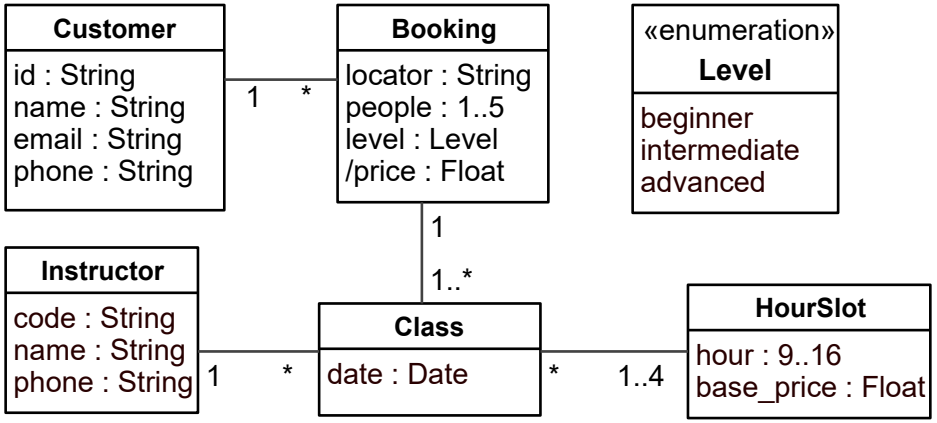
Les notes es publicaran dimarts 24 de gener de 2023 com a molt tard.

Exercici (7 punts)

Considereu un sistema de reserves (*bookings*) de classes d'esquí. 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, id), (Booking, locator), (Instructor, code), (Class, Booking.locator+date), (HourSlot, hour)

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

1. Consultar la info (tots els atributs) dels *bookings* que tenen una *class* en una data determinada. Cal retornar també tota la info del *customers* que les han fet.
2. Consultar la info (tots els atributs) d'un instructor concret. Opcionalment, es pot passar una data com a paràmetre addicional. En aquest cas es retornen també les hores dels *hour slots* que aquell instructor té disponibles per aquella data.
3. Modificar l'instructor d'una *class*. Cal que el nou instructor estigui disponible i que System.date < date de la *class*.
4. Cancel·lar (eliminar) una *class*. Cal que System.date < date de la *class*.
5. Fer un *booking* nou. S'ha d'indicar també quines classes es volen fer tal que System.date < date de la primera (més recent) *class* de totes. Per a cada *class*, s'ha de dir necessàriament els *hour slots* que s'usaran, que han de ser d'hores consecutives. Opcionalment, es pot indicar quin instructor es vol (el mateix per a totes les classes). El sistema intentarà assignar-lo a cada *class* si es troba disponible per aquella data i *hour slot*. Altrament, n'assignarà un altre que sí que ho estigui. En el cas que hi hagi un o més *hour slots* per alguna de les dates demanades en què no hi hagi cap instructor disponible, el *booking* no es podrà fer. En cas contrari, el sistema retornarà la info completa del *booking*, incloent-hi el name i code de l'instructor assignat a cada classe. Els atributs locator i price són generats pel sistema.

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 (*booking* 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 al *customer* amb id = "40999056Y" fer un *booking* per a una sola classe el dia 4 de febrer de 2023, per a dues persones de nivell intermediate. Es vol que l'instructor sigui code = "MADTRX" i que es faci durant els dos primers *hour slots* consecutius que aquest instructor tingui disponibles aquell dia.

a) Servei web SOAP per a les 4 primeres operacions

Operation name	Input	Output
getBookings	getBookingsReq = <date: Date>	getBookingsRes= Set(BookingInfo) BookingInfo = <locator: String, people: 1..5, level: level, price: Float, customer: CustomerInfo> CustomerInfo = <id:, String, name: String, email: String, phone: String>
getInstructor	getInstructorReq= <code : String, availability_on: Date [0..1]>	getInstructoResr = <code: String, name: String, phone: EstatV, availabilty: Availability [0..1]> Availability= 9..16[*]
changeClassInstructor	changeClassInstructorReq= <locator: String, date: Date, instructor_code: String>	changeClassInstructorRes= <code: Integer, message: String >
cancelClass	cancelClassReq = <locator: String, date: Date >	cancelClassRes= <code: Integer, message: String >

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

Operation name	Input	Output
createBooking	createBookingReq= <customer_id: String, people: 1..5, level: Level, classes: ClassReq [1..*], instructor_code: String [0..1]> ClassReq: < date: Date, hours: 9..16[1..4]>	createBookingAck= <id: String>
getResults	createBookingtAck= <id: String>	createBookingRes= <code: Integer, message: String booking: NewBooking[0..1]> NewBooking = < booking_info: BookinfInfo, class_instructors: ClassInstructor [1..*] > ClassInstructor= < date: Date instructor_code: String instructor_name: String >

c) Api REST per a les 5 operacions

Title	Get the bookings with a classe on a give date	
URI	/api/bookings	
Query Params	date=[String]: required	
Method	GET	
Body Data		
Returns	Success Response	200 OK [{ “locator”: “ERT34W”, “people”: 1, “level”: “beginner”, “price”: 103.5, “customer”: { “id”: “40889056X”, “name”: “Andrew Tate”, “email”: smalldickenergy@getalife.com”, “phone”: “34 648 224 987”}}, { … }, …]
	Error Response	400 Bad Request {“message”: “Invalid data format”}

Title	Get the info for a given Instructor	
URI	/api/instructors/:code	
Query Params	availability_on=[String]: optional	
Method	GET	
Body Data		
Returns	Success Response	200 OK { “code”: “MADTRX”, “name”: “Amadeu Estruch”, “phone”: “34 689 542 698”, “availability”: [9, 11, 12 , 16] }
	Error Response	400 Bad Request {“message”: “Invalid data format”} 404 Not Found {“message”: “There is no instructor with such a code"}

Title	Change class instructor	
URI	/api/bookings/:locator/classes/:date	
Query Params		
Method	PUT	
Body Data	{ “instructor_code”: “KILLBILL” }	
Returns	Success Response	200 OK { “locator”: “ERT34W”, “date”: “2023-01-15”, “instructor_code”: “KILLBILL”, “hours”: [10, 11] }
	Error Response	400 Bad Request {“message”: “Invalid instructor code”} 404 Not Found {“message”: “No class for the provided locator and date"} 409 Conflict {“message”: “The instructor is not available"}

Title	Class cancelation	
URI	/api/bookings/:locator/classes/:date	
Query Params		
Method	DELETE	
Body Data		
Returns	Success Response	204 No Content
	Error Response	404 Not Found {“message”: “No class for the provided locator and date"} 409 Conflict {“message”: “Past classes cannot be canceled"}

Title	Create a new booking	
URI	/api/bookings	
Query Params		
Method	POST	
Body Data	{ “customer_id”: “40889056X”, “people”: 1, “level”: “beginner”, “classes”: [{“date”: “2023-01-15”, “hours”: [10, 11] }, { … }, …], “instructor_code”: “KILLBILL” }	
Returns	Success Response	201 Created { “locator”: “ERT34W”, “customer_id”: “40889056X”, “people”: 1, “level”: “begginer”, “price”: 666.6, “class_instructors” : [{“date”: “2023-01-15”, “instructor_code”: “KILLBILL”, “instructor_name” : “Killian Bilbao”}, { … }, …]
	Error Response	400 Bad Request {“message”: “hours out of range"} 409 Conflict {“message”: “No available instructors for each class requested"}

d) Client

1. Faig la crida

GET /api/instructors/MADTRX?availability_on=2023-02-04

2. Agafo de la resposta les dues primeres hores disponibles consecutives, h i h+1 i faig

POST /api/bookings

{ “customer_id”: “40889056X”,
“people”: 2,
“level”: “intermediate”,
“classes”: [{“date”: “2023-02-04”, “hours”: [h, h+1] }],
“instructor_code”: “MADTRX” }