

Pràctica 1.

API Web lloguer habitacions.

GEI URV
Curs 2019-2020
Sergio Candalija Valderrama
Miquel Prats Segura

Introducció	3
Estructura de la pràctica	3
POJO i context de persistència	3
Room	3
RoomType i requeriments	4
Tenant	5
RoomFacadeREST	6
TenantFacadeREST	6
Sistema de fitxers	7
Requeriments i tipus de l'habitació	8
Decisions de disseny	8
Jocs de proves	8
GET /rest/api/v1/room/location=\${city}&sort=\${criterion}	8
GET /rest/api/v1/room/\${id}	12
POST /rest/api/v1/room	13
DELETE /rest/api/v1/books/\${id}	15
GET /rest/api/v1/tenant	16
GET /rest/api/v1/tenant/\${id}	17
POST /rest/api/v1/tenant	18
PUT /rest/api/v1/tenant/\${id}	18
DELETE /rest/api/v1/tenant/\${id}	19
Conclusions	19
Manual d'instal·lació	20

1. Introducció

Per a aquesta pràctica, hem de construir un servidor REST amb JPA que simula un lloc web on els inquilins poden buscar habitacions i els arrendadors poden pujar-ne les habitacions que tinguin disponibles.

La comunicació entre client i servidor es farà amb objectes JSON.

Per a aquesta primera fase, totes les operacions seran disponibles i possibles per a qualsevol persona, a excepció de fer una sol·licitud de lloguer, que requereix que l'usuari estigui autenticat.

Primer de tot construirem una API Web per així crear la funcionalitat necessària per al lloguer d'habitacions. Això ho farem implementant les interfícies RESTful de l'API Web, a través d'un model de dades relacional implementat amb JPA.

Per poder realitzar correctament la pràctica, aquesta haurà de suportar 4 funcionalitats diferents.

- Definir una API REST.
- Implementar serveis REST per a que treballin i retornin JSON.
- Fer servir versionat per la sostenibilitat de l'API.
- Implementar un client REST per així testejar amb la pràctica.

2. Estructura de la pràctica

2.1. POJO i context de persistència

2.1.1. Room

Classe POJO que conté tota la informació referent a les habitacions. Aquesta mateixa conté les *NamedQueries* que s'utilitzaran al servei REST del tractament de les habitacions.

```
@Entity
@Table(name = "ROOM")
@NamedQueries({
    @NamedQuery(name= "Room.findAll", query= "SELECT r FROM Room
r"),
    @NamedQuery(name = "Room.findByRoomId", query = "SELECT r FROM
Room r WHERE r.roomId = :roomId"),
    @NamedQuery(name = "Room.findByLocation", query = "SELECT r FROM
Room r WHERE r.location = :location"),
    @NamedQuery(name = "Room.orderByASC", query = "SELECT r FROM
Room r ORDER BY r.preu ASC"),
    @NamedQuery(name = "Room.orderByDESC", query = "SELECT r FROM
Room r ORDER BY r.preu DESC"),
```

```

        @NamedQuery(name = "Room.findByLocationASC", query = "SELECT r
FROM Room r WHERE r.location = :location ORDER BY r.preu ASC"),
        @NamedQuery(name = "Room.findByLocationDESC", query = "SELECT r
FROM Room r WHERE r.location = :location ORDER BY r.preu DESC")
    })
@XmlRootElement
public class Room implements Serializable, Comparable<Room>{
    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.SEQUENCE, generator =
"Room_Gen")
    @Column(name="ROOM_ID")
    private int roomId;
    @Column(name = "LOCATION")
    private String location; //ciutat
    @Column(name = "ADDRESS")
    private String adreca;
    @Embedded
    private RoomType roomType;
    @Column(name = "PRICE")
    private float preu;
    @Embedded
    private Requeriments requeriments;
    @Column(name="DESCRIPTION")
    private String description;

    public Room() {
    }

    // Getters i setters

}

```

2.1.2. RoomType i requeriments

Els requeriments del llogaters, així com les característiques de les habitacions han implementat al JPA s'han implementat com un atribut de la classe *RoomType*. Aquests atributs són *@Embeddable* a la classe *Room*.

```

public class RoomType {

    @Column(name = "SIMPLE")
    private int simple;
    @Column(name = "EXTERIOR")
    private int exterior;
    @Column(name = "MOBLADA")

```

```

        private int moblada;

...

}

public class Requeriments implements Serializable{
    private static final long serialVersionUID = 1L;
    @Column (name = "SEXE")
    private String sexe;
    @Column (name = "MIN_EDAT")
    private Integer minEdat;
    @Column (name = "MAX_EDAT")
    private Integer maxEdat;
    @Column(name = "FUMADOR")
    private int fumador;
    @Column(name = "MASCOTES")
    private int mascotes;
}

```

2.1.3. Tenant

Tenant és una altra classe POJO que conté les dades referents als potencials inquilins que estan donats d'alta al sistema. Tanmateix, al contrari que *Room*, no implementa *@NamedQueries*.

```

@Entity
@Table(name= "TENANT")
@XmlRootElement
public class Tenant implements Serializable{
    private static final long serialVersionUID = 1L;
    @Id @GeneratedValue(strategy=AUTO)
    @Column(name= "TENANT_ID")
    private Integer tenantId;
    @Size(max = 30)
    @Column(name = "NAME")
    private String nom;
    @Size(max = 40)
    @Column(name = "EMAIL")
    private String email;
    @Size(max = 12)
    @Column(name = "PHONE")
    private String tlf;
    @Column(name = "EDAT")
    private int edat;
    @Column(name= "SEXE")
}

```

```

    private String sexe;
    @Column(name= "MASCOTES")
    private int mascotes;
    @Column(name= "FUMADOR")
    private int fumador;

    public Tenant() {
    }

    // Getters i setters

}

```

2.1.4. RoomFacadeREST

El servei REST de les habitacions implementa els diferents mètodes que demana l'enunciat d'aquesta pràctica. Les funcions d'aquest servei es troben sota el *path* /room/v1.

Implementa:

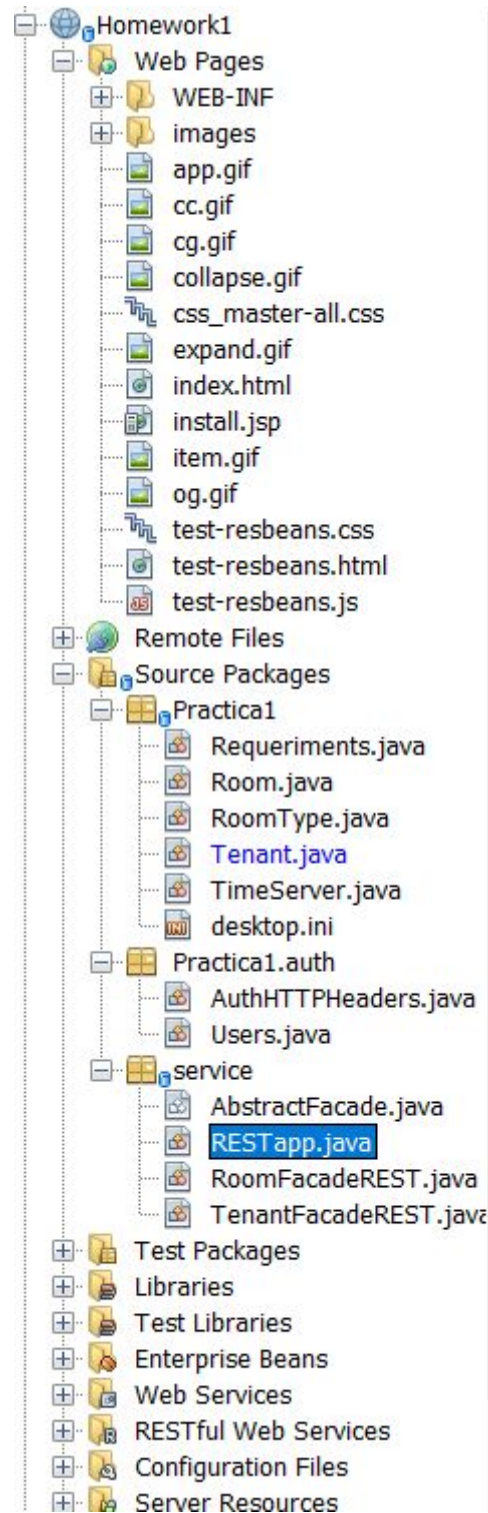
- Obtenir totes les habitacions al sistema ordenades en funció del preu de manera tant ascendent com descendent.
- Obtenir totes les habitacions al sistema ordenades d'una localitat específica en funció del preu de manera tant ascendent com descendent.
- Obtenir una habitació en funció del seu identificador.
- Afegir una nova habitació
- Modificar una habitació en funció del seu identificador.
- Eliminar una habitació en funció del seu identificador.

2.1.5. TenantFacadeREST

Les funcions a les que es poden fer-se sobre els inquilins estan definides sota la ruta /tenant/v1. Les funcions que permet són les següents:

- Obtenir tots els llogaters
- Obtenir un inquilí en funció del seu identificador.
- Registrar un inquilí.
- Eliminar un nou inquilí.
- Assignar un lloger a un inquilí per part d'un usuari autènticat.

2.2. Sistema de fitxers



Volem destacar:

- A Practica1 estan les classes creades per la pràctica
- Al paquet service estan els serveis REST implementats
- /web/install.jsp té els *INSERT* que es fan a la base de dades al arrancar el projecte.

2.3. Requeriments i tipus de l'habitació

Cada habitació serà d'un cert tipus, aquesta inclou:

- Simple o doble.
- Exterior o interior.
- Moblada.

A més del tipus d'habitació, cada arrendador pot ficar uns certs requeriments que el llogater ha de complir.

- Sexe: pot ser home, dona o unisex.
- Rang d'edat: pot ser de 0 a 99 anys.
- Fumadors.
- Mascotes.

3. Decisions de disseny

La relació entre els diferents objectes POJO i les bases de dades s'ha fet amb JPA.

Per a crear el servei web, em empleat JAX-WS.

En quant a les classes, hem decidit que les que anirien a la BD serien Room(habitació) i Tenant(llogater).

En quant a les classes Requeriments i RoomType són classes `@Embedded` que a la BD estaran "encastades" a la taula de Room on cada variable serà una columna més. Això ho hem fet així perquè considerem que no contenen suficients columnes per separat i és més eficient juntar-les a la taula de Room.

4. Jocs de proves

Per realitzar el joc de proves, al fitxer `install.jsp` introduïm nous tennants i habitacions per així poder fer consultes.

Aquestes consultes les hem fet a través de l'eina Postman.

Consultes fetes per les habitacions:

`GET /rest/api/v1/room/location=${city}&sort=${criterion}`

Es dona per vàlida: Sí

Exemples

- **Comanda realitzada:** GET
`localhost:8080/Sob_grup_03/rest/api/v1/room?sort=asc&location=Tarragona`
- **Codi de retorn:** 200 OK
- **Contingut de la resposta:**


```
[
  {
    "adreca": "Plaça",
    "adreça": "Plaça",
    "description": "Centrica",
    "location": "Tarragona",
    "preu": 72.0,
    "requeriments": {
      "fumador": 1,
      "mascotes": 1,
      "maxEdat": 50,
      "minEdat": 0,
      "sexe": "HOME"
    },
    "roomId": 350,
    "roomType": {
      "exterior": 0,
      "moblada": 1,
      "simple": 1
    }
  },
  {
    "adreca": "Carrer",
    "adreça": "Carrer",
    "description": "Àtic de primera classe",
    "location": "Tarragona",
    "preu": 98.0,
    "requeriments": {
      "fumador": 1,
      "mascotes": 1,
      "maxEdat": 99,
      "minEdat": 0,
      "sexe": "UNISEX"
    },
    "roomId": 300,
    "roomType": {
      "exterior": 1,
      "moblada": 1,
      "simple": 1
    }
  },
  {
    "adreca": "Placa",
    "adreça": "Placa",
    "description": "Garage",
    "location": "Tarragona",
    "preu": 200.0,
    "requeriments": {
```

```

        "fumador": 1,
        "mascotes": 1,
        "maxEdat": 99,
        "minEdat": 0,
        "sexe": "UNISEX"
    },
    "roomId": 250,
    "roomType": {
        "exterior": 1,
        "moblada": 1,
        "simple": 1
    }
},
{
    "adreca": "Placa",
    "adreça": "Placa",
    "description": "HOLA",
    "location": "Tarragona",
    "preu": 200.0,
    "requeriments": {
        "fumador": 1,
        "mascotes": 1,
        "maxEdat": 99,
        "minEdat": 0,
        "sexe": "DONA"
    },
    "roomId": 50,
    "roomType": {
        "exterior": 1,
        "moblada": 1,
        "simple": 0
    }
},
{
    "adreca": "Placa",
    "adreça": "Placa",
    "description": "HOLA",
    "location": "Tarragona",
    "preu": 600.0,
    "requeriments": {
        "fumador": 1,
        "mascotes": 1,
        "maxEdat": 99,
        "minEdat": 0,
        "sexe": "DONA"
    },
    "roomId": 150,
    "roomType": {

```

```

        "exterior": 1,
        "moblada": 1,
        "simple": 1
    }
},
{
    "adreca": "Placa",
    "adreça": "Placa",
    "description": "HOLA",
    "location": "Tarragona",
    "preu": 600.0,
    "requeriments": {
        "fumador": 1,
        "mascotes": 1,
        "maxEdat": 99,
        "minEdat": 0,
        "sexe": "DONA"
    },
    "roomId": 100,
    "roomType": {
        "exterior": 1,
        "moblada": 1,
        "simple": 1
    }
}
]

```

- **Cas d'ús:** Habitacions a Reus en ordre descendent
- **Comanda realitzada:** GET
localhost:8080/Sob_grup_03/rest/api/v1/room?sort=desc&location=Resu
- **Codi de retorn:** 200 OK
- **Contingut de la resposta:**

```

[
    {
        "adreca": "Avinguda",
        "adreça": "Avinguda",
        "description": "Apta per cadira de rodes",
        "location": "Reus",
        "preu": 168.0,
        "requeriments": {
            "fumador": 0,
            "mascotes": 0,
            "maxEdat": 99,
            "minEdat": 20,
            "sexe": "Unisex"
        },
        "roomId": 200,
    }
]

```

```

        "roomType": {
            "exterior": 1,
            "moblada": 1,
            "simple": 1
        }
    }
}
1

```

- **Cas d'ús:** Habitacions a Reus, sense marcar el camp obligatori *sort*
- **Comanda realitzada:** GET localhost:8080/Sob_grup_03/rest/api/v1/room?location=Reus
- **Codi de retorn:** 400 Bad Request
- **Contingut de la resposta:**

GET /rest/api/v1/room/\${id}

- **Es dona per vàlida:** Sí

Exemples

- **Cas d'ús:** *room* que existeix
- **Comanda realitzada:** GET localhost:8080/Sob_grup_03/rest/api/v1/room/200
- **Codi de retorn:** 200 OK
- **Contingut de la resposta:**

```

{
    "adreca": "Avinguda",
    "adreça": "Avinguda",
    "description": "Apta per cadira de rodes",
    "location": "Reus",
    "preu": 168.0,
    "requeriments": {
        "fumador": 0,
        "mascotes": 0,
        "maxEdat": 99,
        "minEdat": 20,
        "sexe": "Unisex"
    },
    "roomId": 200,
    "roomType": {
        "exterior": 1,
        "moblada": 1,
        "simple": 1
    }
}

```

- **Cas d'ús:** *room* que no existeix
- **Comanda realitzada:** GET localhost:8080/Sob_grup_03/rest/api/v1/room/2222
- **Codi de retorn:** 404 NOT FOUND
- **Contingut de la resposta:**

ROOM no trobada

POST /rest/api/v1/room

Es dóna per vàlida: Sí

Exemples

- **Cas d'ús:** Pujar una nova *room*.
- **Comanda realitzada:** POST localhost:8080/Sob_grup_03/rest/api/v1/room 200 OK
- **JSON enviat:** {
 - "adreca": "Placa",
 - "adreça": "Placa",
 - "description": "Habitació POST",
 - "location": "Amposta",
 - "preu": 200.0,
 - "requeriments": {
 - "fumador": 1,
 - "mascotes": 1,
 - "maxEdat": 99,
 - "minEdat": 0,
 - "sexe": "DONA"
 - },
 - "roomType": {
 - "exterior": 1,
 - "moblada": 1,
 - "simple": 0
 - }
- }
- **Codi de retorn:** 204
- **Contingut de la resposta:**

SELECT * FROM ROOT.ROOM F... X				
Max. rows: 100 Fetched Rows: 8				
#	ROOM_ID	ADDRESS	DESCRIPTION	LOCATION
1		50 Placa	HOLA	Tarragona
2		100 Placa	HOLA	Tarragona
3		150 Placa	HOLA	Tarragona
4		200 Avinguda	Apta per cadira de rodes	Reus
5		250 Placa	Garage	Tarragona
6		300 Carrer	Àtic de primera classe	Tarragona
7		350 Placa	Centrica	Tarragona
8		351 Placa	HabitaciÃ³ POST	Amposta

PUT /rest/api/v1/room/\${id}

- **Es dóna per vàlida:** NO

Exemples

- **Cas d'ús:** Modifiquem la *room* anterior
- **Comanda realitzada:** PUT localhost:8080/Sob_grup_03/rest/api/v1/room/351
- **JSON enviat:** {
 - "adreca": "Placa",

- "adreça": "Placa",
- "description": "Habitació PUT",
- "location": "Madrid",
- "preu": 200.0,
- "requeriments": {
- "fumador": 1,
- "mascotes": 1,
- "maxEdat": 99,
- "minEdat": 0,
- "sexe": "DONA"
- },
- "roomType": {
- "exterior": 1,
- "moblada": 1,
- "simple": 0
- }
- }
- }
-
- **Codi de retorn:** 200 OK
- **Contingut de la resposta:** `<?xml version="1.0" encoding="UTF-8" standalone="yes"?>`
- `<room>`
- `<adreca>Placa</adreca>`
- `<adreça>Placa</adreça>`
- `<description>Habitació³ POST</description>`
- `<location>Amposta</location>`
- `<preu>200.0</preu>`
- `<requeriments>`
- `<fumador>1</fumador>`
- `<mascotes>1</mascotes>`
- `<maxEdat>99</maxEdat>`
- `<minEdat>0</minEdat>`
- `<sexe>DONA</sexe>`
- `</requeriments>`
- `<roomId>351</roomId>`
- `<roomType>`
- `<exterior>1</exterior>`
- `<moblada>1</moblada>`
- `<simple>0</simple>`
- `</roomType>`
- `</room>`
-
- **Cas d'ús:** Modifiquem la *room* anterior

- **Comanda realitzada:** PUT localhost:8080/Sob_grup_03/rest/api/v1/room/351
- **JSON enviat:** {
 - "adrecà": "Placa",
 - "adreça": "Placa",
 - "description": "Habitació PUT",
 - "location": "Madrid",
 - "preu": 200.0,
 - "requeriments": {
 - "fumador": 1,
 - "mascotes": 1,
 - "maxEdat": 99,
 - "minEdat": 0,
 - "sexe": "DONA"
 - },
 - "roomType": {
 - "exterior": 1,
 - "moblada": 1,
 - "simple": 0
 - }
- }
-
- **Codi de retorn:** 404 NOT FOUND
- **Contingut de la resposta:** ROOM no trobada

DELETE /rest/api/v1/books/\${id}

- **Es dona per vàlida:** Sí

Exemples

- **Cas d'ús:** Eliminar la *room* existent 351
- **Comanda realitzada:** DELETE localhost:8080/Sob_grup_03/rest/api/v1/room/351
- **Codi de retorn:** 200 OK
- **Resultat:**

Max. rows: 100 Fetched Rows: 6				
#	ROOM_ID	ADDRESS	DESCRIPTION	LOCATION
1		50 Placa	HOLA	Tarragona
2		100 Placa	HOLA	Tarragona
3		150 Placa	HOLA	Tarragona
4		200 Avinguda	Apta per cadira de rodes	Reus
5		250 Placa	Garage	Tarragona
6		300 Carrer	Àtic de primera classe	Tarragona

- **Cas d'ús:** Eliminar la *room* no existent 35100
- **Comanda realitzada:** DELETE localhost:8080/Sob_grup_03/rest/api/v1/room/35100
- **Codi de retorn:** 404 NOT FOUND
- **Resultat:** Room no trobada

GET /rest/api/v1/tenant

- Es dóna per vàlida: Sí

Exemples

- **Comanda realitzada:** GET localhost:8080/Sob_grup_03/rest/api/v1/tenant
- **Codi de retorn:** 200 OK
- **Resultat:**

```
[
  {
    "edat": 50,
    "email": "mail@mail.com",
    "fumador": 0,
    "mascotes": 1,
    "nom": "PEPE",
    "sexe": "HOME",
    "tenantId": 213,
    "tlf": "+34777666888"
  },
  {
    "edat": 50,
    "email": "mail@mail.com",
    "fumador": 0,
    "mascotes": 1,
    "nom": "Susan",
    "sexe": "DONA",
    "tenantId": 500,
    "tlf": "+34777666888"
  },
  {
    "edat": 20,
    "email": "mail2@mail.com",
    "fumador": 0,
    "mascotes": 0,
    "nom": "Susana",
    "sexe": "DONA",
    "tenantId": 1,
    "tlf": "+34777666888"
  },
  {
    "edat": 55,
    "email": "mail3@mail.com",
    "fumador": 0,
    "mascotes": 1,
    "nom": "PEPE",
    "sexe": "HOME",
    "tenantId": 151,
    "tlf": "+34777666888"
  }
]
```



```

    },
    {
        "edat": 32,
        "email": "mail5@mail.com",
        "fumador": 0,
        "mascotes": 0,
        "nom": "Karim",
        "sexe": "HOME",
        "tenantId": 650,
        "tlf": "+34777666888"
    },
    {
        "edat": 19,
        "email": "mail6@mail.com",
        "fumador": 1,
        "mascotes": 1,
        "nom": "Mike",
        "sexe": "HOME",
        "tenantId": 202,
        "tlf": "+34777666888"
    }
}
]

```

GET /rest/api/v1/tenant/\${id}

- Es dóna per válida: Sí

Exemples

- **Cas d'ús:** Obtenir el *tenant* 213
- **Comanda realitzada:** GET localhost:8080/Sob_grup_03/rest/api/v1/tenant/213
- **Codi de retorn:** 200 OK
- **Resultat:**

```

{
    "edat": 50,
    "email": "mail@mail.com",
    "fumador": 0,
    "mascotes": 1,
    "nom": "PEPE",
    "sexe": "HOME",
    "tenantId": 213,
    "tlf": "+34777666888"
}

```

- **Cas d'ús:** Obtenir el *tenant* que no existeix 21300
- **Comanda realitzada:** GET localhost:8080/Sob_grup_03/rest/api/v1/tenant/21300
- **Codi de retorn:** 404 Not found
- **Resultat:** Tenant no trobat

POST /rest/api/v1/tenant

- **Es dóna per válida:** SÍ

Exemples

- **Cas d'ús:** Afegir un nou *tenant*
- **JSON enviat:**

```
{
  "edat": 50,
  "email": "mail@mail.com",
  "fumador": 0,
  "mascotes": 1,
  "nom": "POST",
  "sexe": "HOME",
  "tlf": "+34777666888"
}
```

- **Comanda realitzada:** POST localhost:8080/Sob_grup_03/rest/api/v1/tenant/
- **Codi de retorn:** 200 OK
- **Resultat**

#	TENANT_ID	EDAT	EMAIL	FUMADOR	MASCOTES	NAME
1	213		50 mail@mail.com	0		1 PEPE
2	500		50 mail@mail.com	0		1 Susan
3	1		20 mail2@mail.com	0		0 Susana
4	151		55 mail3@mail.com	0		1 PEPE
5	650		32 mail5@mail.com	0		0 Karim
6	202		19 mail6@mail.com	1		1 Mike
7	2		50 mail@mail.com	0		1 POST

PUT /rest/api/v1/tenant/\${id}

- **Es dóna per válida:** NO

Exemples

- **Cas d'ús:** Modificar el *tenant* anterior
- **JSON enviat:**

```
{
  "edat": 50,
  "email": "mail@mail.com",
  "fumador": 0,
  "mascotes": 1,
  "nom": "ESPOSA DE POST",
  "sexe": "DONA",
  "tlf": "+34777666888"
}
```

- **Codi de retorn:** 200 OK

- Resultat

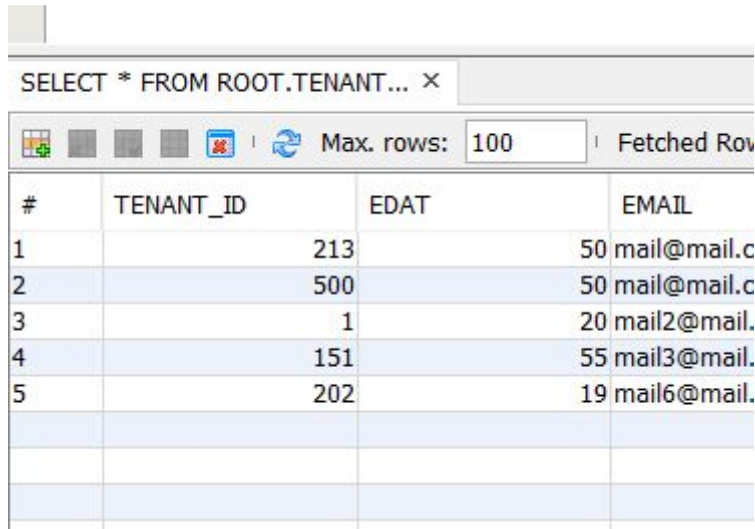
5	650	32 mail5@mail.com	0	0 Karim
6	202	19 mail6@mail.com	1	1 Mike
7	2	50 mail@mail.com	0	1 POST
8	3	50 mail@mail.com	0	1 ESPOSA DE POST

DELETE /rest/api/v1/tenant/\${id}

- **Es dona per vàlida:** Sí

Exemples

- **Cas d'ús:** Eliminar un *tenant* existent
- **Comanda realitzada:** DELETE localhost:8080/Sob_grup_03/rest/api/v1/tenant/650
- **Codi de retorn:** 200 OK
- **Resultat**



#	TENANT_ID	EDAT	EMAIL
1	213		50 mail@mail.c
2	500		50 mail@mail.c
3	1		20 mail2@mail.
4	151		55 mail3@mail.
5	202		19 mail6@mail.

- **Cas d'ús:** Eliminar un *tenant* no existent
- **Comanda realitzada:** DELETE localhost:8080/Sob_grup_03/rest/api/v1/tenant/65000
- **Codi de retorn:** 404 Not found
- **Contingut de la resposta:** Tenant no trobat

POST /rest/api/v1/tenant/\${id}/rent

- **NO IMPLEMENTAT**

5. Conclusions

Hem après bastant en l'utilització del servei REST i considerem que hem realitzat una pràctica suficient sòlida. Malgrat tot, ens hauria agradat implementar la autenticació amb èxit.

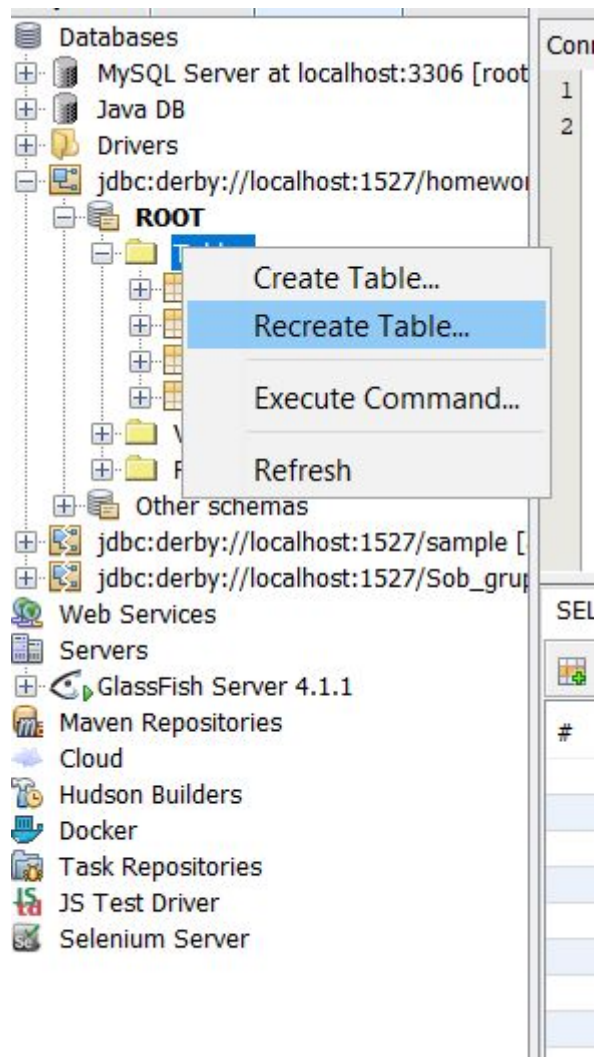
6. Manual d'instal·lació

1. Crear la BD Glassfish amb les següents especificacions
 - a. Crear BD amb nom *sob_grup_03*
 - b. Usuari *sob* i contrasenya *sob*
 - c. Crear les taules necessàries

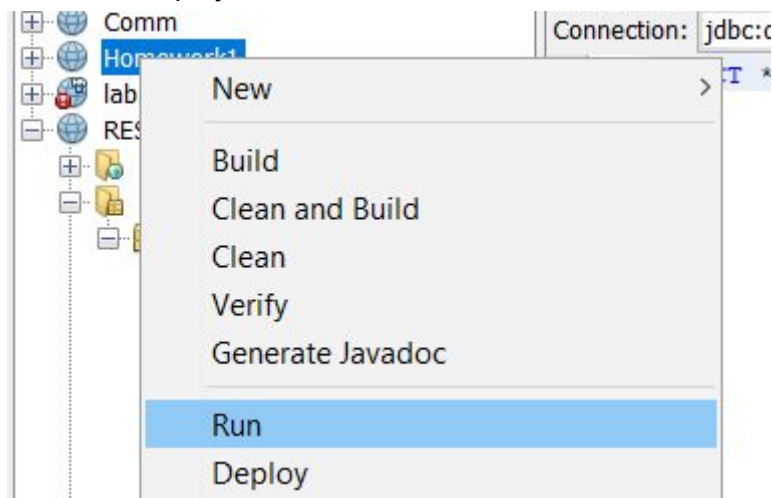
```
create table "SOB".ROOM
(
    ROOM_ID INTEGER not null primary key,
    ADDRESS VARCHAR(255),
    DESCRIPTION VARCHAR(255),
    LOCATION VARCHAR(255),
    PRICE DOUBLE,
    FUMADOR INTEGER,
    MASCOTES INTEGER,
    MAX_EDAT INTEGER,
    MIN_EDAT INTEGER,
    SEXE VARCHAR(255),
    EXTERIOR INTEGER,
    MOBLADA INTEGER,
    SIMPLE INTEGER
)
```

```
create table "SOB".TENANT
(
    TENANT_ID INTEGER not null primary key,
    EDAT INTEGER,
    EMAIL VARCHAR(255),
    FUMADOR INTEGER,
    MASCOTES INTEGER,
    NAME VARCHAR(255),
    SEXE VARCHAR(255),
    PHONE VARCHAR(255)
)
```

Podeu crear-les maualment o amb els fitxer *.grab a la arrel de l'entrega, tot fent SOB>
Tables > (*click dret*) Recreate Table



2. Fer RUN del projecte de NetBeans



3. Una vegada arrencat, s'obrirà una finestra al navegador amb un botó per fer les insercions a la base de dades.