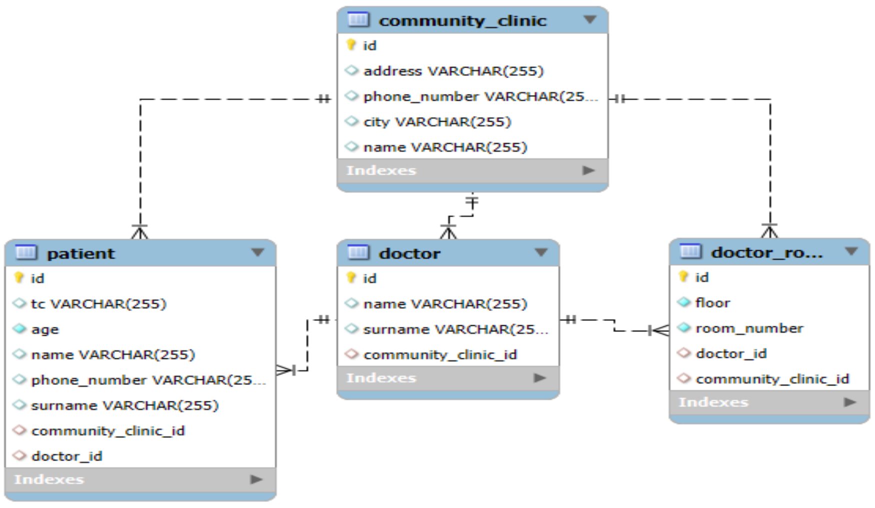
Community Healthcare Management System API

Irmak Kahya

The aim of this project is to manage and control community clinics effectively. This system doesn't just manage the clinics but also holds doctors, patients separate from clinics so that the doctors and patients can be assigned to other clinics in the future, giving a real-world example. The entities are as shown below:

CommunityClinic	Doctor	DoctorRoom	Patient
Long id	Long id	Long id	Long id
String name	String name	int floor	String name
String phoneNumber	String surname	int roomNumber	String surname
String city	CommunityClinic communityClinic	Doctor doctor	String phoneNumber
String address	List<Patient> patients	CommunityClinic communityClinic	int age
List<Doctor> doctors	DoctorRoom doctorRoom		String TC
List<Patient> patients			CommunityClinic communityClinic
List<DoctorRoom> rooms			Doctor doctor

Database Diagram:



Repositories: Each repository contains at least one custom function.

DoctorRoomRepository:

This repository custom functions get rooms by doctor, floor, and clinic.

- findByDoctorId(Long id)
- findByFloor(int floor)
- findByCommunityClinicId(Long id)

DoctorRepository:

- findByCommunityClinicId(Long id): This repository custom function gets doctors by clinic.

CommunityClinicRepository:

- findByCity(String city): This is for getting clinics by city.

PatientRepository:

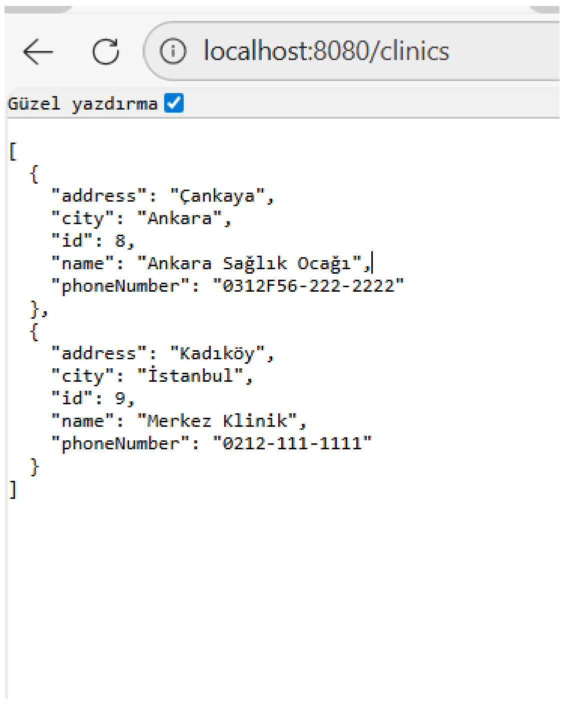
In this repository we get patient by TC and patient list by doctor id, clinic id.

- findByDoctorId(Long id)
- findByCommunityClinicId(Long id)
- findByTC(String tc)

Service Functions:

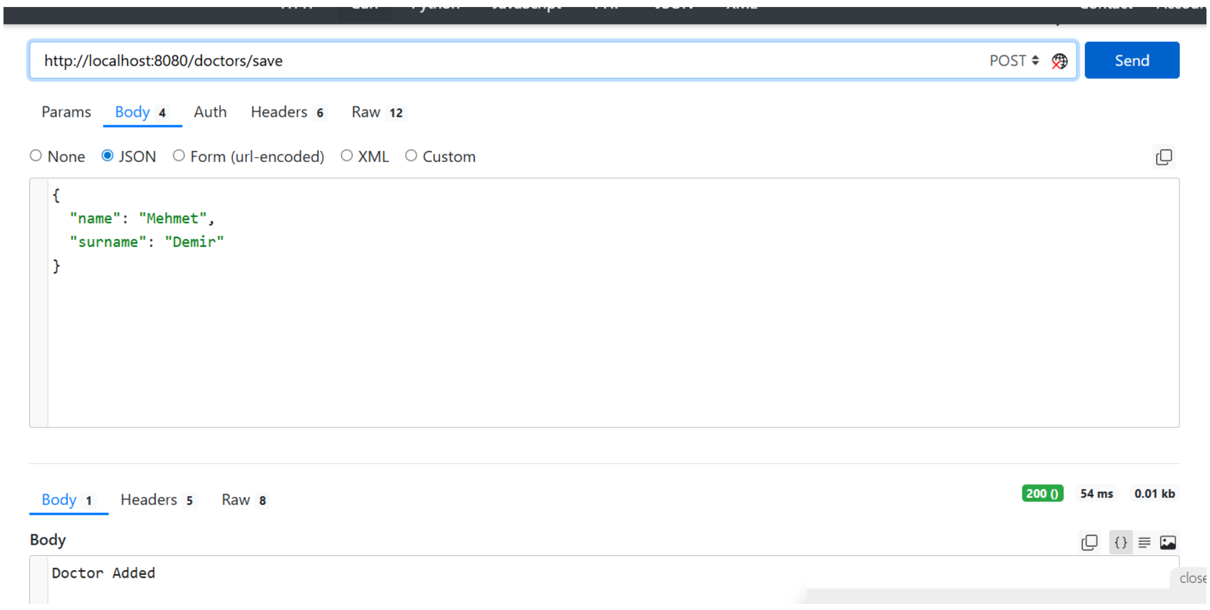
CommunityClinicService	DoctorRoomService	DoctorService	PatientService
getClinics()	getRooms()	getDoctors()	getPatients()
GetAClinic (Long id)	getRoom(Long id)	getDoctor(Long id)	getPatient(Long id)
GetClinicsByCity (String city)	GetRoomByDoctor (Long id)	addDoctor(Doctor d)	getPatientsByDoctorId (Long id)
SaveClinic(Community Clinic clinic)	GetRoomsByFloor (int floor)	DeleteDoctor (Long id)	GetPatientByTC (String tc)
DeleteClinic (Long id)	AddRoom (DoctorRoom r)	GetPatients (Long id)	AddPatient (Patient p)
GetClinicDoctors (Long id)	UpdateRoom (DoctorRoom r)	assignPatient(Long doctorId, Long patientId)	DeletePatient (Long id)
GetClinicPatients (Long id)		assignDoctorRoom(Long doctorId, Long roomId)	UpdatePatient (Patient p)
assignDoctor(Long clinicId, Long doctorId)		removePatient (Long id, Long patientId)	
RemovePatientFromClinic (Long clinicId, Long patientId)		RemoveRoom (Long id,Long roomId)	
removeDoctorFromClinic (Long clinicId, Long doctorId)		UpdateDoctor (Doctor d)	
assignPatient(Long clinicId, Long patientId)			
updateClicic(CommunityClinic c)			
getClinicRooms(Long id)			
assignRoom(Long clinicId, Long roomId)			

In the below, there are screenshots to show how API works.



```
[
  {
    "address": "Çankaya",
    "city": "Ankara",
    "id": 8,
    "name": "Ankara Sağlık Ocağı",
    "phoneNumber": "0312F56-222-2222"
  },
  {
    "address": "Kadıköy",
    "city": "İstanbul",
    "id": 9,
    "name": "Merkez Klinik",
    "phoneNumber": "0212-111-1111"
  }
]
```

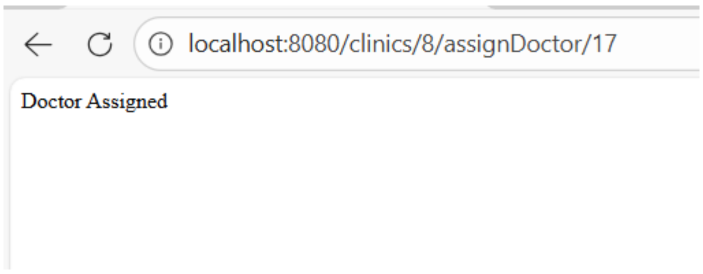
A new doctor is created:



```
{
  "name": "Mehmet",
  "surname": "Demir"
}
```

```
Doctor Added
```

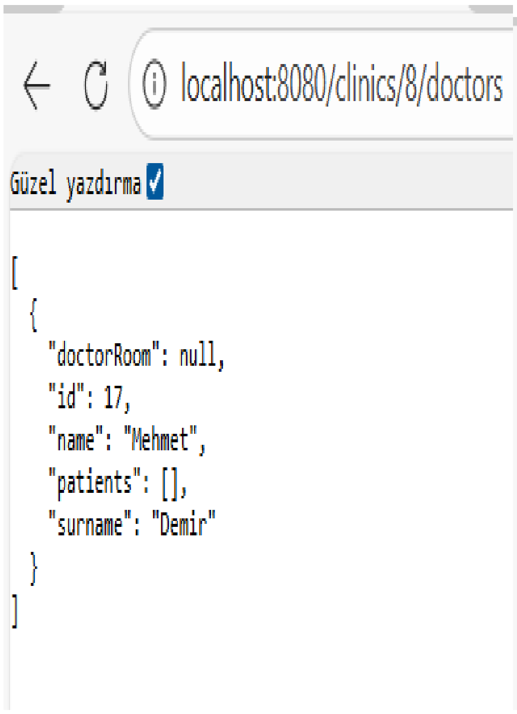
The doctor is assigned to a clinic.



localhost:8080/clinics/8/assignDoctor/17

Doctor Assigned

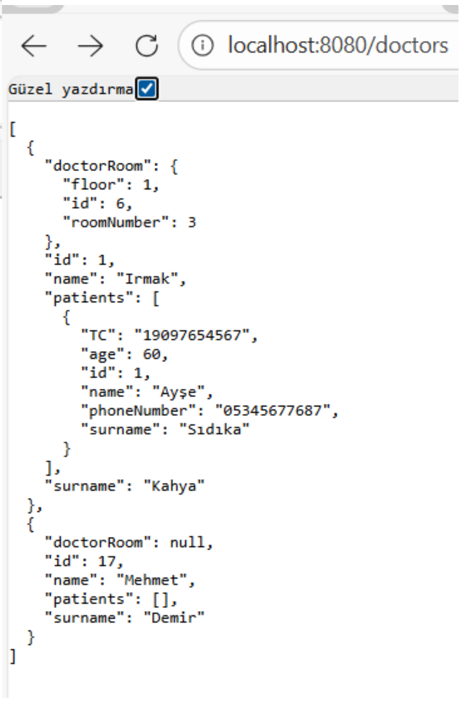
We can list doctors by clinic, or we can list all the doctor records we have in the whole clinic system.



localhost:8080/clinics/8/doctors

Güzel yazdırma ☑

```
[
  {
    "doctorRoom": null,
    "id": 17,
    "name": "Mehmet",
    "patients": [],
    "surname": "Demir"
  }
]
```

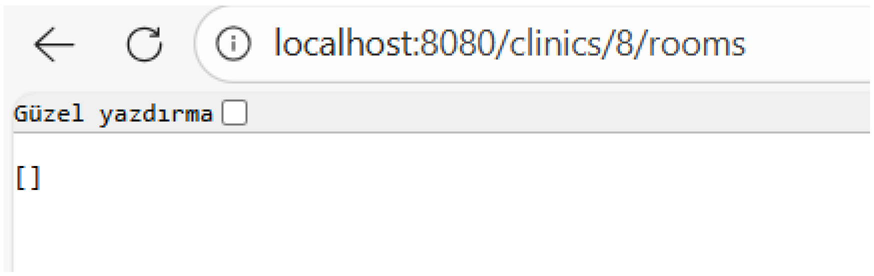


localhost:8080/doctors

Güzel yazdırma ☑

```
[
  {
    "doctorRoom": {
      "floor": 1,
      "id": 6,
      "roomNumber": 3
    },
    "id": 1,
    "name": "Irmak",
    "patients": [
      {
        "TC": "19097654567",
        "age": 60,
        "id": 1,
        "name": "Ayşe",
        "phoneNumber": "05345677687",
        "surname": "Sıdıka"
      }
    ],
    "surname": "Kahya"
  },
  {
    "doctorRoom": null,
    "id": 17,
    "name": "Mehmet",
    "patients": [],
    "surname": "Demir"
  }
]
```

In the system, there is no room for clinic id=8.

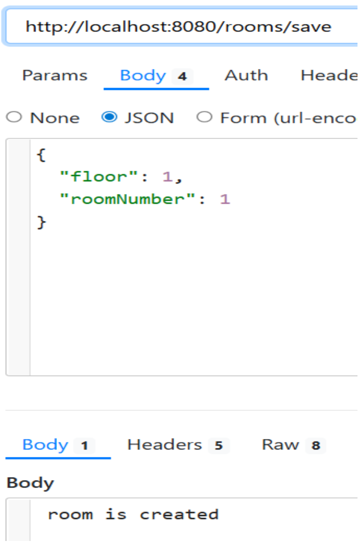


localhost:8080/clinics/8/rooms

Güzel yazdırma ☐

```
[]
```

The room is created and assigned to the clinic.


http://localhost:8080/rooms/save

Params | Body 4 | Auth | Heade

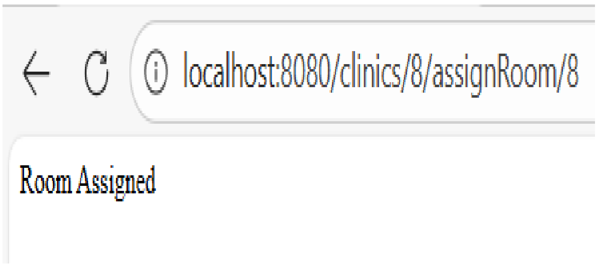
○ None ◉ JSON ○ Form (url-enco

```
{
  "floor": 1,
  "roomNumber": 1
}
```

Body 1 | Headers 5 | Raw 8

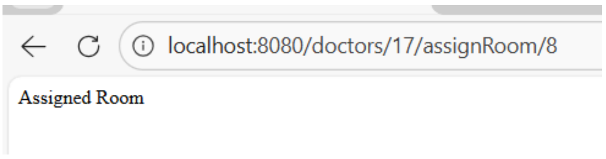
Body

```
room is created
```

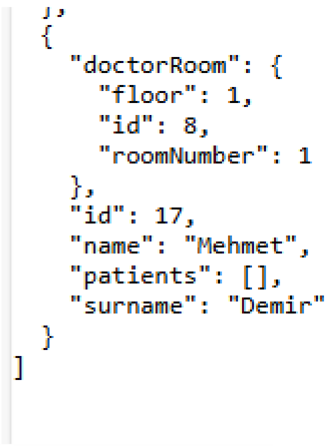

localhost:8080/clinics/8/assignRoom/8

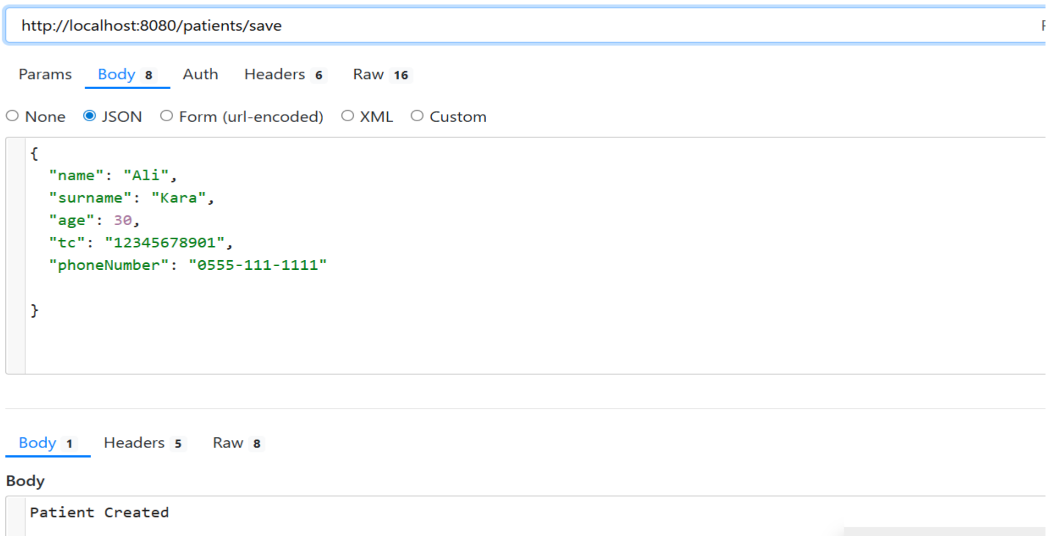
Room Assigned

The room is assigned to the doctor created earlier. New patient is created.


localhost:8080/doctors/17/assignRoom/8

Assigned Room


```
  },
  {
    "doctorRoom": {
      "floor": 1,
      "id": 8,
      "roomNumber": 1
    },
    "id": 17,
    "name": "Mehmet",
    "patients": [],
    "surname": "Demir"
  }
]
```


http://localhost:8080/patients/save

Params | Body 8 | Auth | Headers 6 | Raw 16

○ None ◉ JSON ○ Form (url-encoded) ○ XML ○ Custom

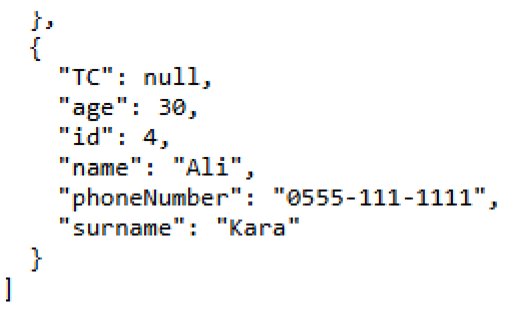
```
{
  "name": "Ali",
  "surname": "Kara",
  "age": 30,
  "tc": "12345678901",
  "phoneNumber": "0555-111-1111"

}
```

Body 1 | Headers 5 | Raw 8

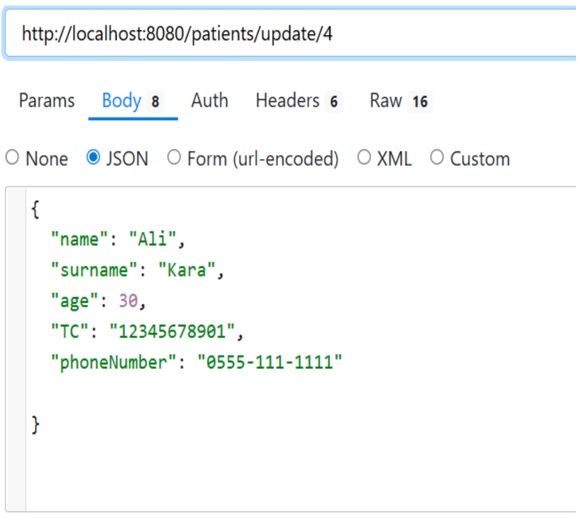
Body

```
Patient Created
```

In the system, updates can be made to whole entities. On the following, updates are made to the patient.



```
  },
  {
    "TC": null,
    "age": 30,
    "id": 4,
    "name": "Ali",
    "phoneNumber": "0555-111-1111",
    "surname": "Kara"
  }
]
```

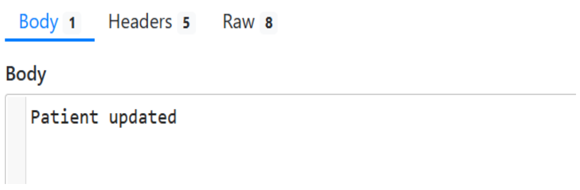


http://localhost:8080/patients/update/4

None [✓ JSON] Form (url-encoded) XML Custom

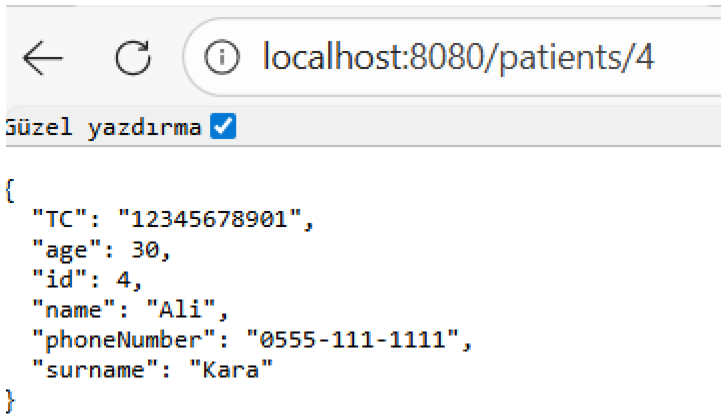
```
{
  "name": "Ali",
  "surname": "Kara",
  "age": 30,
  "TC": "12345678901",
  "phoneNumber": "0555-111-1111"

}
```

Update result:



Body

```
Patient updated
```

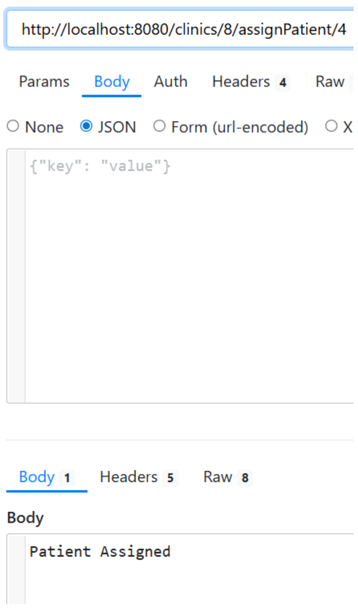


localhost:8080/patients/4

Güzel yazdırma ✓

```
{
  "TC": "12345678901",
  "age": 30,
  "id": 4,
  "name": "Ali",
  "phoneNumber": "0555-111-1111",
  "surname": "Kara"
}
```

The patient is assigned to a clinic and then assigned to a doctor so that the patient will have a community clinic and a family doctor who will take care of him/her.

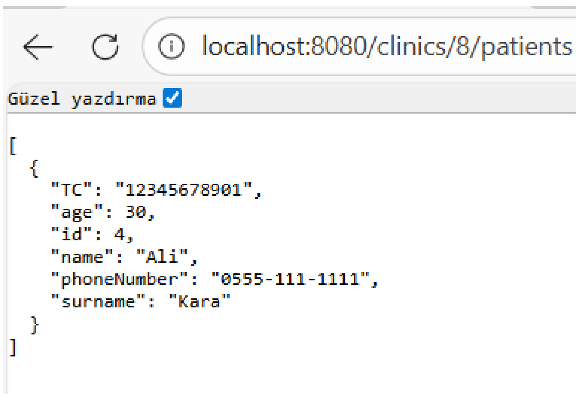

http://localhost:8080/clinics/8/assignPatient/4

None [✓ JSON] Form (url-encoded) X

```
{"key": "value"}
```

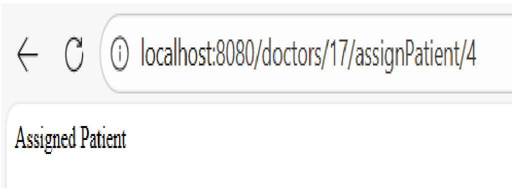
Body

```
Patient Assigned
```

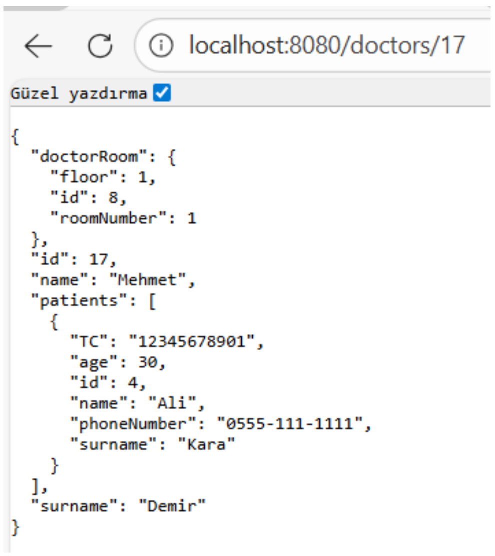

localhost:8080/clinics/8/patients

Güzel yazdırma ✓

```
[
  {
    "TC": "12345678901",
    "age": 30,
    "id": 4,
    "name": "Ali",
    "phoneNumber": "0555-111-1111",
    "surname": "Kara"
  }
]
```


localhost:8080/doctors/17/assignPatient/4

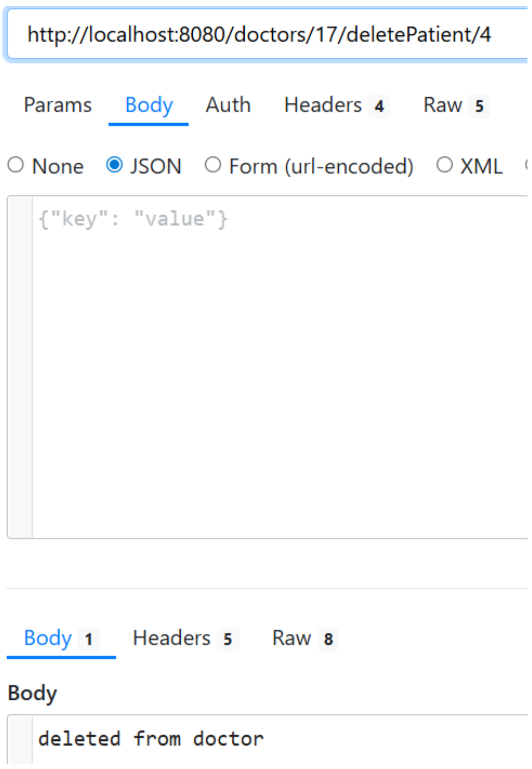
Assigned Patient


localhost:8080/doctors/17

Güzel yazdırma ✓

```
{
  "doctorRoom": {
    "floor": 1,
    "id": 8,
    "roomNumber": 1
  },
  "id": 17,
  "name": "Mehmet",
  "patients": [
    {
      "TC": "12345678901",
      "age": 30,
      "id": 4,
      "name": "Ali",
      "phoneNumber": "0555-111-1111",
      "surname": "Kara"
    }
  ],
  "surname": "Demir"
}
```

The system can delete doctors and patients from clinics without deleting them from the database to assign them to other clinics later, which is more reliable and accurate for real -world examples.


http://localhost:8080/doctors/17/deletePatient/4

Params Body Auth Headers 4 Raw 5

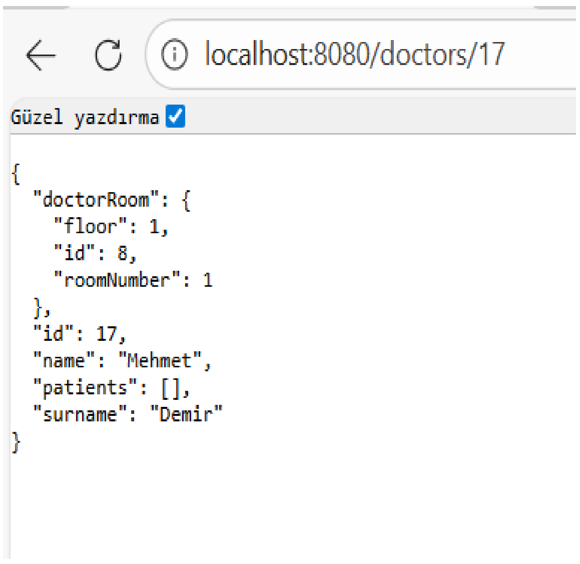
None [✓ JSON] Form (url-encoded) XML

```
{"key": "value"}
```

Body 1 Headers 5 Raw 8

Body

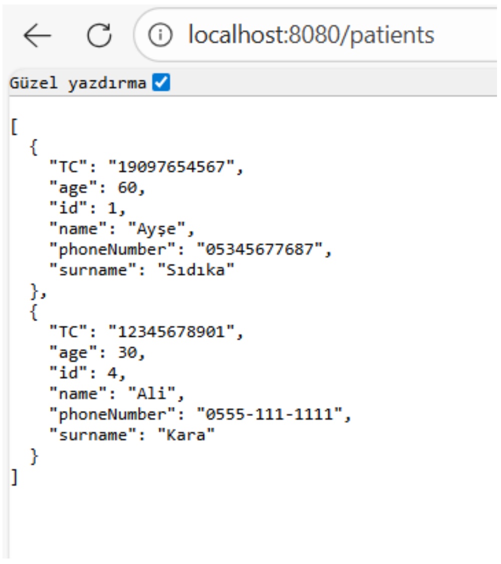
```
deleted from doctor
```


localhost:8080/doctors/17

Güzel yazdırma ✓

```
{
  "doctorRoom": {
    "floor": 1,
    "id": 8,
    "roomNumber": 1
  },
  "id": 17,
  "name": "Mehmet",
  "patients": [],
  "surname": "Demir"
}
```

T

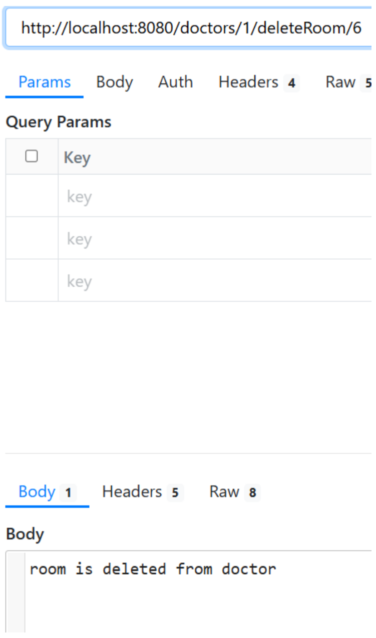

localhost:8080/patients

Güzel yazdırma ✓

```
[
  {
    "TC": "19097654567",
    "age": 60,
    "id": 1,
    "name": "Ayşe",
    "phoneNumber": "05345677687",
    "surname": "Sıdıka"
  },
  {
    "TC": "12345678901",
    "age": 30,
    "id": 4,
    "name": "Ali",
    "phoneNumber": "0555-111-1111",
    "surname": "Kara"
  }
]
```

As shown, the patient is still in our database while it is no longer a patient of the doctor with id=17 so that we can assign this patient to other doctors.

In this system, there is a function to remove room from doctor which empties the room, then a new doctor can be assigned to a room by assignDoctorRoom function.



http://localhost:8080/doctors/1/deleteRoom/6

Params Body Auth Headers 4 Raw 5

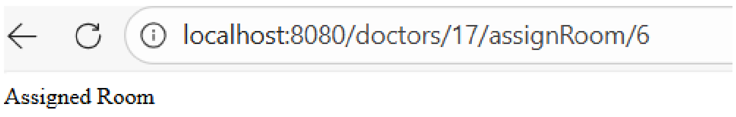
Query Params

	Key
	key
	key
	key

Body 1 Headers 5 Raw 8

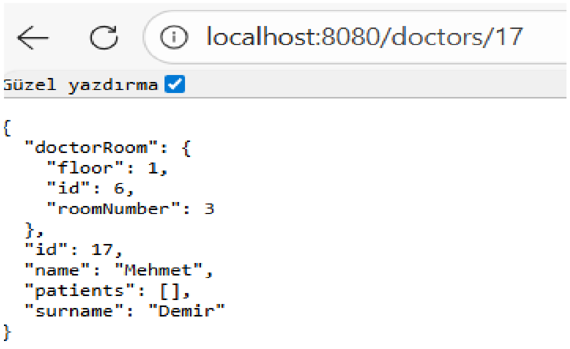
Body

```
room is deleted from doctor
```



localhost:8080/doctors/17/assignRoom/6

Assigned Room

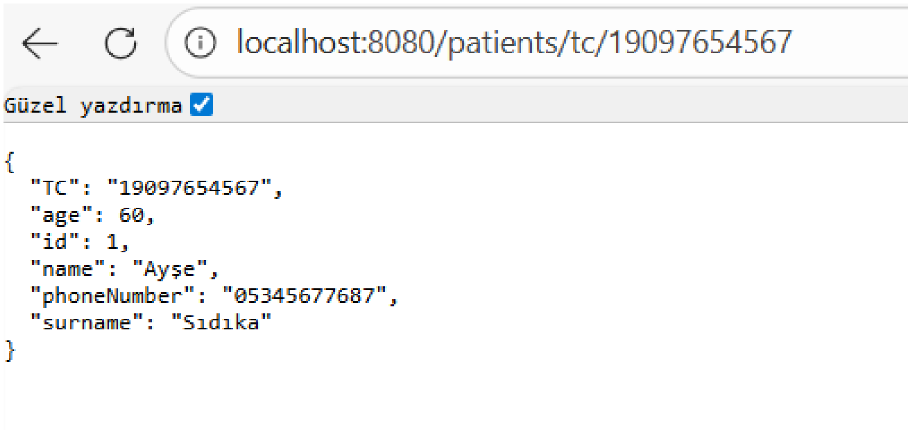


localhost:8080/doctors/17

Güzel yazdırma ✓

```
{
  "doctorRoom": {
    "floor": 1,
    "id": 6,
    "roomNumber": 3
  },
  "id": 17,
  "name": "Mehmet",
  "patients": [],
  "surname": "Demir"
}
```

GetPatientByTC (String tc):



localhost:8080/patients/tc/19097654567

Güzel yazdırma ✓

```
{
  "TC": "19097654567",
  "age": 60,
  "id": 1,
  "name": "Ayşe",
  "phoneNumber": "05345677687",
  "surname": "Sıdıka"
}
```

In conclusion, this community healthcare management system API has four entities, well-structured service and controllers, and repositories with custom functions.