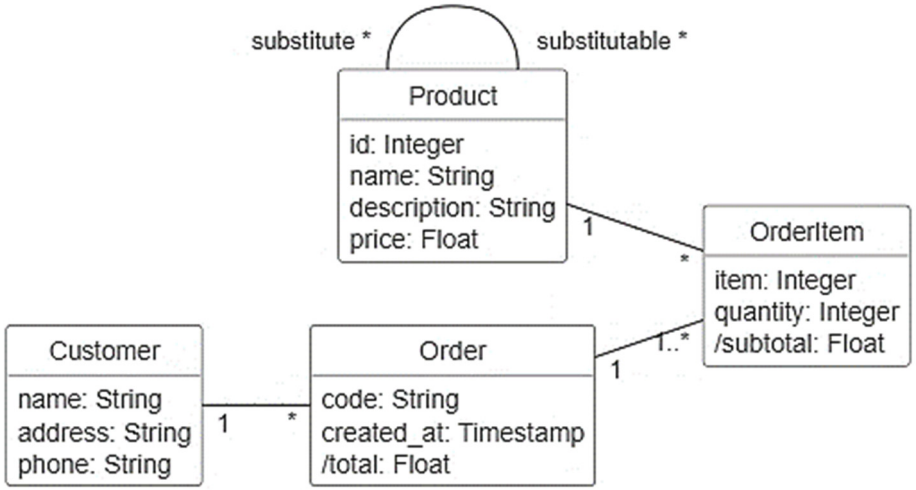
Les notes es publicaran dimarts 27 de juny de 2023 com a molt tard.

Exercici (7 punts)

Considereu un sistema de gestió de comandes (*orders*) de productes. Suposeu que els possibles clients (programes) dels serveis que dissenyareu ja disposen de les credencials de seguretat (autentificació i autorització) necessàries i que en aquest exercici no cal tenir en compte. El diagrama de classes del Domini és:



Identificadors: (Customer, phone), (Order, code), (Product, id), (OrderItem, Order.code+item)

Informació derivada: Order.total = Sum(Order.OrderItem.subtotal), OrderItem.subtotal = OrderItem.Product.price * OrderItem.quantity

Caldrà dissenyar els serveis que permetin implementar les 5 operacions següents:

1. Consultar tots els *products*, retornant els seus atributs. Hi ha d'haver la possibilitat d'obtenir només aquells que són *substitute* d'un *product* determinat i/o d'ordenar-los segon el *price* (ascendent o descendent).
2. Consultar la info (tots els atributs) d'un *order* concret. Per cada *orderItem*, cal incloure tots els atributs d'aquest i l'*id* del *product*. També cal incloure la info completa del *customer*.
3. Substituir el *product* d'un *orderItem* per un altre. Cal que el nou *product* sigui un *substitute* de l'anterior. En cas de canviar, s'ha de retornar el nou *subtotal*.
4. Eliminar un *orderItem*. Cal que a *l'order* li quedi un altre *orderItem* com a mínim.
5. Crear un *order* nou amb els seus *orderItems*. Per cada *orderIem* cal indicar la *quantity* del *product* corresponent. Si l'operació té èxit, el sistema genera o calcula els atributs *item* i *subtotal* de cada *orderItem*, així com tots els atributs del nou *order*, i retorna la mateixa info que l'operació 2.

Més concretament,

a) **[2 punts]** Dissenyeu un servei web SOAP que inclogui les **4 primeres** operacions. Heu d'utilitzar els patrons RPC API i DTO. Per a cada operació cal definir els paràmetres d'entrada i el resultat. Utilitzeu el mode d'interacció *Request/Response*.

b) **[1 punt]** Dissenyeu un servei web SOAP utilitzant el mode d'interacció *Request/Acknowledge/Poll* per a l'operació 5 (*order* nou). Heu d'utilitzar els patrons RPC API i DTO.

c) **[3 punts]** Dissenyeu un servei web REST per a les **5 operacions**. Per descriure cada operació caldrà omplir la plantilla que teniu a continuació:

Title	Operation name	
URI	Example: `/users` or `/users/:id`	
Query Params	`q=[String]`: required | opcional	
Method	`GET` | `POST` | `DELETE` | `PUT`	
Body Data	Example: `{"name": String, "email": ArrayOf(String)}`	

Returns	Success Response	Code and Content. Example:
		`200 OK`
		`{"id": String, "name": String, "email": ArrayOf(String)}`

	Error Responses	Code and Content. Example:
		`404 Not Found`
		`{message: "no user exists with that id"}`

d) **[1 punt]** Definiu l'escenari que utilitzi el servei web REST anterior i que permeti substituir cada *product* de l'*order* amb *code* = RW4687 pel seu substitut amb el *price* inferior més baix, en cas d'existir.

a) Servei web SOAP per a les 4 primeres operacions

Operation name	Input	Output
getProducts	getProductsReq = <substitute_for: Integer [0..1], order_by: {price_asc, price_desc} [0..1] >	getProductsRes= Set(ProductInfo) ProductInfo = <id: Integer, name: String, description: level, price: Float>
getOrder	getOrderReq= <code : String>	getOrderRes = <code: String, created_at: Timestamp, total: Float, order_items: OrderItemInfo [1..*], customer: CustomerInfo > OrderItemInfo = <item: Integer, quantity: Integer, subtotal: Float, product_id: Integer> CustomerInfo = <name: String, address: String, phone: String >
substituteProduct	substituteProductReq= <order_code: String, order_item: Integer, new_product_id: Integer>	substituteProductRes= <code: Integer, message: String, new_subtotal: Float [0…1] >
destroyOrderItem	destroyOrderItemReq = <order_code: String, order_item: Integer >	destroyOrderItemRes= <code: Integer, message: String >

b) Servei web SOAP utilitzant el mode d'interacció Request/Acknowledge/Poll per a l'operació de crear un nou order.

Operation name	Input	Output
createOrder	createOrderReq= < new_items: NewItem [1..*], customer_phone: String > NewItem: < quantity: Integer, product_id: Integer >	createOrderAck= <id: String>
getResults	createOrderAck= <id: String>	createOrderRes= <code: Integer, message: String order: getOrderRes [0..1]>

c) Api REST per a les 5 operacions

Title	Get products	
URI	/api/products	
Query Params	substitute_for=[Integer]: optional order_by={price_asc, price_desc}: optional	
Method	GET	
Body Data		
Returns	Success Response	200 OK [{ “id”: 3447, “name”: “Prisoner’s Dilemma”, “description”: “Some boring stuff”, “price” : 25 }, { … }, …]
	Error Response	400 Bad Request {“message”: “Invalid order_by option”}

Title	Get an order	
URI	/api/orders/:code	
Query Params		
Method	GET	
Body Data		
Returns	Success Response	200 OK { “code”: “ERT34W”, “created_at”: “2023-06-20T08:30”, “total”: 103, “order_items” : [{“item”: 1 , “quantity”: 2, “subtotal”: 50, “product_id”: 3447} , { … }, …], “customer”: { “name”: “Andrew Tate”, “address”: “Some creepy Romanian jail”, “phone”: “34 648 224 987” } }
	Error Response	404 Not Found {“message”: “There is no order with such a code"}

Title	Substitute the product of a given order item	
URI	/api/orders/:code/order_items/:item	
Query Params		
Method	PUT	
Body Data	{ “new_product_id”: 4657 }	
Returns	Success Response	200 OK {“item”: 1 , “quantity”: 2, “subtotal”: 48, “product_id”: 4657 }
	Error Response	400 Bad Request {“message”: “Invalid new product id”} 404 Not Found {“message”: “No item for such an order"} 409 Conflict {“message”: “The new product cannot substitute the old one"}

Title	Destroy a given order item	
URI	/api/orders/:code/order_items/:item	
Query Params		
Method	DELETE	
Body Data		
Returns	Success Response	204 No Content
	Error Response	404 Not Found {“message”: “No item for such an order"} 409 Conflict {“message”: “The order cannot be left with no order items"}

Title	Create a new order	
URI	/api/orders	
Query Params		
Method	POST	
Body Data	{ “new_items” : [{“quantity”: 2, “product_id”: 3447} , { … }, …], “customer_phone”: “34 648 224 987” }	
Returns	Success Response	201 Created { “code”: “ERT34W”, “created_at”: “2023-06-20T08:30”, “total”: 103, “order_items” : [{“item”: 1 , “quantity”: 2, “subtotal”: 50, “product_id”: 3447} , { … }, …], “customer”: { “name”: “Andrew Tate”, “address”: “Some creepy Romanian jail”, “phone”: “34 648 224 987” } }
	Error Response	400 Bad Request {“message”: “Invalid customer phone"}

d) Client

1. Faig la crida

```
GET /api/orders/RW4687
```

2. La crida em retorna una resposta **O**. Per cada order item **I** ∈ **O**.order_items faig

```
GET /api/products?substitute_for=I.product_id&order_by=price_asc
```

3. Si la resposta és un array no buit el primer element **P** del qual compleix que **P**.price < **I**.subtotal / **I**.quantity, llavors faig la crida

```
PUT /api/orders/RW4687/order_items/I.item

{ “new_product_id”: P.id }
```