular Amazon Web Services (AWS), Google Cloud Platform (GCP) veya Microsoft Azure kullanabilir.

s için sıkça kullanılır.

---

2. Veri Tabanı Teknolojisi ve Mimarisi

Büyük pazar yerleri, farklı veri türleri ve performans gereksinimleri için **poliglotsal (polyglot) kalıcılık** yaklaşımını benimserler; yani birden fazla verit

- **İlişkisel Veritabanları (SQL):**
  - **PostgreSQL, MySQL:** Genellikle ana iş verileri, kullanıcı bilgileri, sipariş detayları gibi yapısal ve ilişkisel veriler için kullanılır. Veri tuta
- **NoSQL Veritabanları:**
  - **Couchbase:** Yüksek performanslı veri erişimi ve ölçeklenebilirlik gerektiren durumlar için kullanılır. Trendyol'un altyapısında Couchba
  - **MongoDB:** Büyük hacimli, esnek şemalı doküman tabanlı veriler için (örn. ürün katalogları, kullanıcı profilleri, içerik).
  - **Redis / Memcached:** Hızlı önbellekleme (caching) ve oturum yönetimi için kullanılır. Performansı artırmak kritik önem taşır.
  - **Cassandra / HBase:** Genellikle büyük veri analitiği, zaman serisi verileri veya dağıtık, yüksek yazma/okuma performansına ihtiyaç du
- **Veri Ambarları (Data Warehouses) ve Veri Gölleri (Data Lakes):**
  - **Apache Kafka:** Büyük ölçekli olay akışı (event streaming) ve veri boru hatları (data pipelines) için kullanılır. Mikroservisler arası iletişi
  - **Apache Spark / Hadoop:** Büyük veri işleme ve analitiği için.
  - **Snowflake, Google BigQuery, AWS Redshift:** Analitik sorgular ve iş zekası için bulut tabanlı veri ambarları.
- **Mimari Yaklaşım:**
  - **Dağıtık Veritabanı Sistemleri:** Veriler farklı sunuculara veya bölgelere dağıtılır (sharding, replication).
  - **Çoklu Veritabanı Türleri:** Farklı iş yükleri için en uygun veritabanı seçimi yapılır (polyglot persistence).
  - **Mikroservis Mimarisi ile Veritabanı:** Her mikroservisin kendi veritabanına sahip olabileceği veya paylaşılan ancak mantıksal olarak a

---

3. Nesnelerin Mimarisi (ERD)

Büyük pazar yerleri, karmaşık ilişkileri ve veri modellerini yönetmek için detaylı **Varlık-İlişki Diyagramları (ERD)** ve geniş bir veri modelleme pratiği k

- **Çok Sayıda Varlık (Entity):** Kullanıcılar, Satıcılar, Ürünler, Kategoriler, Siparişler, Sipariş Kalemleri, Ödemeler, İadeler, Sepetler, Yorumlar, Ka
- **Karmaşık İlişkiler:** Bu varlıklar arasında bire-bir (1:1), bire-çok (1:N), çoka-çok (N:M) ilişkilerin yanı sıra, **kalıtım (inheritance)** veya **uzmanla**
- **Denormalizasyon Stratejileri:** Performans kaygıları nedeniyle, bazı durumlarda veriler normalizasyon kurallarına tam uymasa da (denormal
- **Mikroservis Bağımlı Veri Modeli:** Mikroservis mimarilerinde her servisin kendi veri modelinden sorumlu olması nedeniyle, tek bir devasa ER

---

4. İş Akış Şemaları

Büyük pazar yerlerinde, süreçlerin karmaşıklığı nedeniyle **kapsamlı iş akış şemaları** (iş süreçleri diyagramları, akış şemaları, BPMN diyagramları) h

- **Çok Kapsamlı Süreçler:**
  - **Ürün Listeleme Süreci:** Satıcının ürün yüklemesinden, ürünün onaylanmasına, kategoriye atanmasına kadar.
  - **Sipariş Süreci:** Müşterinin ürün eklemesi, ödeme yapması, siparişin onaylanması, stok kontrolü, satıcıya bildirim, kargo ve teslimat.
  - **İade/İptal Süreci:** Ürün iade talebi, onay, kargo, geri ödeme.
  - **Ödeme Akışları:** Farklı ödeme yöntemleri, güvenlik kontrolleri, banka entegrasyonları.
  - **Kullanıcı Kayıt/Giriş Akışları:** Hesap oluşturma, doğrulama, şifre sıfırlama, iki faktörlü kimlik doğrulama.
  - **Pazarlama ve Kampanya Yönetimi:** Kampanya oluşturma, indirim uygulama, kişiselleştirme.
- **Otomasyon ve Entegrasyon:** Çoğu iş akışı, farklı mikroservisler, harici API'ler (bankalar, kargo şirketleri) ve üçüncü taraf hizmetlerle entegra
- **Hata Yönetimi ve İstisnalar:** Akış şemaları, olası hata durumlarını (örn. ödeme başarısızlığı, stok yetersizliği) ve bu durumlarda sistemin nas
- **UML Aktivite Diyagramları / BPMN Diyagramları:** Bu tür karmaşık iş akışlarını modellemek için genellikle standartlaştırılmış diyagramlama

itabanı teknolojisini bir arada kullanırlar.

arlılığı ve işlemsel bütünlük kritik olduğunda tercih edilirler.

ıse kullandığı biliniyor.

uyan senaryolar için.

imin temelini oluşturabilir.

ayrılmış veritabanları kullanıldığı bir yapı.

---

kullanırlar.

ampanyalar, Kargo Bilgileri, Adresler, Bildirimler vb. gibi yüzlerce farklı varlık olabilir.

aşma/genelleşme (örneğin, bir `User`'ın `Customer`, `Seller` veya `Admin` olarak uzmanlaşması) gibi daha karmaşık yapılandırmalar da bulunur.

lizasyon) veritabanında tekrar edebilir veya birleştirilmiş bir şekilde tutulabilir. Özellikle NoSQL veritabanları bu esnekliği sunar.

RD yerine, her servise özgü daha küçük ERD'ler veya veri modelleri bulunur. Ancak bu servisler arası veri iletişimi ve tutarlılık büyük bir mimari zorluktur.

---

ayati öneme sahiptir.

asyonu gerektirir. İş akış şemaları, bu entegrasyon noktalarını ve veri akışını gösterir.

sıl tepki vereceğini de detaylandırır.

teknikleri kullanılır.