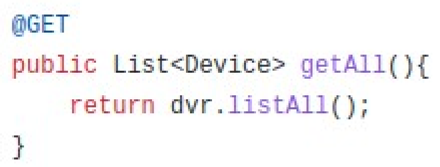
Sew Test Verbesserung

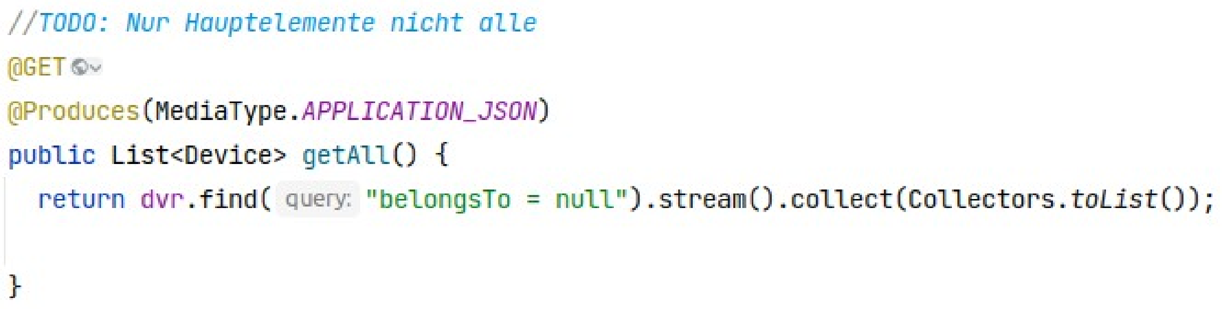
GET /api/device

Zuvor:



```java
@GET
public List<Device> getAll(){
    return dvr.listAll();
}
```

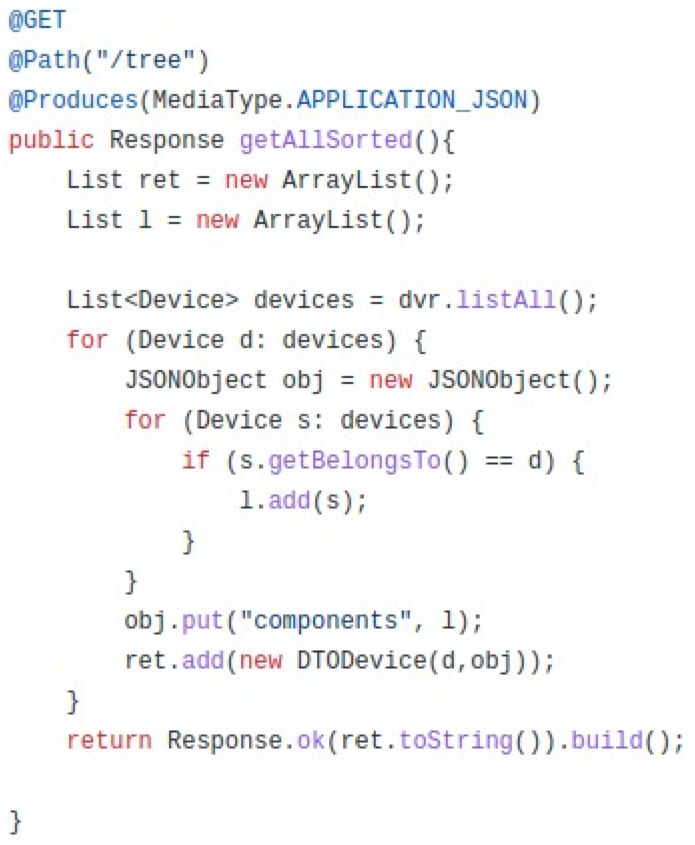
Verbessert:



```java
//TODO: Nur Hauptelemente nicht alle
@GET
@Produces(MediaType.APPLICATION_JSON)
public List<Device> getAll() {
  return dvr.find( query: "belongsTo = null").stream().collect(Collectors.toList());

}
```

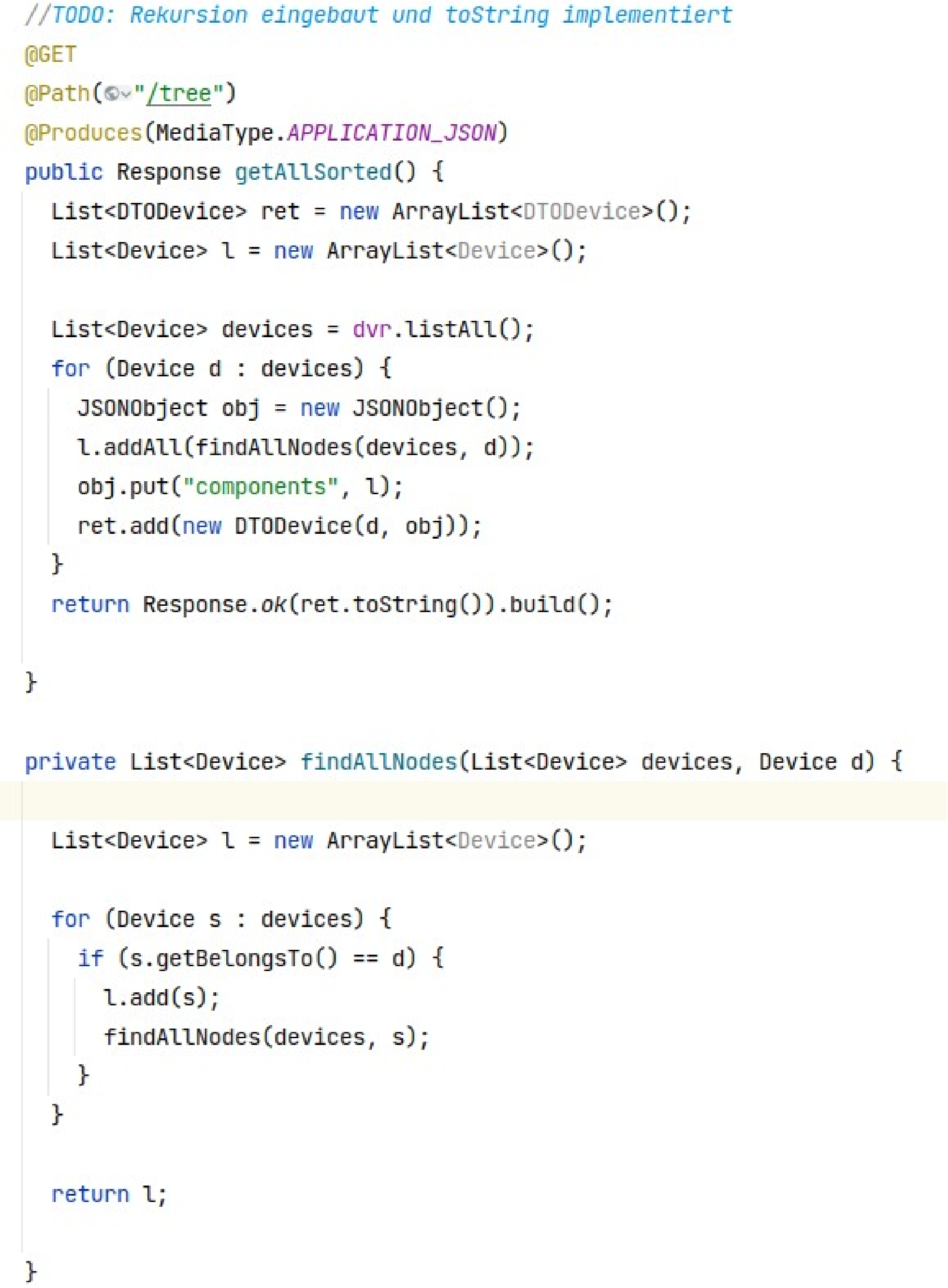
GET /api/device/tree

Zuvor:



```java
@GET
@Path("/tree")
@Produces(MediaType.APPLICATION_JSON)
public Response getAllSorted(){
    List ret = new ArrayList();
    List l = new ArrayList();

    List<Device> devices = dvr.listAll();
    for (Device d: devices) {
        JSONObject obj = new JSONObject();
        for (Device s: devices) {
            if (s.getBelongsTo() == d) {
                l.add(s);
            }
        }
        obj.put("components", l);
        ret.add(new DTODevice(d,obj));
    }
    return Response.ok(ret.toString()).build();

}
```

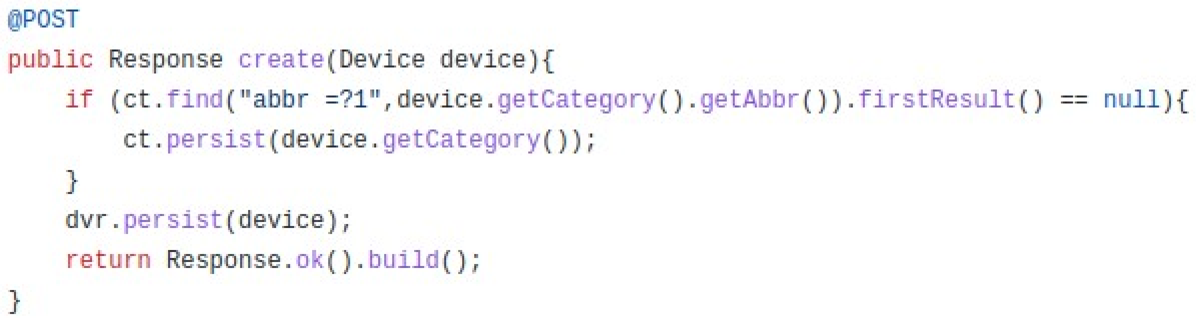
Verbessert:



```java
//TODO: Rekursion eingebaut und toString implementiert
@GET
@Path("/tree")
@Produces(MediaType.APPLICATION_JSON)
public Response getAllSorted() {
    List<DTODevice> ret = new ArrayList<DTODevice>();
    List<Device> l = new ArrayList<Device>();

    List<Device> devices = dvr.listAll();
    for (Device d : devices) {
        JSONObject obj = new JSONObject();
        l.addAll(findAllNodes(devices, d));
        obj.put("components", l);
        ret.add(new DTODevice(d, obj));
    }
    return Response.ok(ret.toString()).build();

}

private List<Device> findAllNodes(List<Device> devices, Device d) {

    List<Device> l = new ArrayList<Device>();

    for (Device s : devices) {
        if (s.getBelongsTo() == d) {
            l.add(s);
            findAllNodes(devices, s);
        }
    }

    return l;

}
```

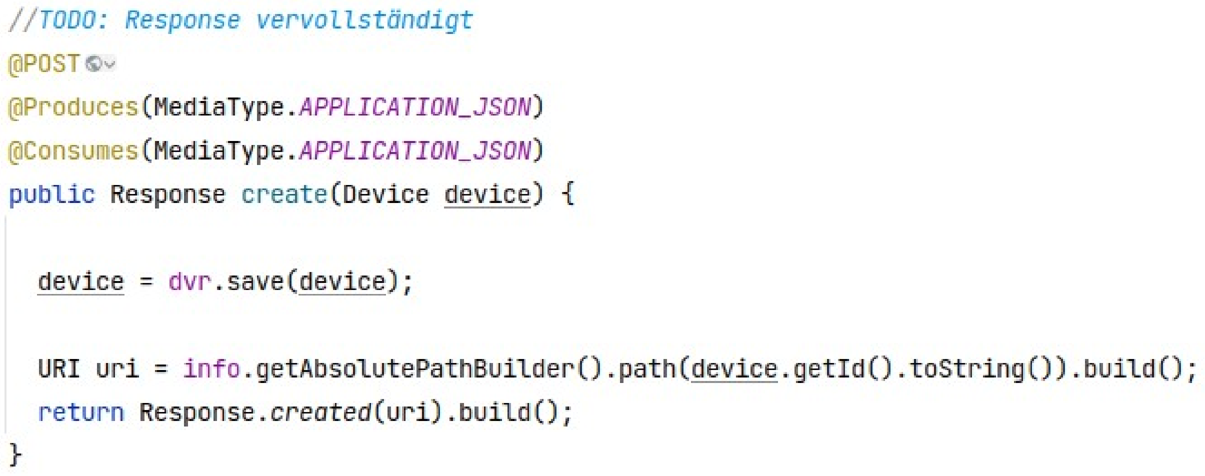
POST /api/device

Zuvor:



```
@POST
public Response create(Device device){
    if (ct.find("abbr =?1",device.getCategory().getAbbr()).firstResult() == null){
        ct.persist(device.getCategory());
    }
    dvr.persist(device);
    return Response.ok().build();
}
```

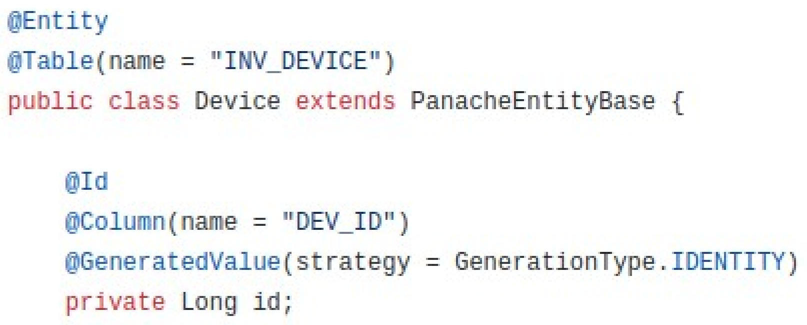
Verbessert:



```
//TODO: Response vervollständigt
@POST
@Produces(MediaType.APPLICATION_JSON)
@Consumes(MediaType.APPLICATION_JSON)
public Response create(Device device) {

    device = dvr.save(device);

    URI uri = info.getAbsolutePathBuilder().path(device.getId().toString()).build();
    return Response.created(uri).build();
}
```

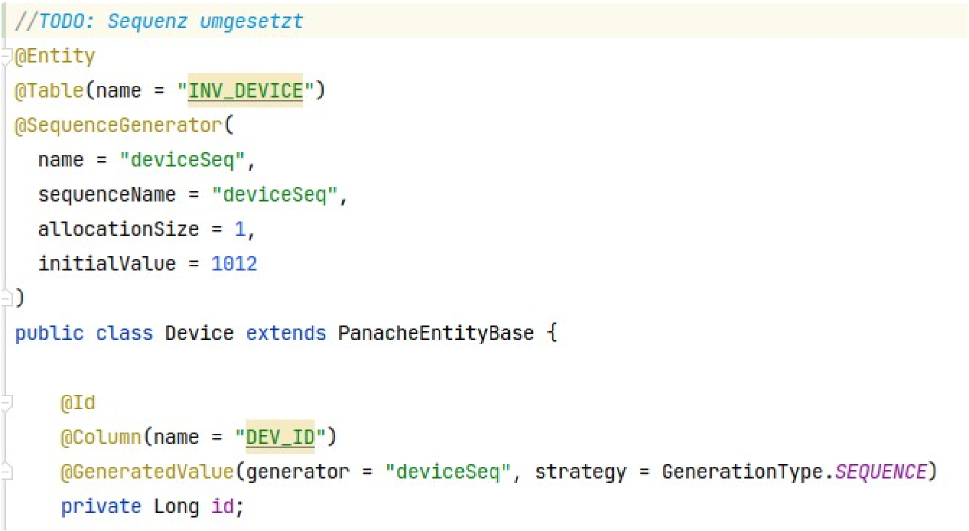
Device Entity

Zuvor:



```
@Entity
@Table(name = "INV_DEVICE")
public class Device extends PanacheEntityBase {

    @Id
    @Column(name = "DEV_ID")
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
```

Verbessert:



```
//TODO: Sequenz umgesetzt
@Entity
@Table(name = "INV_DEVICE")
@SequenceGenerator(
  name = "deviceSeq",
  sequenceName = "deviceSeq",
  allocationSize = 1,
  initialValue = 1012
)
public class Device extends PanacheEntityBase {

    @Id
    @Column(name = "DEV_ID")
    @GeneratedValue(generator = "deviceSeq", strategy = GenerationType.SEQUENCE)
    private Long id;
```

Außerdem wurden bei den Repositories die @Transactional entfernt.