

# Enginyeria del Programari de Components i Sistemes Distribuïts

## SEGONA PROVA PRÀCTICA – PRAC 2

### Presentació

A la PRAC 1 s'ha dissenyat una solució per al cas **Photo&Film4You** basada en microserveis, usant models de representació i patrons de disseny. En aquesta PRAC 2 es demana implementar un subconjunt d'aquest disseny a partir de programar 3 microserveis considerant diferents aspectes, com, per exemple: estat propi, comunicació síncrona, lògica de negoci, consistència de les dades i desplegament en local.

Per elaborar aquesta pràctica es formulen 2 exercicis a partir del cas **Photo&Film4You**:

1. Implementar tres dels microserveis dissenyats a la PRAC1, segons un conjunt de consideracions tècniques i la documentació de les evidències.
2. Raonar com resoldre l'accés als microserveis des de diferents clients.

La PRAC 2 abasta els continguts del llibre "Microservices Patterns" (veure en l'apartat Recursos els capítols que cal estudiar).

### Competències

En aquesta PRAC 2 es treballa les següents competències del Grau en Enginyeria Informàtica:

- Saber construir aplicacions informàtiques mitjançant tècniques de desenvolupament, integració i reutilització.
- Saber proposar i avaluar diferents alternatives per resoldre un problema concret.

### Objectius

Els objectius de la PRAC 2 són:

- Implementar microserveis centrats en la lògica de negoci i l'accés a dades.
- Desplegar microserveis amb estat propi i intercomunicació asíncrona.
- Aplicar conceptes bàsics de l'arquitectura de microserveis.
- Dominar les tecnologies bàsiques necessàries per a la implementació de microserveis.

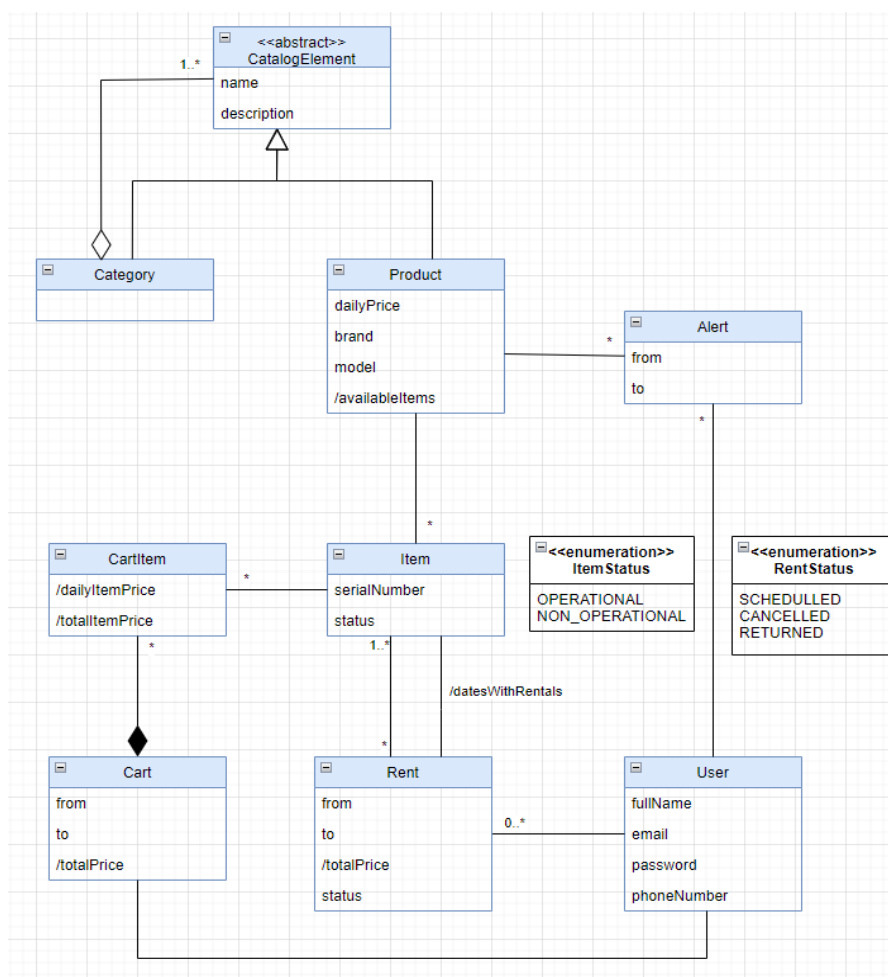
## Descripció de la PRAC 2

### Exercici 1 (8 punts)

L'exercici 1 consta de dues parts, la primera part consistirà en la implementació d'un programari, i la segona part en la documentació de les evidències del que s'ha implementat. A continuació, es descriuen les tasques concretes a realitzar en cadascuna de les dues parts.

### Primera part (6 punts)

Consisteix a implementar un subconjunt dels microserveis que s'han dissenyat a la PRAC1. S'ha de basar en la solució publicada per la PRAC anterior i per tant es parteix del següent diagrama del model general del domini, del qual es poden extreure las principals entitats (amb els seus atributs) i relacions:



El subconjunt de serveis a implementar es compon dels serveis *Product Catalog*, *User* i *Notification*:

- El microservei *Product Catalog* servirà tant per donar cobertura a les diverses operacions que els administradors de la plataforma necessiten per gestionar els equips (crear/eliminar productes, seccions/subseccions i unitats de producte) com per a les consultes del catàleg de protagonistes que realitzaran els usuaris.
- El microservei *User* serveix per gestionar les altes, les dades personals dels usuaris, i la configuració d'alertes. Els usuaris han de poder donar d'alta alertes per a un producte concret i un rang de dates. Quan una unitat d'aquest producte quedi lliure per a aquest rang de dates, s'ha d'emetre un missatge a la cua perquè un altre servei s'encarregui de fer efectiva l'alerta (microservei *Notification*).
- El microservei *Notification* té la finalitat d'emetre notificacions als usuaris interessats en un producte concret, quan s'afegeixen unitats o alguna queda lliure per ser llogada. Aquest microservei ha d'escoltar la cua de missatges de manera que quan s'alliberi una unitat de producte per a una data en la que algun usuari tingui configurada una alerta, a aquest usuari se li envii un correu electrònic amb l'avís.

D'acord amb el descrit, aquests tres microserveis tindran com a mínim les següents operacions:

Microservei *Product Catalog* (catàleg de productes)

- crear secció/subsecció
- crear producte
- crear unitat de producte
- eliminar producte
- canviar l'estat d'una unitat de producte a operatiu/no-operatiu
- consulta de seccions/subseccions per nom
- consulta de seccions/subseccions per descripció
- consulta de seccions/subseccions per secció "pare"
- consulta de productes per nom
- consulta de productes per secció
- consulta del detall d'un producte
- consulta del detall d'una unitat

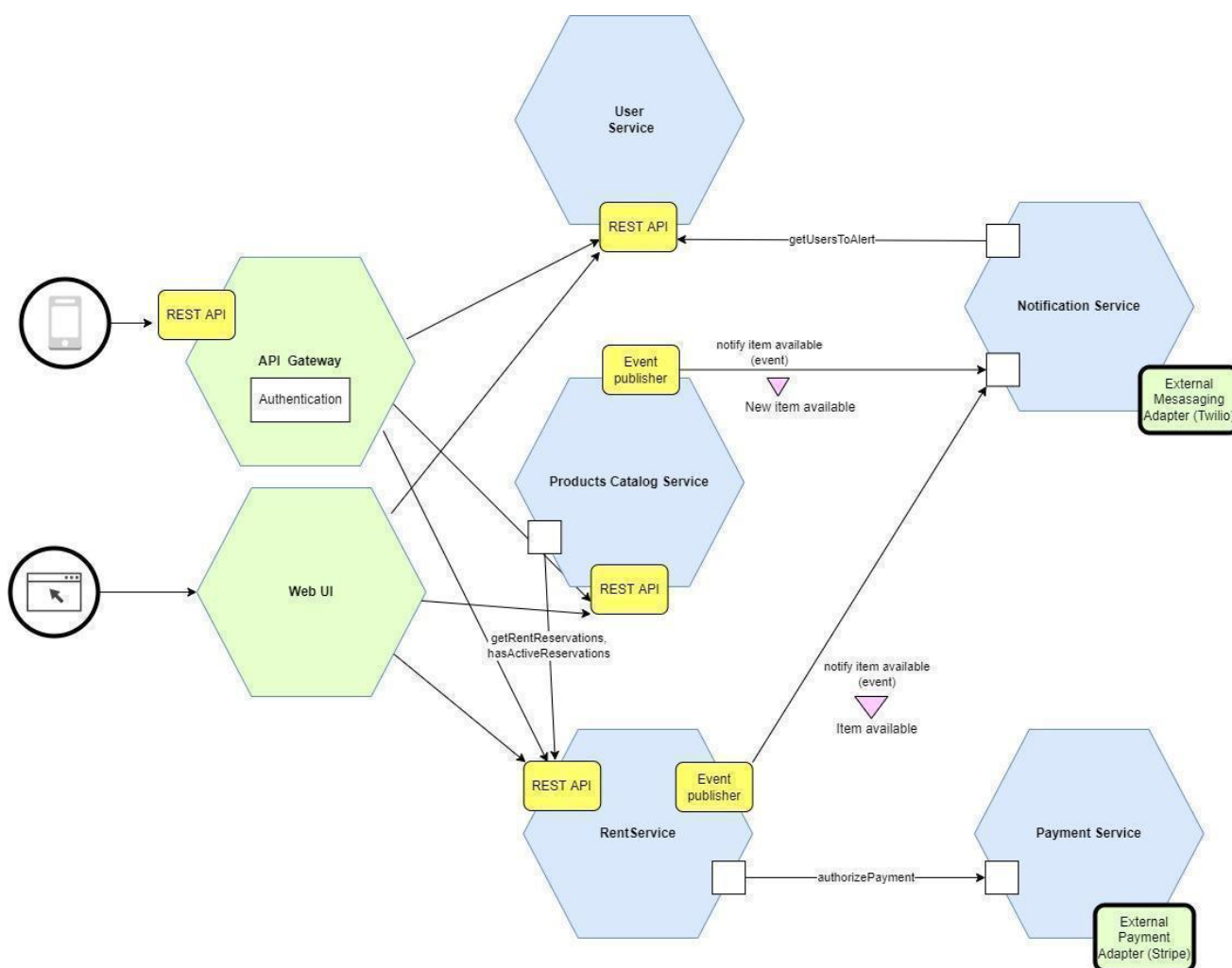
Microservei *User* (gestió d'usuaris)

- alta d'usuari
- alta d'alerta
- consulta de les alertes d'un usuari per a un interval concret de dates
- consulta de les alertes per producte i data
- consulta d'usuaris a alertar per a un producte i data (operació *getUsersToAlert*)
- consulta del detall d'una alerta

Microservei *Notification* (notificacions automàtiques a usuaris)

- enviar notificació (operació de sistema, no disponible per als usuaris)

En el següent diagrama es pot veure com ha de ser la col·laboració entre els diversos serveis del sistema:



## Consideracions prèvies

Per poder aconseguir la flexibilitat, resiliència i rendiment que caracteritza aquesta arquitectura, cal complir una sèrie de requeriments:

- Cada microservei:
  - s'ha de poder executar independentment,
  - ha de tenir estat propi (i accedir de manera independent a la BBDD),
  - implementa només la seva lògica de negoci,
  - exposa el seu contracte (definició de l'API),
  - gestiona el seu propi *log*.
- Sempre que sigui possible, la comunicació entre els diversos microserveis ha de ser asíncrona i per mitjà de cues de missatges.

- Per a certes operacions, pot ser necessària la col·laboració entre diversos microserveis per obtenir informació. En aquest cas, els microserveis poden realitzar peticions a altres microserveis per mitjà de trucades REST.
- Coses que queden fora de l'abast del projecte en la fase pilot que es demana implementar:
  - Autenticació: no serà necessari cap tipus d'autenticació per a cap de les operacions, es dona per fet que aquesta part de la funcionalitat del sistema la proporciona l'API Gateway (que no s'ha d'implementar).
  - Service Discovery & Registration: no s'ha d'implementar cap tipus de funcionalitat de registre i recerca de serveis, es dona per fet que només existeix una única instància, en una ubicació coneguda.
  - Enviament de correus: per no haver d'implementar aquesta part, de poc valor didàctic dins la PRAC, us suggerim que simuleu el seu comportament amb l'ús de Mocks. S'ha de poder demostrar almenys una línia de log cada vegada que s'envii una notificació (per exemple, mostrar mitjançant el log del microservei *Notification* una línia de log amb nivell INFO que digui "S'ha notificat per correu l'usuari XXX YYY ZZZ que existeix una unitat disponible del producte AAA").

### Plataforma de desenvolupament

- Entorn de desenvolupament: IntelliJ IDEA / Eclipse
- JDK: OpenJDK 11
- Framework: Spring / Spring Boot
- Gestió de dependències i build: Maven
- Servidor HTTP: Tomcat (integrat per defecte amb Spring Boot)
- Llibreria crides REST: Jersey
- Comunicació entre serveis: Apache Kafka
- BBDD: PostgreSQL
- Implementació ORM: Hibernate
- Generació de la documentació de l'API OpenAPI v3: springdoc-openapi
- Log: Log4j2
- Mocks: Mockneat (per generar dades pseudo-aleatòries)

### Punt de partida i recursos

Per tenir un punt de partida comú i homogeni, i poder centrar els esforços en el desenvolupament dels microserveis, juntament amb aquest enunciat es proporciona un enllaç a un repositori públic (<https://github.com/ppinedar/epcsd-spring-2023>) amb els següents recursos:

- Un arxiu *docker-compose.yml* per instal·lar la infraestructura bàsica per a la implementació d'aquest exercici i un arxiu *README.md* amb instruccions per arrencar-la.
- Esquelets d'aplicació Spring Boot amb les funcionalitats bàsiques per generar:

- un arxiu java jar executable amb totes les dependències contingudes en el projecte (per poder fer debug localment);
- un arxiu d'imatge docker amb el microservei i totes les seves dependències (per poder fer l'entrega).
- Un conjunt d'enllaços a documentació, guies i tutorials relatius tant a les eines a utilitzar com a la pròpia implementació dels microserveis sol·licitats (es penjaran a l'aula de Laboratori).

Cadascun dels esquelets d'aplicació que es proporcionen són un projecte Spring Boot complet pre-configurat amb les dependències mínimes necessàries per fer tota la implementació que es demana en aquest enunciat. Això no significa que no es puguin afegir altres dependències si us sembla adequat, però amb el que n'hi ha és suficient per fer tota la implementació.

Amb els contenidors de l'arxiu *docker-compose* arrencats, tots els esquelets de microservei haurien de poder posar-se en marxa i connectar a la infraestructura tal com es lliuren amb aquesta PRAC2. A més, aprofitant que els esquelets porten el seu propi servidor HTTP encastat, s'ha inclòs un mòdul *springdoc-openapi* que auto-genera i publica l'arxiu de definició del microservei (API) en format *JSON/OpenAPI v3*, així com una petita interfície web *SwaggerUI* que permet fer crides a les operacions dels microserveis.

Les adreces per accedir a aquestes funcionalitats són:

- arxius de definició dels serveis:
  - *Product Catalog*: <http://localhost:18081/v3/api-docs/>
  - *User*: <http://localhost:18082/v3/api-docs/>
- interfícies web SwaggerUI per a proves:
  - *Product Catalog*: <http://localhost:18081/swagger-ui/index.html>
  - *User*: <http://localhost:18082/swagger-ui/index.html>

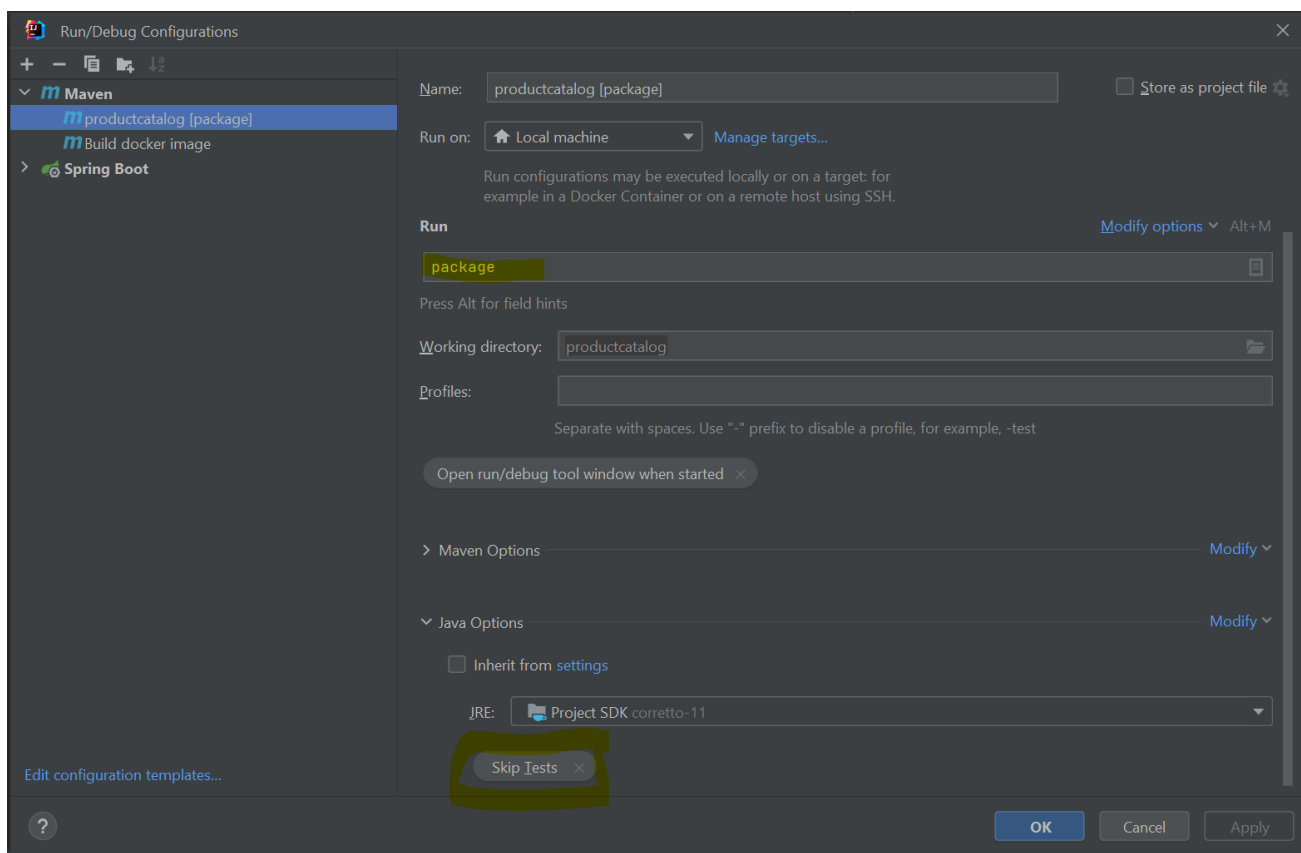
Com es pot veure, aquests exemples formen una base força completa des d'on començar el desenvolupament i, per tant, s'hauria d'evitar fer modificacions en els seus components bàsics (excepte petites configuracions, etc.), i només afegir les funcionalitats que es demanen en aquest enunciat. Trobareu més informació i suport tècnic al Laboratori de l'assignatura.

## Lliuraments

### Especificacions tècniques de l'aplicació a lliurar

El lliurament d'aquesta part de l'exercici serà un únic arxiu ZIP que ha de contenir:

- Tot el codi i arxius de configuració del projecte en un arxiu ZIP (dins del ZIP "principal").
- Un arxiu JAR java executable (generat pel propi esquelet del projecte lliurat) per a cadascun dels serveis, que contindrà totes les classes i llibreries necessàries (dependències) per executar el servei en qüestió. Aquest arxiu es pot generar a partir de l'esquelet d'aplicació proporcionat amb una execució maven i el paquet "goal".



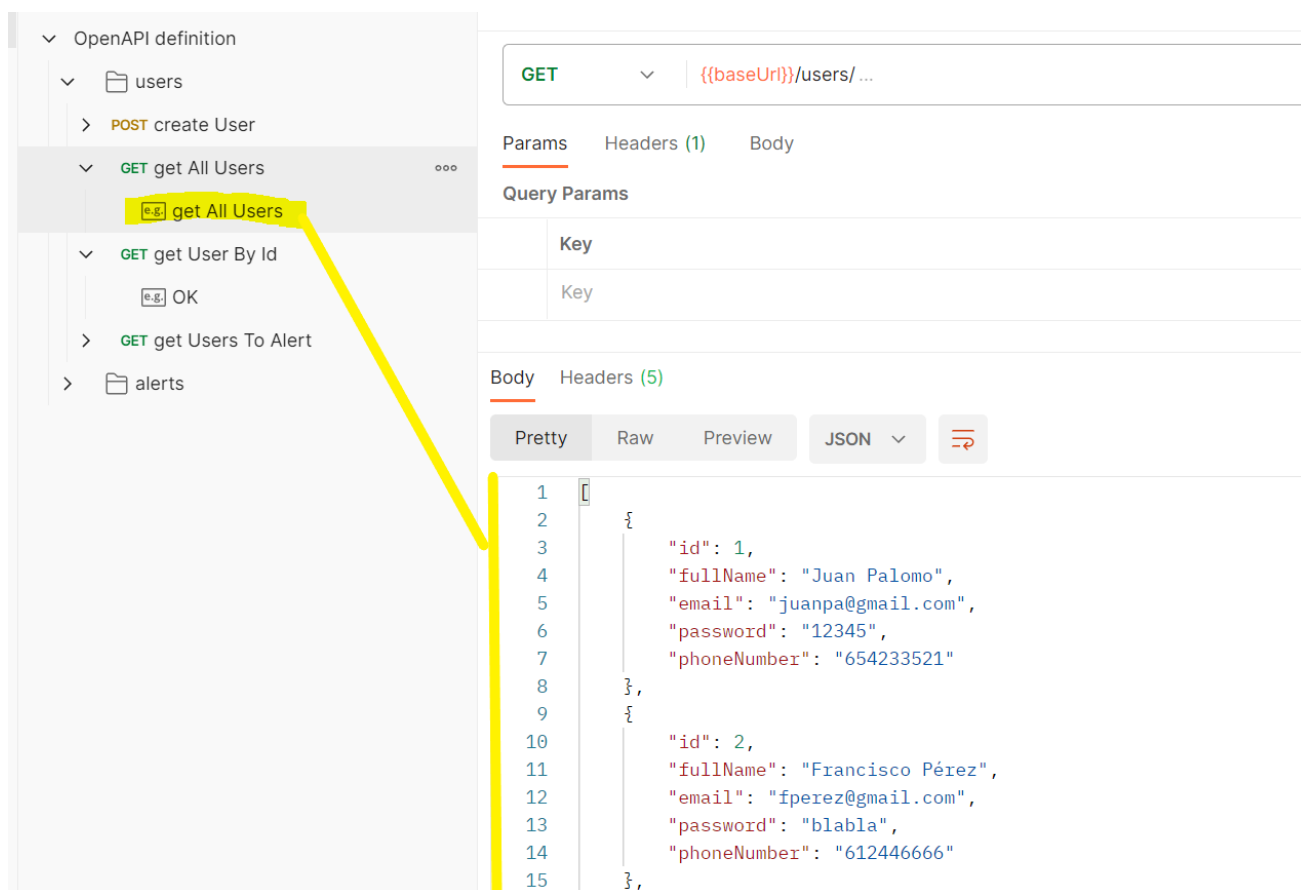
- Un arxiu readme.txt amb la informació següent:
  - instruccions per posar en marxa la infraestructura necessària per a la plataforma (en cas que sigui diferent a la que es proporciona amb docker);
  - qualsevol configuració necessària per poder executar els serveis;
  - la comanda concreta per executar cada servei per separat (si no s'han realitzat canvis, n'hi hauria prou amb "java -jar productcatalog-VERSION.jar", etc.)

## Segona part (2 punts)

Per a poder demostrar que s'han assolit els objectius de la implementació, haureu de proporcionar evidències del correcte funcionament dels microserveis. Per elaborar aquestes evidències us proposem utilitzar l'aplicació Postman, una eina que us permetrà importar l'arxiu de definició del servei (accessible des de la ruta `v3/api-docs` dels tres microserveis) i generar una col·lecció que contindrà totes les operacions disponibles. Com a mínim es realitzarà un exemple de cada tipus de crida amb dades "reals" un cop posats en marxa els microserveis.

## Liuraments

El lliurament d'aquesta part de l'exercici serà el propi arxiu de la col·lecció amb els exemples de crida per a cadascuna de les operacions implementades en la primera part.



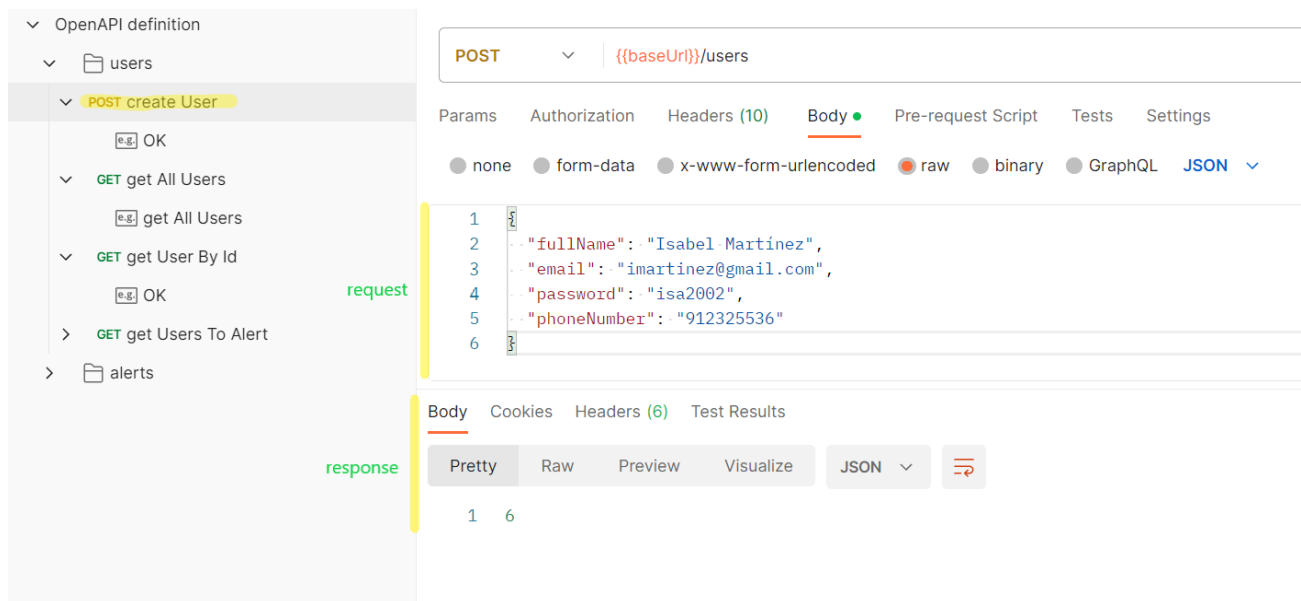
The screenshot shows the Postman interface. On the left, the 'OpenAPI definition' is expanded, showing a collection of endpoints under 'users'. The endpoint 'GET get All Users' is selected, and a yellow arrow points to it. The main panel shows the request details for this endpoint. The URL is `{{baseUrl}}/users/...`. The 'Query Params' section is empty. The 'Body' section is expanded, showing a JSON response with two user objects. The response is formatted as JSON and is shown in a 'Pretty' view.

```

1  [
2    {
3      "id": 1,
4      "fullName": "Juan Palomo",
5      "email": "juanpa@gmail.com",
6      "password": "12345",
7      "phoneNumber": "654233521"
8    },
9    {
10     "id": 2,
11     "fullName": "Francisco Pérez",
12     "email": "fperez@gmail.com",
13     "password": "blabla",
14     "phoneNumber": "612446666"
15   }
  ]

```





## Exercici 2 (2 punts)

La posada en marxa del projecte **Photo&Film4You** ha estat tan exitosa que diverses empreses s'han mostrat molt interessades a establir col·laboracions. Una de les propostes que l'actual equip de direcció està valorant és la del cercador d'equips per a rodatges **ShootMe**.

Aquesta plataforma, que és líder en el sector, funciona com un cercador unificat (agrupador) per a diversos serveis de lloguer d'equips de fotografia, rodatge, il·luminació, etc. Si s'arriba a un acord, suposaria poder arribar a un públic molt més gran, només a canvi d'un petit % per cada *referral*. Funcionaria de manera que les recerques dins de **ShootMe** consultarien el catàleg de **Photo&Film4You** (juntament amb d'altres) i mostrarien els resultats. Si l'usuari s'interessa per algun dels models disponibles per llogar a **Photo&Film4You** i fa clic, llavors se li redirigirà a la nostra aplicació amb un codi d'afiliat, des d'on es gestionarà el lloguer de la forma habitual.

Tot i que la direcció veu amb molt bons ulls aquest acord, el CTO té algunes reticències. En una primera reunió tècnica amb el CTO de **ShootMe** ha pogut observar que es tracta d'una aplicació basada en serveis que necessitarà realitzar consultes al catàleg de productes. El que passa és que l'arquitectura de microserveis de **Photo&Film4You** és elàstica per naturalesa (el nombre d'instàncies de cada microservei fluctua, i cap servei té una IP o nom fixos) i això fa que no es pugui proporcionar un únic endpoint on l'aplicació **ShootMe** pugui apuntar.

Raoneu justificant, en base a l'escenari plantejat en aquest exercici, quin patró aplicareu per resoldre l'accés als microserveis de la nostra plataforma en les condicions acordades, i quins són els pros/cons de la vostra decisió.

## Lliurables

La resposta a aquest exercici es lliurarà en format text PDF.

## Recursos

### Recursos Bàsics

- Llibre "Microservices Patterns" (Chris Richardson): Chapter 6: "Developing business logic with event sourcing" (excepte secció 6.3).
- Llibre "Microservices Patterns" (Richardson): Chapter 7: "Implementing queries in a microservice architecture"
- Llibre "Microservices Patterns" (Richardson): Chapter 8: "External APIO patterns" (excepte secció 8.3).

### Recursos Complementaris

- Repositori de codi font de l'exemple del llibre Microservices Patterns. <https://github.com/microservices-patterns/ftgo-application>.
- Exemple de programació de microserveis. <https://github.com/vmudigal/microservices-sample>

## Criteris d'avaluació

- La PRAC 2 s'ha de resoldre de forma **estrictament individual**. En cas de detectar còpies o qualsevol altra forma de frau acadèmic, es penalitzarà la prova amb una D com a nota.
- El pes de cada exercici està indicat en l'enunciat.
- Cal justificar la solució a cadascun dels exercicis. Es valorarà tant la correcció de la solució com la justificació donada.

## Format i data de lliurament

Cal lliurar un únic arxiu ZIP amb les respostes a tots els exercicis. El nom de l'arxiu ha de ser: *CognomsNom\_EPCSD\_PRAC2.zip*.

Estructura de l'arxiu lliurable *CognomsNom\_EPCSD\_PRAC2.zip*:

- ZIP amb el lliurament de l'exercici 1 part 1, que ha de contenir:
  - ZIP amb el codi font
  - un arxiu JAR executable per a cada microservei implementat
  - readme.txt amb instruccions addicionals per arrencar la infraestructura i/o microserveis (sobretot si s'han realitzat canvis en la configuració o en el docker-compose)
- Col·lecció Postman de l'exercici 1 part 2.
- PDF de resposta de l'exercici 2.

Aquest document es lliurarà a l'espai de Registre d'Avaluació Contínua (RAC) de l'aula abans de les **23:59 hores del dia 29 de maig de 2023**. No s'acceptaran lliuraments fora de termini.