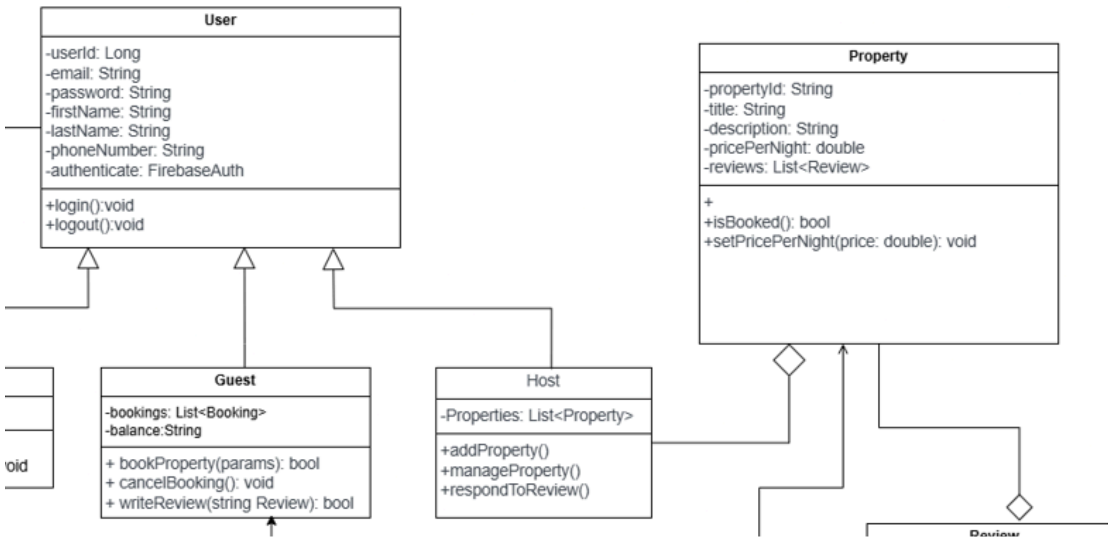
CSE434 Aspect and Service-Oriented Software Systems

Lab 3 Assignment

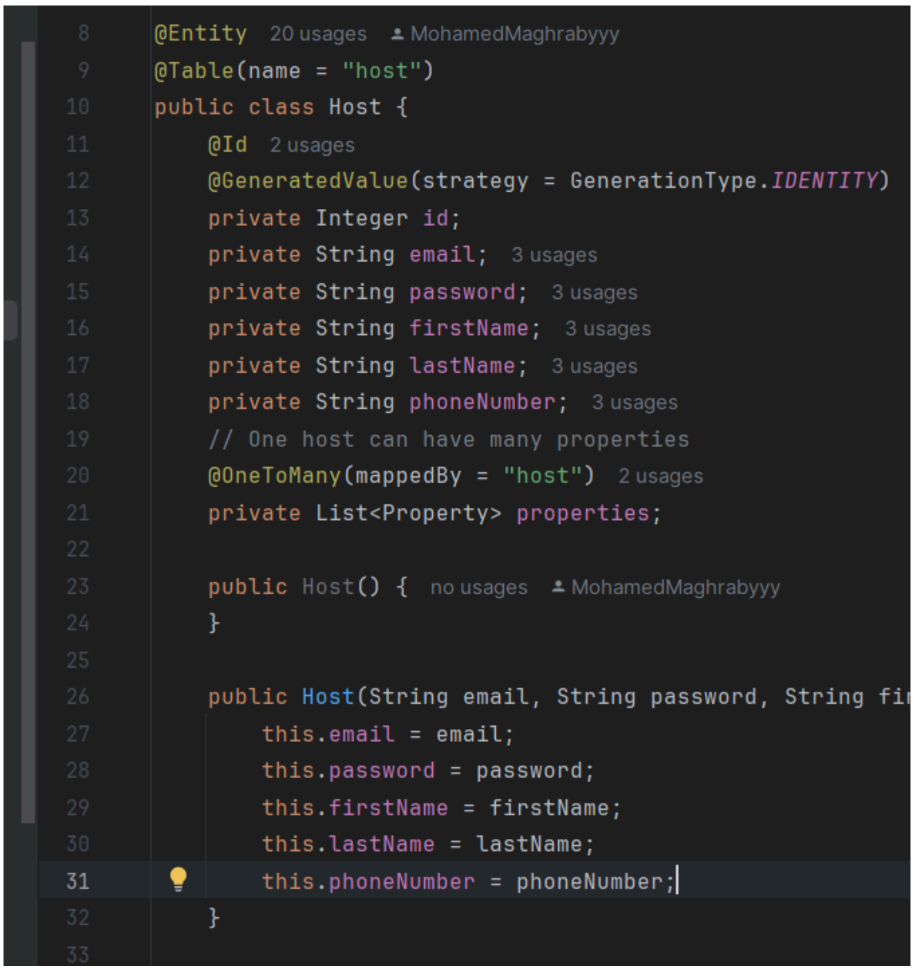
Name	Mohamed Mostafa Bedair ElMaghraby
ID	20P7732

1- Class Diagram snippet

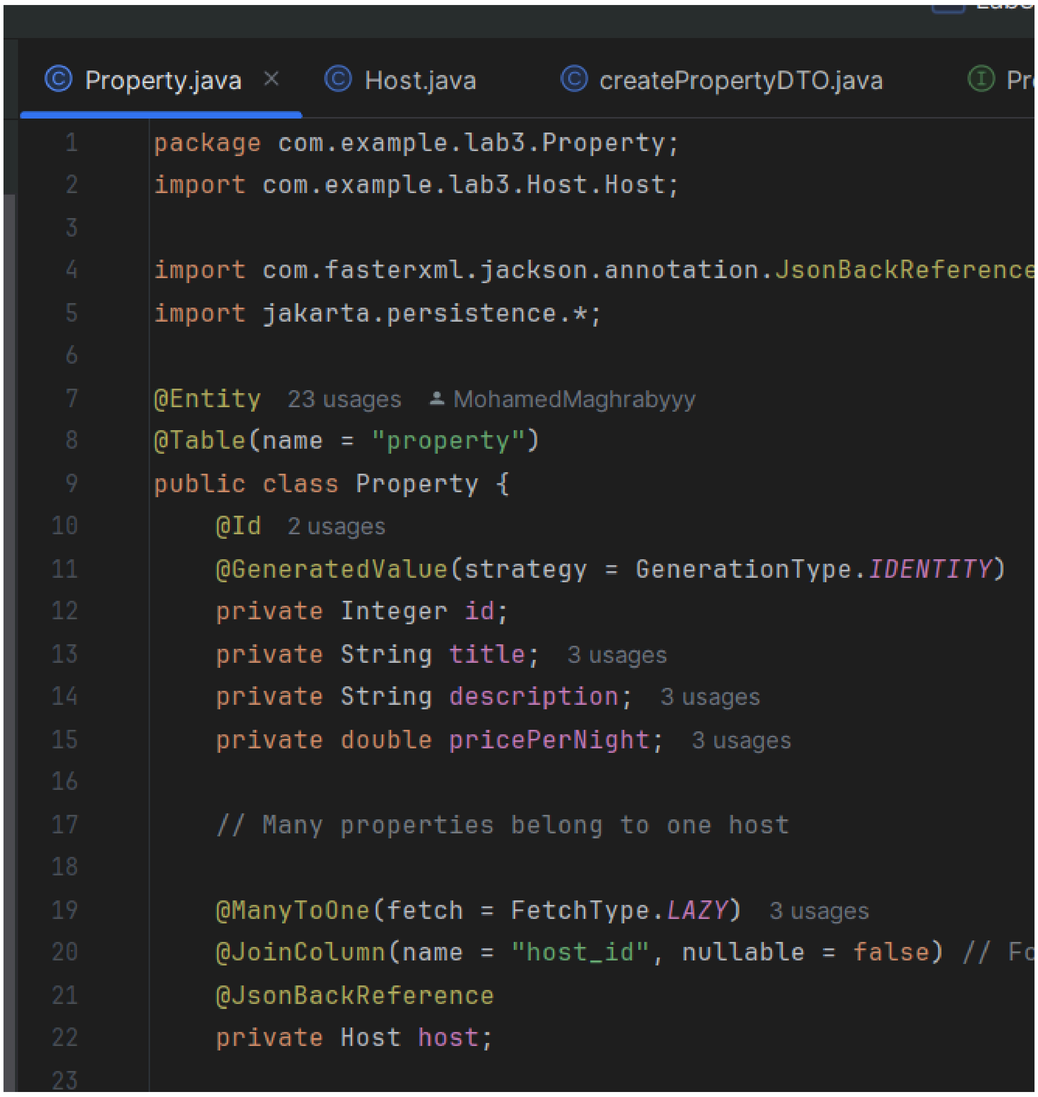


- Two entities that I implemented are **Host** and **Property**, with a one to many relationship where a host can have many properties but a property can only be owned by one user.

2- Models

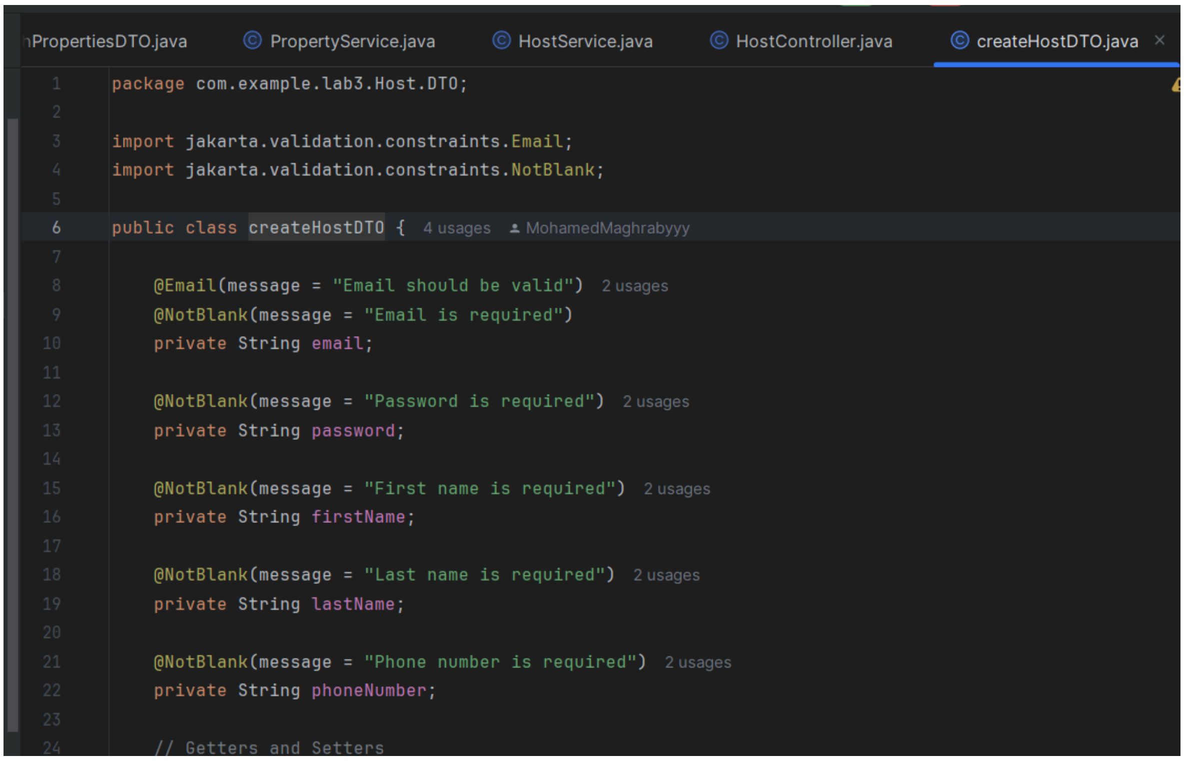


```java
@Entity  20 usages  MohamedMaghrabyyy
@Table(name = "host")
public class Host {
    @Id  2 usages
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;
    private String email;  3 usages
    private String password;  3 usages
    private String firstName;  3 usages
    private String lastName;  3 usages
    private String phoneNumber;  3 usages
    // One host can have many properties
    @OneToMany(mappedBy = "host")  2 usages
    private List<Property> properties;

    public Host() {  no usages  MohamedMaghrabyyy
    }

    public Host(String email, String password, String fi
        this.email = email;
        this.password = password;
        this.firstName = firstName;
        this.lastName = lastName;
        this.phoneNumber = phoneNumber;
    }

```

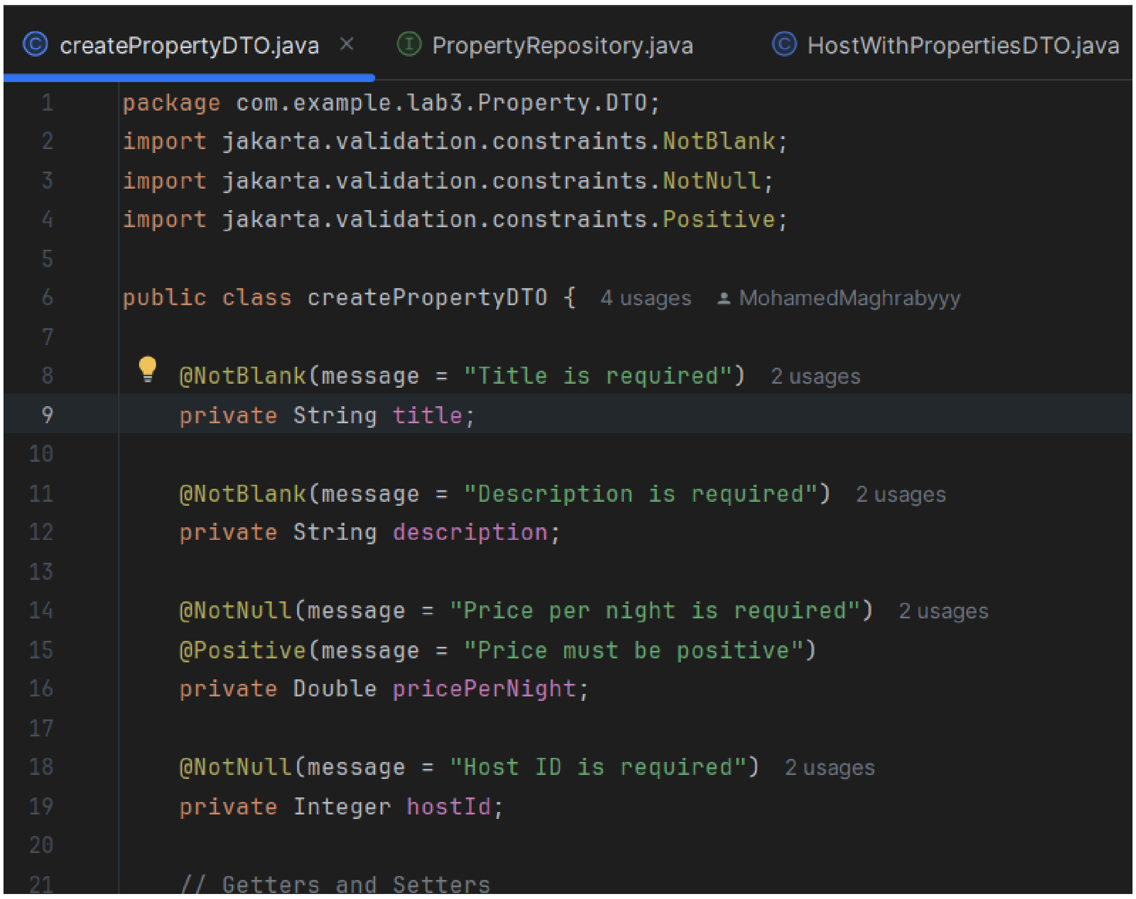


```
package com.example.lab3.Property;
import com.example.lab3.Host.Host;

import com.fasterxml.jackson.annotation.JsonBackReference
import jakarta.persistence.*;

@Entity  23 usages  MohamedMaghrabyyy
@Table(name = "property")
public class Property {
    @Id  2 usages
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;
    private String title;  3 usages
    private String description;  3 usages
    private double pricePerNight;  3 usages

    // Many properties belong to one host

    @ManyToOne(fetch = FetchType.LAZY)  3 usages
    @JoinColumn(name = "host_id", nullable = false) // Fo
    @JsonBackReference
    private Host host;

```

3- DTOs

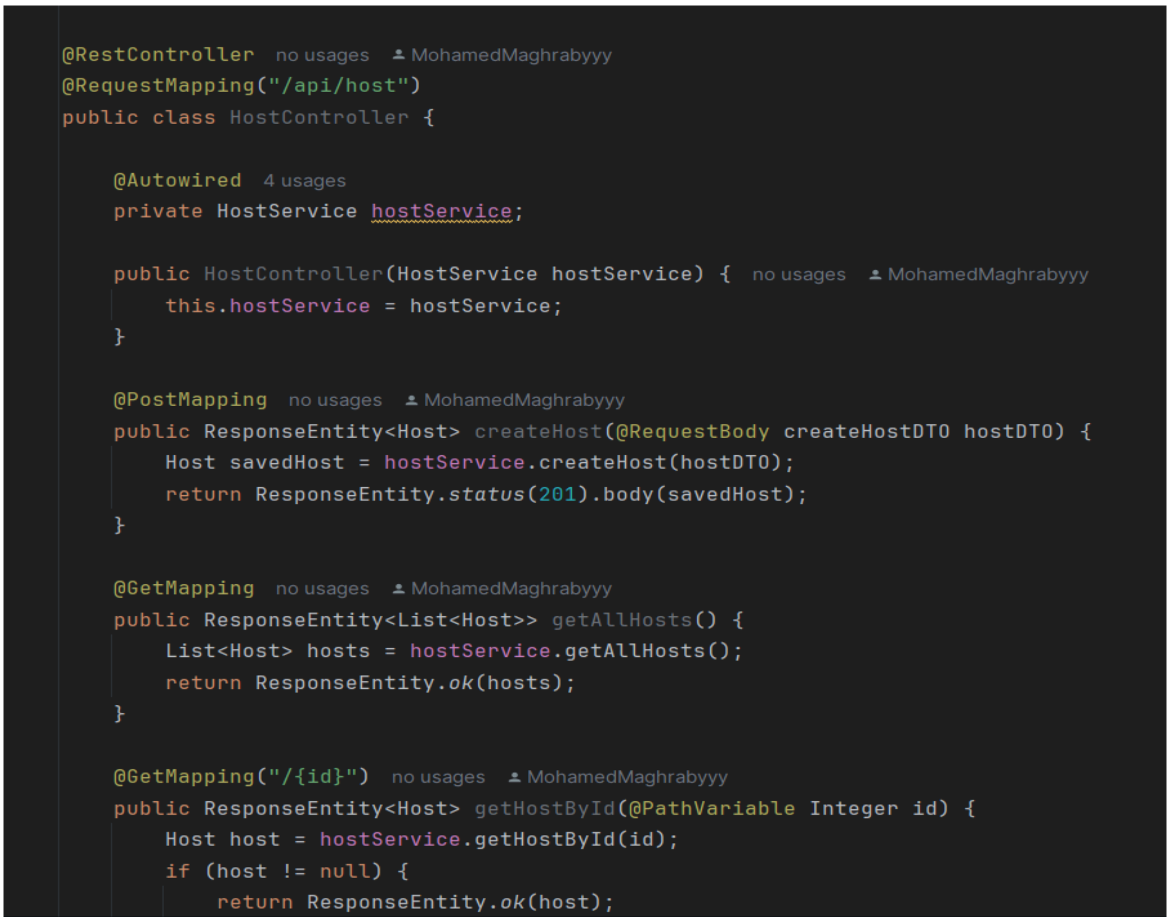


```java
package com.example.lab3.Host.DTO;

import jakarta.validation.constraints.Email;
import jakarta.validation.constraints.NotBlank;

public class createHostDTO {  4 usages  MohamedMaghrabyyy

    @Email(message = "Email should be valid")  2 usages
    @NotBlank(message = "Email is required")
    private String email;

    @NotBlank(message = "Password is required")  2 usages
    private String password;

    @NotBlank(message = "First name is required")  2 usages
    private String firstName;

    @NotBlank(message = "Last name is required")  2 usages
    private String lastName;

    @NotBlank(message = "Phone number is required")  2 usages
    private String phoneNumber;

    // Getters and Setters
```

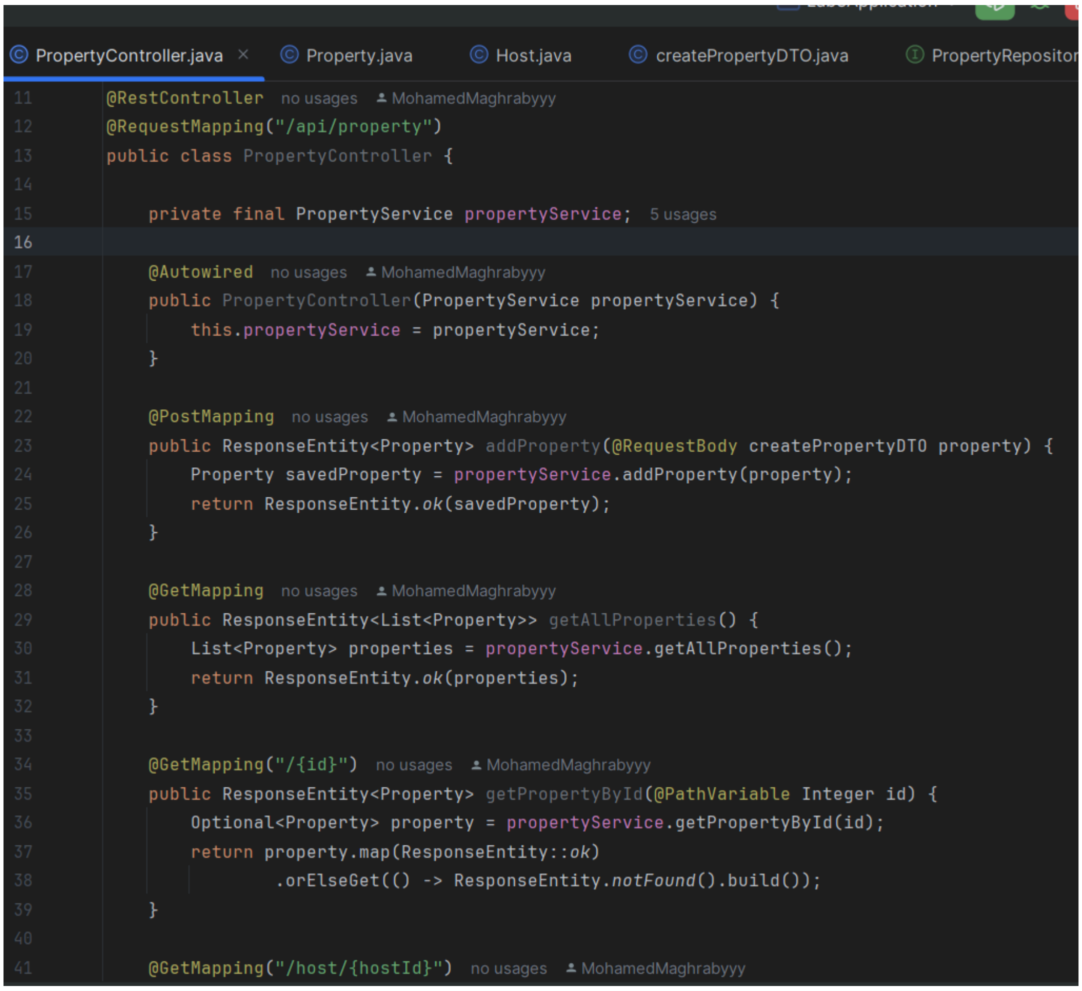


```java
package com.example.lab3.Property.DTO;
import jakarta.validation.constraints.NotBlank;
import jakarta.validation.constraints.NotNull;
import jakarta.validation.constraints.Positive;

public class createPropertyDTO {  4 usages  MohamedMaghrabyyy

    @NotBlank(message = "Title is required")  2 usages
    private String title;

    @NotBlank(message = "Description is required")  2 usages
    private String description;

    @NotNull(message = "Price per night is required")  2 usages
    @Positive(message = "Price must be positive")
    private Double pricePerNight;

    @NotNull(message = "Host ID is required")  2 usages
    private Integer hostId;

    // Getters and Setters
```

4- Controllers

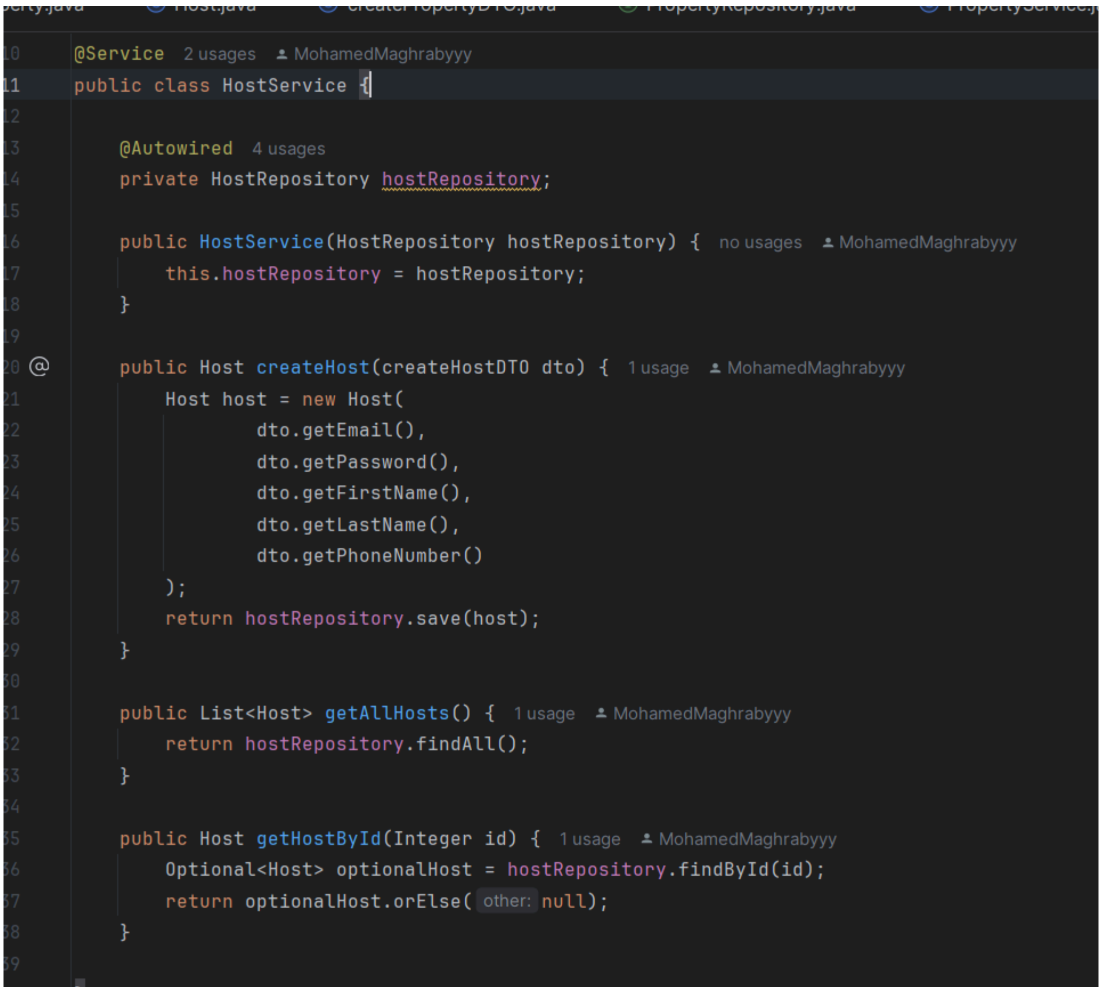


```
@RestController  no usages  MohamedMaghrabyyy
@RequestMapping("/api/host")
public class HostController {

    @Autowired  4 usages
    private HostService hostService;

    public HostController(HostService hostService) {  no usages  MohamedMaghrabyyy
        this.hostService = hostService;
    }

    @PostMapping  no usages  MohamedMaghrabyyy
    public ResponseEntity<Host> createHost(@RequestBody createHostDTO hostDTO) {
        Host savedHost = hostService.createHost(hostDTO);
        return ResponseEntity.status(201).body(savedHost);
    }

    @GetMapping  no usages  MohamedMaghrabyyy
    public ResponseEntity<List<Host>> getAllHosts() {
        List<Host> hosts = hostService.getAllHosts();
        return ResponseEntity.ok(hosts);
    }

    @GetMapping("/{id}")  no usages  MohamedMaghrabyyy
    public ResponseEntity<Host> getHostById(@PathVariable Integer id) {
        Host host = hostService.getHostById(id);
        if (host != null) {
            return ResponseEntity.ok(host);
```

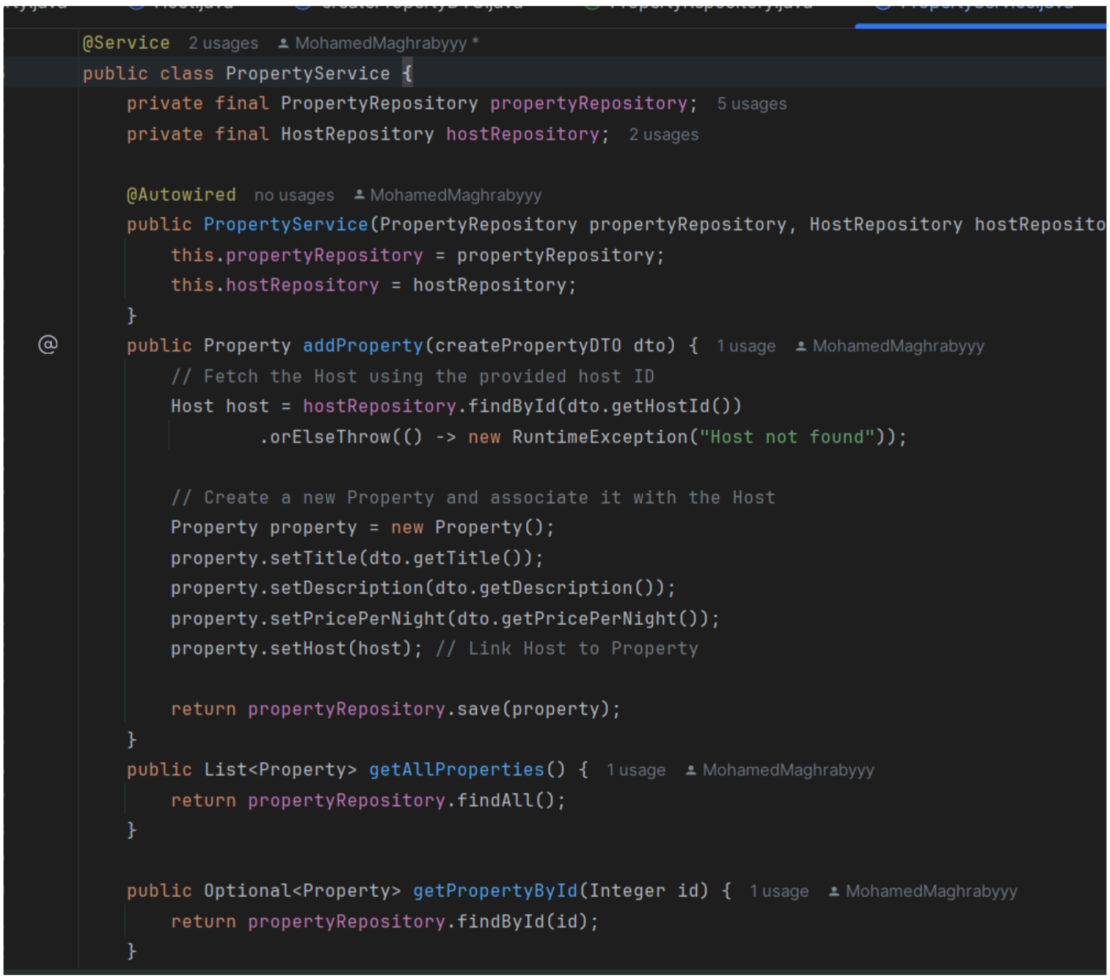


```
@RestController  no usages  MohamedMaghrabyyy
@RequestMapping("/api/property")
public class PropertyController {

    private final PropertyService propertyService;  5 usages

    @Autowired  no usages  MohamedMaghrabyyy
    public PropertyController(PropertyService propertyService) {
        this.propertyService = propertyService;
    }

    @PostMapping  no usages  MohamedMaghrabyyy
    public ResponseEntity<Property> addProperty(@RequestBody createPropertyDTO property) {
        Property savedProperty = propertyService.addProperty(property);
        return ResponseEntity.ok(savedProperty);
    }

    @GetMapping  no usages  MohamedMaghrabyyy
    public ResponseEntity<List<Property>> getAllProperties() {
        List<Property> properties = propertyService.getAllProperties();
        return ResponseEntity.ok(properties);
    }

    @GetMapping("/{id}")  no usages  MohamedMaghrabyyy
    public ResponseEntity<Property> getPropertyById(@PathVariable Integer id) {
        Optional<Property> property = propertyService.getPropertyById(id);
        return property.map(ResponseEntity::ok)
                .orElseGet(() -> ResponseEntity.notFound().build());
    }

    @GetMapping("/host/{hostId}")  no usages  MohamedMaghrabyyy
```

5- Services



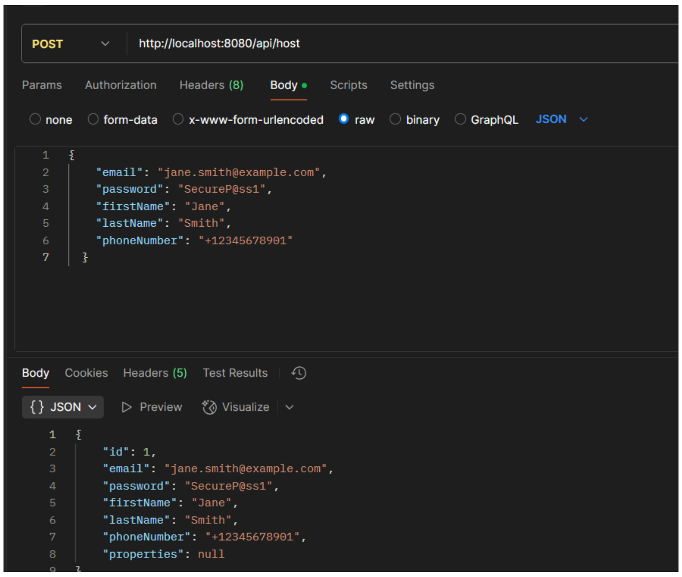
```
@Service  2 usages  MohamedMaghrabyyy
public class HostService {

    @Autowired  4 usages
    private HostRepository hostRepository;

    public HostService(HostRepository hostRepository) {  no usages  MohamedMaghrabyyy
        this.hostRepository = hostRepository;
    }

    public Host createHost(createHostDTO dto) {  1 usage  MohamedMaghrabyyy
        Host host = new Host(
                dto.getEmail(),
                dto.getPassword(),
                dto.getFirstName(),
                dto.getLastName(),
                dto.getPhoneNumber()
        );
        return hostRepository.save(host);
    }

    public List<Host> getAllHosts() {  1 usage  MohamedMaghrabyyy
        return hostRepository.findAll();
    }

    public Host getHostById(Integer id) {  1 usage  MohamedMaghrabyyy
        Optional<Host> optionalHost = hostRepository.findById(id);
        return optionalHost.orElse( other: null);
    }

```



```
@Service  2 usages  MohamedMaghrabyyy *
public class PropertyService {
    private final PropertyRepository propertyRepository;  5 usages
    private final HostRepository hostRepository;  2 usages

    @Autowired  no usages  MohamedMaghrabyyy
    public PropertyService(PropertyRepository propertyRepository, HostRepository hostReposito
        this.propertyRepository = propertyRepository;
        this.hostRepository = hostRepository;
    }
    public Property addProperty(createPropertyDTO dto) {  1 usage  MohamedMaghrabyyy
        // Fetch the Host using the provided host ID
        Host host = hostRepository.findById(dto.getHostId())
                .orElseThrow(() -> new RuntimeException("Host not found"));

        // Create a new Property and associate it with the Host
        Property property = new Property();
        property.setTitle(dto.getTitle());
        property.setDescription(dto.getDescription());
        property.setPricePerNight(dto.getPricePerNight());
        property.setHost(host); // Link Host to Property

        return propertyRepository.save(property);
    }
    public List<Property> getAllProperties() {  1 usage  MohamedMaghrabyyy
        return propertyRepository.findAll();
    }

    public Optional<Property> getPropertyById(Integer id) {  1 usage  MohamedMaghrabyyy
        return propertyRepository.findById(id);
    }
```

6- testing with Postman

1- Create Host endpoint:



POST http://localhost:8080/api/host

Params Authorization Headers (8) Body Scripts Settings

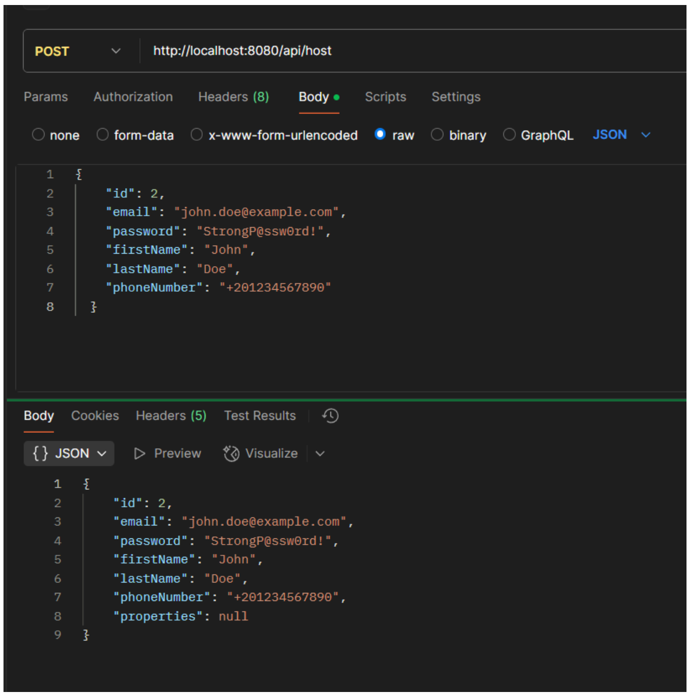
none form-data x-www-form-urlencoded [✓ raw] binary GraphQL JSON

```
{
    "email": "jane.smith@example.com",
    "password": "SecureP@ss1",
    "firstName": "Jane",
    "lastName": "Smith",
    "phoneNumber": "+12345678901"
}
```

Body Cookies Headers (5) Test Results

{} JSON Preview Visualize

```
{
    "id": 1,
    "email": "jane.smith@example.com",
    "password": "SecureP@ss1",
    "firstName": "Jane",
    "lastName": "Smith",
    "phoneNumber": "+12345678901",
    "properties": null
```



POST http://localhost:8080/api/host

Params Authorization Headers (8) Body Scripts Settings

none form-data x-www-form-urlencoded [✓ raw] binary GraphQL JSON

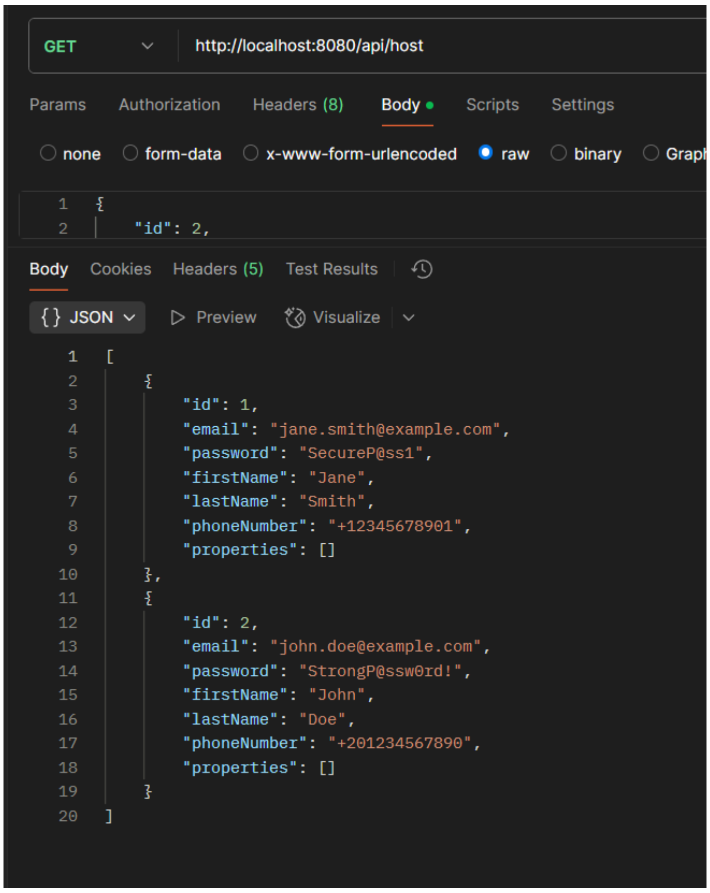
```
{
    "id": 2,
    "email": "john.doe@example.com",
    "password": "StrongP@ssw0rd!",
    "firstName": "John",
    "lastName": "Doe",
    "phoneNumber": "+201234567890"
}
```

Body Cookies Headers (5) Test Results

{} JSON Preview Visualize

```
{
    "id": 2,
    "email": "john.doe@example.com",
    "password": "StrongP@ssw0rd!",
    "firstName": "John",
    "lastName": "Doe",
    "phoneNumber": "+201234567890",
    "properties": null
}
```

2- get all Hosts endpoint:



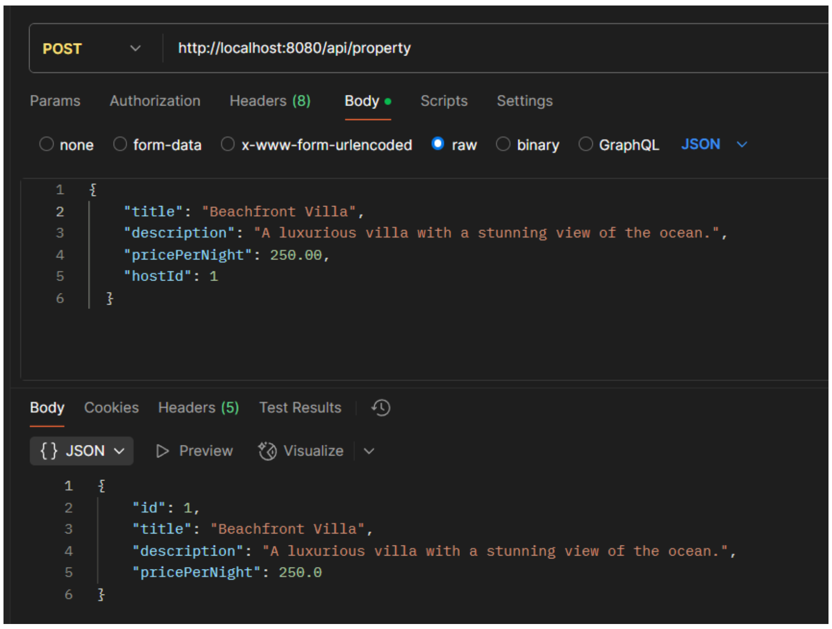
GET http://localhost:8080/api/host

Body: [✓ raw]

```
{
    "id": 2,
```

```
[
    {
        "id": 1,
        "email": "jane.smith@example.com",
        "password": "SecureP@ss1",
        "firstName": "Jane",
        "lastName": "Smith",
        "phoneNumber": "+12345678901",
        "properties": []
    },
    {
        "id": 2,
        "email": "john.doe@example.com",
        "password": "StrongP@ssw0rd!",
        "firstName": "John",
        "lastName": "Doe",
        "phoneNumber": "+201234567890",
        "properties": []
    }
]
```

3- add Property endpoint:



POST http://localhost:8080/api/property

Params Authorization Headers (8) Body Scripts Settings

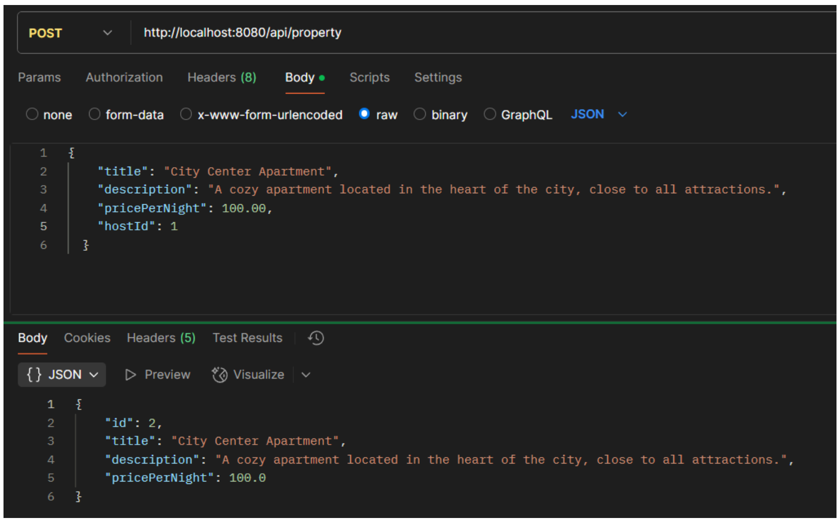
none form-data x-www-form-urlencoded [✓ raw] binary GraphQL JSON

```json
{
    "title": "Beachfront Villa",
    "description": "A luxurious villa with a stunning view of the ocean.",
    "pricePerNight": 250.00,
    "hostId": 1
}
```

Body Cookies Headers (5) Test Results

{} JSON Preview Visualize

```json
{
    "id": 1,
    "title": "Beachfront Villa",
    "description": "A luxurious villa with a stunning view of the ocean.",
    "pricePerNight": 250.0
}
```



POST http://localhost:8080/api/property

Params Authorization Headers (8) Body Scripts Settings

none form-data x-www-form-urlencoded [✓ raw] binary GraphQL JSON

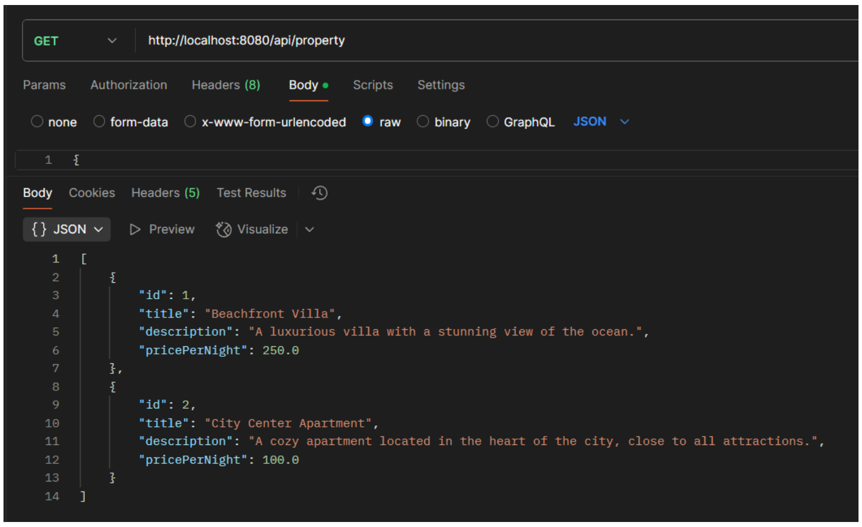
```json
{
    "title": "City Center Apartment",
    "description": "A cozy apartment located in the heart of the city, close to all attractions.",
    "pricePerNight": 100.00,
    "hostId": 1
}
```

Body Cookies Headers (5) Test Results

{} JSON Preview Visualize

```json
{
    "id": 2,
    "title": "City Center Apartment",
    "description": "A cozy apartment located in the heart of the city, close to all attractions.",
    "pricePerNight": 100.0
}
```

4- get all Properties endpoint:

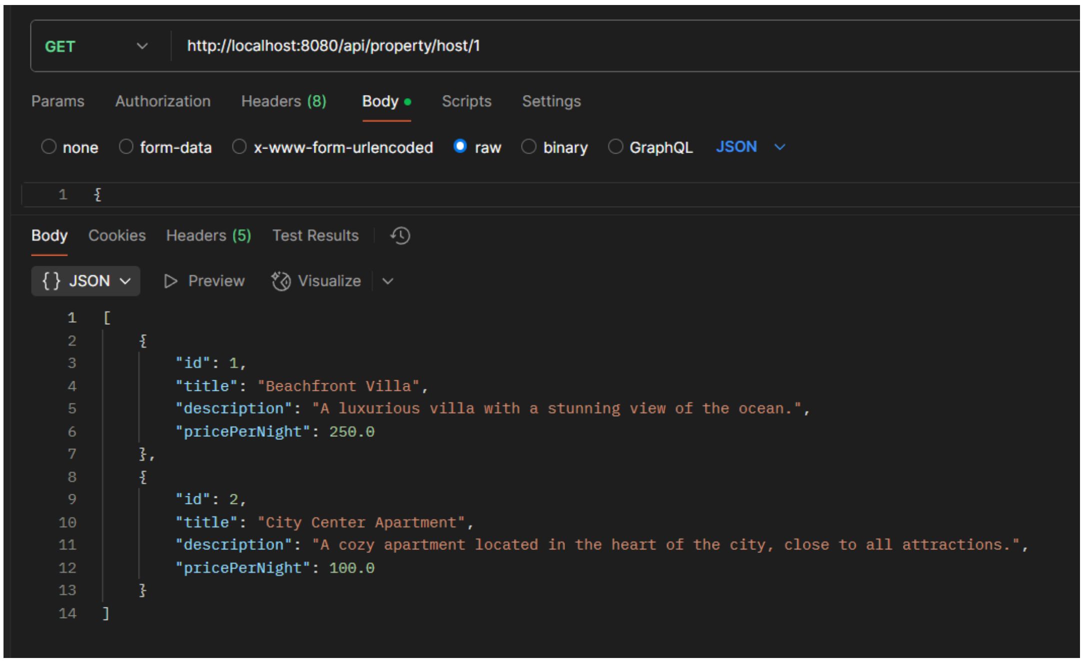


```
GET http://localhost:8080/api/property
```

```
{
```

```
[
    {
        "id": 1,
        "title": "Beachfront Villa",
        "description": "A luxurious villa with a stunning view of the ocean.",
        "pricePerNight": 250.0
    },
    {
        "id": 2,
        "title": "City Center Apartment",
        "description": "A cozy apartment located in the heart of the city, close to all attractions.",
        "pricePerNight": 100.0
    }
]
```

5- get all Properties of a certain user endpoint:



```
GET http://localhost:8080/api/property/host/1
```

```
{
```

```
[
    {
        "id": 1,
        "title": "Beachfront Villa",
        "description": "A luxurious villa with a stunning view of the ocean.",
        "pricePerNight": 250.0
    },
    {
        "id": 2,
        "title": "City Center Apartment",
        "description": "A cozy apartment located in the heart of the city, close to all attractions.",
        "pricePerNight": 100.0
    }
]
```

Github Repo link:
https://github.com/MohamedMaghrabyyy/Aspect-and-Service-Oriented-Programming-Labs/tree/main/lab3