Les notes d'aquest examen es publicaran dimarts 18 de juny de 2019.
Les revisions es faran dijous 20 de juny d'11:30 a 13 al despatx Omega-133.

Pregunta 1 (1,5 punts): Expliqueu mitjançant un exemple com es pot aplicar el patró de disseny *Request/Acknowledge*, variant *Poll*, amb REST.

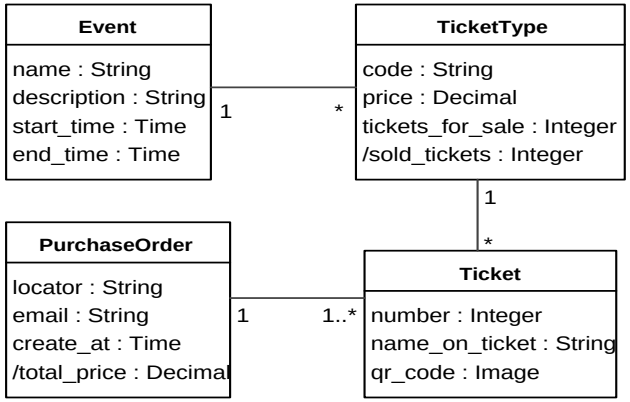
Pregunta 2 (1,5 punts): Responeu **una sola** de les preguntes següents, i que correspongui a un tema **no presentat per vosaltres**:

a) Expliqueu 3 característiques rellevants dels Microserveis.

b) Citeu 3 punts febles de les *Serverless Architectures*.

Exercici (7 punts)

Considereu un sistema de venda de *tickets* per a *events*. Suposeu que els possibles clients (programes) dels serveis que dissenyareu ho faran des d'una intranet segura i que, per tant, no cal considerar el tema de l'autenticació/ autorització. El diagrama de classes que en descriu el domini és el següent:



Identificadors: (Event, name), (TicketType, event.name+code), (PurchaseOrder, locator), (Ticket, purchaseOrder.locator+number)

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

1. Consultar la info (tots els atributs) d'un *PurchaseOrder* concret. Caldrà incloure també els atributs de tots els *tickets* que en formen part, així com també el price i el code del *TicketType* i el name i l'start_time de l'*Event* corresponents.

2. Consultar els *events* que comencen després d'una data determinada i obligatòria. Hi ha d'haver també la possibilitat de filtrar-los perquè acabin abans d'una certa data. Per cada *Event*, cal retornar tots els seus atributs així com els dels seus *TicketTypes*. Els *events* es retornaran ordenats per start_time asc.

3. Eliminar un determinat *TicketType*. Només es pot fer si sold_tickets == 0.

4. Canviar el name_on_ticket d'un *Ticket* determinat. Es pot fer fins a 24 hores abans que comenci l'*Event*. El sistema ha de regenerar el qr_code.

5. Crear una *PurchaseOrder* amb els seus *Tickets* corresponents. Les dades que s'han de proporcionar són: l'email del comprador i una llista de name_on_tickets amb la referència al *TicketType* que es vol per cadascun. L'operació només tindrà èxit si hi ha disponibilitat en els *TicketType* que es demanen. En aquest cas, el sistema retornarà la mateixa info que l'operació **1**. Els atributs locator, total_price i create_at de *PurchaseOrder* i els number i qr_code dels *Tickets* són generats pel sistema. Un *TicketType* té disponibilitat mentre tickets_for_sale > sold_tickets & event.start_time > (System.current_time + 24h).

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**. 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	(Only for POST and UPDATE requests) 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 comprar per a la "Mavis Batey", la "Jane Fawcett" i la "Jean Valentine" els *tickets* més cars disponibles per al primer *Event* que comenci després de l'11 de setembre de 2019.

a) Servei web SOAP per a les 4 primeres operacions

Operation name	Input	Output
getPOByLocator	getPOByLocatorReq= <locator: String>	getPOByLocatorRes= <locator: String, email: String, create_at: Time total_price: Decimal, tickets: Set(TicketInfo)> where TicketInfo = <number: Integer, name_on_ticket: String, qur_code: Image, tt_code: String, tt_price: Decimal, event_name: String event_start_time: Time>
getEvents	getEventsReq= <since_date: Date, until_date: Date [0..1]>	getEventsRes= Set(EventInfo) where EventInfo= <name: String, description: String, start_time: Time, end_time: Time, ticket_types: Set(TicketType)> where TicketType= <code: String price: Decimal, tickets_for_sale: Integer sold_tickets: Integer>
deleteTicketType	deleteTicketTypeReq= <event_name: String, code: Time>	deleteTicketTypeRes= <code: Integer, message: String>
changeTicket	changeTicketReq= <po_locator: String number: Integer new_name_on_ticket: String>	changeTicketRes= <code: Integer, message: String new_qr_code: Image [0..1]>

b) Servei web SOAP utilitzant el mode d'interacció Request/Acknowledge/Poll per a l'operació de reprogramar una Lesson:

Operation name	Input	Output
createPO	createPOReq= <email: String, tickets_for: Set(TicketReq)> where TicketReq = <name_on_ticket: String, tt_event_name: String tt_code: String>	createPOAck= <id: String>
getResults	createPOAck = <id: String>	createPORes= <code: Integer, message: String, getPOByLocatorRes [0..1]>

c) Api REST per a les 5 operacions

Title	Consultar la info d'un PurchaseOrder concret	
URI	/api/purchase_orders/:locator	
Query Params		
Method	GET	
Body Data		
Returns	Success Response	200 OK { "locator:" "LMER45S", "email": "espe@ppm.com", "create_at": "2019-06-11T10:30:00.000Z", "total_price": 143.50, "tickets" : [{"number": 1, "name_on_ticket": "M. Rajoy", "qr_code": "...ggg==", "tt_code": "VIP", "tt_price": 50.00, "event_name": "Wildest Party 2019", "event_start_time": "2019-08-11T22:30:00.000Z"}, ...] }
	Error Response	404 Not Found {message: "no PurchaseOrder with this locator"}

Title	Consultar events	
URI	/api/events	
Query Params	since_date=[String]: required until_date=[String]: optional	
Method	GET	
Body Data		
Returns	Success Response	200 OK [{"name": "Wildest Party 2019", "description": "The wildest party ever", "start_time": "2019-08-11T22:30:00.000Z", "end_time": "2019-08-13T06:30:00.000Z", "ticket_types": [{"code": "VIP", "price": 50.00, "tickets_for_sale": 100, "sold_tickets": 54}, ...] }, ...]
	Error Response	400 Bad Request {message: "incorrect parameters"}

Title	Eliminar un TicketType concret	
URI	/api/events/:name/ticket_types/:code	
Query Params		
Method	DELETE	
Body Data		
Returns	Success Response	204 No content
	Error Response	404 Not Found {message: “no ticket_type for this event name and code"} 409 Conflict {message: “this ticket type has some tickets sold”}

Title	Canviar el name_on_ticket d’un ticket	
URI	/api/purchase_orders/:locator/tickets/:number	
Query Params		
Method	PUT	
Body Data	{ “new_name_on_ticket”: “P. Married”}	
Returns	Success Response	200 OK {“number”: 1, “name_on_ticket”: “P. Married”, “qr_code”: “...ggg==”, “tt_code”: “VIP”, “tt_price”: 50.00, “event_name”: “Wildest Party 2019”, “event_start_time”: “2019-08-11T22:30:00.000Z”}
	Error Response	404 Not Found {message: “no ticket for this po locator and number"} 400 Bad request {message: “missing parameter new_name_on_ticket"} 409 Conflict {message: “it is too late for changes in the ticket”}

Title	Crear un PurchaseOrder	
URI	/api/purchase_orders	
Query Params		
Method	POST	
Body Data	{ “email”: ““espe@ppm.com”, “tickets_for”: [{ “name_on_ticket”: “M. Rajoy”, “tt_event_name”: “Quietest Party 2019”, “tt_code”: “OG”}] }	
Returns	Success Response	201 Created { “locator:” “PRET99Q”, “email”: “espe@ppm.com”, “create_at”: “2019-08-11T11:32:00.000Z”, “total_price”: 50.00, “tickets” : [{“number”: 45, “name_on_ticket”: “M. Rajoy”, “qr_code”: “...mhu==”, “tt_code”: “OG”, “tt_price”: 50.00, “event_name”: “Quietest Party 2019”, “event_start_time”: “2019-09-21T12:00:00.000Z”}] }
	Error Response	400 Bad request {message: “wrong parameters"} 409 Conflict {message: “Sorry, there is no availability for all the requested tickets”}

d) Client

1. Obtinc els events que comencen a partir de 2019-09-11:

GET /api/events?since_date=2019-09-11

2. Agafo el primer event de la llista, amb name = “SE 2019” i inspecciono els seus TicketTypes. El TicketType amb code = “Premium” és el més car i encara té disponibilitat per als 3 tickets que necessito. Per tant, faig la crida següent:

POST /api/puchase_orders
{ “email”: “farre@essi.upc.edu”,
“tickets_for”:
[{ “name_on_ticket”: “Mavis Batey”, “tt_event_name”: “SE 2019”, “tt_code”: Premium”},
{“name_on_ticket”: “Jane Fawcett”, “tt_event_name”: “SE 2019”, “tt_code”: “Premium”},
{“name_on_ticket”: “Jean Valentine”,“tt_event_name”: “SE 2019”,“tt_code”: “Premium”}
] }