UNIVERSIDAD DE LAS FUERZAS ARMADAS – ESPE SEDE LATACUNGA

TEMA

TRACCAR: GROUPS

PARA QUÉ SIRVE ?

En el sistema Traccar, el manejo de grupos es una funcionalidad clave que permite organizar y gestionar colecciones de dispositivos o entidades relacionadas. La API de grupos proporciona un conjunto de endpoints para realizar operaciones CRUD (Crear, Leer, Actualizar, Eliminar) sobre los grupos, facilitando así la administración eficiente de estos elementos dentro del sistema.

Ruta de Acceso a la Documentación del API de Grupos

Para documentar el API de Grupos en Traccar, es importante saber dónde se encuentra la implementación de esta funcionalidad en el código fuente del proyecto. La clase principal que gestiona las operaciones sobre los grupos a través de la API es GroupResource.java. Esta clase se encuentra en la siguiente ruta dentro del repositorio de código:

css

\DockerTraccar\server\src\main\java\org\traccar\api\resource\GroupResource.java

Esta ruta indica que GroupResource.java forma parte del módulo del servidor de Traccar y está ubicada en el paquete org.traccar.api.resource. Este archivo contiene la implementación de los métodos que permiten interactuar con los grupos mediante solicitudes HTTP, ofreciendo funcionalidades como la creación de nuevos grupos, la modificación de grupos existentes, la eliminación de grupos y la recuperación de información sobre grupos.

Propósito de GroupResource.java

La clase GroupResource extiende la funcionalidad de SimpleObjectResource, lo que le permite manipular objetos de tipo Group. Al implementar esta clase, se asegura que las operaciones relacionadas con los grupos sean accesibles a través de la API, proporcionando una interfaz RESTful para la gestión de grupos en el sistema Traccar. Esta clase maneja las anotaciones de JAX-RS necesarias para definir los endpoints y el manejo de formatos de datos, garantizando la interoperabilidad y el acceso seguro a los recursos del sistema.

LA RUTA DE ACCESO PARA DOCUMENTAR EL API DE GRUP

\DockerTraccar\server\src\main\java\org\traccar\api\resource\GroupResource.java

```
package org.traccar.api.resource;

import org.traccar.api.SimpleObjectResource;
import org.traccar.model.Group;
```

```
import jakarta.ws.rs.Consumes;
import jakarta.ws.rs.Path;
import jakarta.ws.rs.Produces;
import jakarta.ws.rs.core.MediaType;

@Path("groups")
@Produces(MediaType.APPLICATION_JSON)
@Consumes(MediaType.APPLICATION_JSON)
public class GroupResource extends SimpleObjectResource<Group> {

    public GroupResource() {
        super(Group.class);
    }

}
```

Propósito: Esta clase es parte de la API de Traccar y se utiliza como una clase base genérica para los recursos API que manejan operaciones CRUD (Crear, Leer, Actualizar, Eliminar) sobre objetos simples.
Función en el código: En este caso, GroupResource extiende de SimpleObjectResource<Group>, lo que significa que hereda las capacidades para manejar objetos del tipo Group, como crear, leer, actualizar o eliminar grupos.

import org.traccar.api.SimpleObjectResource;

```
package org.traccar.api;

import org.traccar.model.BaseModel;
import org.traccar.model.User;
import org.traccar.storage.StorageException;
import org.traccar.storage.query.Columns;
import org.traccar.storage.query.Condition;
import org.traccar.storage.query.Request;

import jakarta.ws.rs.GET;
import jakarta.ws.rs.QueryParam;
import java.util.Collection;
import java.util.LinkedList;

public class SimpleObjectResource<T extends BaseModel> extends
BaseObjectResource<T> {

    public SimpleObjectResource(Class<T> baseClass) {
        super(baseClass);
    }

    @GET
```

```
    public Collection<T> get(
            @QueryParam("all") boolean all, @QueryParam("userId") long userId) throws
StorageException {

        var conditions = new LinkedList<Condition>();

        if (all) {
            if (permissionsService.notAdmin(getUserId())) {
                conditions.add(new Condition.Permission(User.class, getUserId(), baseClass));
            }
        } else {
            if (userId == 0) {
                userId = getUserId();
            } else {
                permissionsService.checkUser(getUserId(), userId);
            }
            conditions.add(new Condition.Permission(User.class, userId, baseClass));
        }

        return storage.getObjects(baseClass, new Request(new Columns.All(),
Condition.merge(conditions)));
    }

}
```

org.traccar.model.BaseModel: Clase base para todos los modelos en Traccar, como User o Group. Los objetos en SimpleObjectResource deben heredar de esta clase.
org.traccar.model.User: Modelo que representa a un usuario en Traccar. Se utiliza para gestionar permisos de acceso.
org.traccar.storage.StorageException: Excepción que se lanza cuando ocurre un error al interactuar con la base de datos.
org.traccar.storage.query.Columns: Clase que define qué columnas de datos se recuperan en una consulta a la base de datos.
org.traccar.storage.query.Condition: Clase utilizada para construir condiciones (filtros) en las consultas a la base de datos, como permisos y filtros por usuario.
org.traccar.storage.query.Request: Encapsula una consulta completa a la base de datos, incluyendo columnas y condiciones.

import org.traccar.model.BaseModel;

```
package org.traccar.model;

public class BaseModel {

    private long id;
```

```
    public long getId() {
        return id;
    }

    public void setId(long id) {
        this.id = id;
    }

}
```

Propósito: BaseModel es una clase base para otros modelos en Traccar. Proporciona un campo id que sirve como identificador único para los objetos que heredan de esta clase.
Métodos:
getId(): Devuelve el valor del identificador (id).
setId(long id): Establece el valor del identificador (id).

import org.traccar.model.User;

```
package org.traccar.model;

import com.fasterxml.jackson.annotation.JsonIgnore;

import org.apache.commons.lang3.builder.EqualsBuilder;
import org.traccar.storage.QueryIgnore;
import org.traccar.helper.Hashing;
import org.traccar.storage.StorageName;

import java.util.Date;
import java.util.HashMap;

@StorageName("tc_users")
public class User extends ExtendedModel implements UserRestrictions, Disableable {

    private String name;

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    private String login;

    public String getLogin() {
        return login;
```

```
}

public void setLogin(String login) {
    this.login = login;
}

private String email;

public String getEmail() {
    return email;
}

public void setEmail(String email) {
    this.email = email.trim();
}

private String phone;

public String getPhone() {
    return phone;
}

public void setPhone(String phone) {
    this.phone = phone != null ? phone.trim() : null;
}

private boolean readonly;

@Override
public boolean getReadonly() {
    return readonly;
}

public void setReadonly(boolean readonly) {
    this.readonly = readonly;
}

private boolean administrator;

@QueryIgnore
@JsonIgnore
public boolean getManager() {
    return userLimit != 0;
}

public boolean getAdministrator() {
    return administrator;
}

public void setAdministrator(boolean administrator) {
    this.administrator = administrator;
}

private String map;
```

```
public String getMap() {
    return map;
}

public void setMap(String map) {
    this.map = map;
}

private double latitude;

public double getLatitude() {
    return latitude;
}

public void setLatitude(double latitude) {
    this.latitude = latitude;
}

private double longitude;

public double getLongitude() {
    return longitude;
}

public void setLongitude(double longitude) {
    this.longitude = longitude;
}

private int zoom;

public int getZoom() {
    return zoom;
}

public void setZoom(int zoom) {
    this.zoom = zoom;
}

private String coordinateFormat;

public String getCoordinateFormat() {
    return coordinateFormat;
}

public void setCoordinateFormat(String coordinateFormat) {
    this.coordinateFormat = coordinateFormat;
}

private boolean disabled;

@Override
public boolean getDisabled() {
    return disabled;
```

```
}

@Override
public void setDisabled(boolean disabled) {
    this.disabled = disabled;
}

private Date expirationTime;

@Override
public Date getExpirationTime() {
    return expirationTime;
}

@Override
public void setExpirationTime(Date expirationTime) {
    this.expirationTime = expirationTime;
}

private int deviceLimit;

public int getDeviceLimit() {
    return deviceLimit;
}

public void setDeviceLimit(int deviceLimit) {
    this.deviceLimit = deviceLimit;
}

private int userLimit;

public int getUserLimit() {
    return userLimit;
}

public void setUserLimit(int userLimit) {
    this.userLimit = userLimit;
}

private boolean deviceReadonly;

@Override
public boolean getDeviceReadonly() {
    return deviceReadonly;
}

public void setDeviceReadonly(boolean deviceReadonly) {
    this.deviceReadonly = deviceReadonly;
}

private boolean limitCommands;

@Override
public boolean getLimitCommands() {
```

```
    return limitCommands;
}

public void setLimitCommands(boolean limitCommands) {
    this.limitCommands = limitCommands;
}

private boolean disableReports;

@Override
public boolean getDisableReports() {
    return disableReports;
}

public void setDisableReports(boolean disableReports) {
    this.disableReports = disableReports;
}

private boolean fixedEmail;

@Override
public boolean getFixedEmail() {
    return fixedEmail;
}

public void setFixedEmail(boolean fixedEmail) {
    this.fixedEmail = fixedEmail;
}

private String poiLayer;

public String getPoiLayer() {
    return poiLayer;
}

public void setPoiLayer(String poiLayer) {
    this.poiLayer = poiLayer;
}

private String totpKey;

public String getTotpKey() {
    return totpKey;
}

public void setTotpKey(String totpKey) {
    this.totpKey = totpKey;
}

private boolean temporary;

public boolean getTemporary() {
    return temporary;
}
```

```
    public void setTemporary(boolean temporary) {
        this.temporary = temporary;
    }

    @QueryIgnore
    public String getPassword() {
        return null;
    }

    @QueryIgnore
    public void setPassword(String password) {
        if (password != null && !password.isEmpty()) {
            Hashing.HashingResult hashingResult = Hashing.createHash(password);
            hashedPassword = hashingResult.getHash();
            salt = hashingResult.getSalt();
        }
    }

    private String hashedPassword;

    @JsonIgnore
    @QueryIgnore
    public String getHashedPassword() {
        return hashedPassword;
    }

    @QueryIgnore
    public void setHashedPassword(String hashedPassword) {
        this.hashedPassword = hashedPassword;
    }

    private String salt;

    @JsonIgnore
    @QueryIgnore
    public String getSalt() {
        return salt;
    }

    @QueryIgnore
    public void setSalt(String salt) {
        this.salt = salt;
    }

    public boolean isPasswordValid(String password) {
        return Hashing.validatePassword(password, hashedPassword, salt);
    }

    public boolean compare(User other, String... exclusions) {
        if (!EqualsBuilder.reflectionEquals(this, other, "attributes", "hashedPassword", "salt"))
{
            return false;
        }
```

```
        var thisAttributes = new HashMap<>(getAttributes());
        var otherAttributes = new HashMap<>(other.getAttributes());
        for (String exclusion : exclusions) {
            thisAttributes.remove(exclusion);
            otherAttributes.remove(exclusion);
        }
        return thisAttributes.equals(otherAttributes);
    }

}
```

org.traccar.model.User:
Clase que representa a un usuario en Traccar. Contiene información como nombre, login, email, permisos, y configuraciones de usuario.

com.fasterxml.jackson.annotation.JsonIgnore:
Anotación que evita que un campo se serialice en JSON. Se usa en campos sensibles, como contraseñas (hashedPassword, salt).

org.apache.commons.lang3.builder.EqualsBuilder:
Utilizado para comparar objetos de manera eficiente, omitiendo ciertos campos, como contraseñas.

org.traccar.storage.QueryIgnore:
Anotación que indica que un campo debe ser ignorado en las consultas a la base de datos, evitando que sea incluido en las operaciones de búsqueda o almacenamiento.

org.traccar.helper.Hashing:
Clase de utilidad para gestionar el hashing y la verificación de contraseñas, proporcionando seguridad al almacenar las contraseñas.

org.traccar.storage.StorageName:
Anotación que asigna un nombre específico de tabla en la base de datos para la clase User (en este caso, "tc_users").

import org.traccar.storage.StorageException;

```
package org.traccar.storage;
```

```
public class StorageException extends Exception {

    public StorageException(String message) {
        super(message);
    }

    public StorageException(Throwable cause) {
        super(cause);
    }

    public StorageException(String message, Throwable cause) {
        super(message, cause);
    }

}
```

StorageException se utiliza para capturar y manejar errores que ocurren en las operaciones de almacenamiento, proporcionando información útil sobre el error mediante mensajes y/o causas encapsuladas.

package org.traccar.storage;

```
package org.traccar.storage;

public class StorageException extends Exception {

    public StorageException(String message) {
        super(message);
    }

    public StorageException(Throwable cause) {
        super(cause);
    }

    public StorageException(String message, Throwable cause) {
        super(message, cause);
    }

}
```

El paquete org.traccar.storage en Traccar se utiliza para manejar todo lo relacionado con el almacenamiento de datos en la plataforma. Este paquete incluye clases y excepciones que facilitan la interacción con las bases de datos y otros sistemas de almacenamiento.

import org.traccar.storage.query.Columns;

```
package org.traccar.storage.query;

import org.traccar.storage.QueryIgnore;

import java.beans.Introspector;
import java.lang.reflect.Method;
import java.util.Arrays;
import java.util.LinkedList;
import java.util.List;
import java.util.Set;
import java.util.stream.Collectors;

public abstract class Columns {

    public abstract List<String> getColumns(Class<?> clazz, String type);

    protected List<String> getAllColumns(Class<?> clazz, String type) {
        List<String> columns = new LinkedList<>();
        Method[] methods = clazz.getMethods();
        for (Method method : methods) {
            int parameterCount = type.equals("set") ? 1 : 0;
            if (method.getName().startsWith(type) && method.getParameterTypes().length ==
parameterCount
                    && !method.isAnnotationPresent(QueryIgnore.class)
                    && !method.getName().equals("getClass")) {
                columns.add(Introspector.decapitalize(method.getName().substring(3)));
            }
        }
        return columns;
    }

    public static class All extends Columns {
        @Override
        public List<String> getColumns(Class<?> clazz, String type) {
            return getAllColumns(clazz, type);
        }
    }

    public static class Include extends Columns {
        private final List<String> columns;

        public Include(String... columns) {
            this.columns = Arrays.stream(columns).collect(Collectors.toList());
        }

        @Override
        public List<String> getColumns(Class<?> clazz, String type) {
            return columns;
        }
    }

    public static class Exclude extends Columns {
```

```
        private final Set<String> columns;

        public Exclude(String... columns) {
            this.columns = Arrays.stream(columns).collect(Collectors.toSet());
        }

        @Override
        public List<String> getColumns(Class<?> clazz, String type) {
            return getAllColumns(clazz, type).stream()
                    .filter(column -> !columns.contains(column))
                    .collect(Collectors.toList());
        }
    }

}
```

package org.traccar.storage.query

Propósito: Este paquete está diseñado para gestionar la creación y manipulación de consultas a la base de datos en Traccar. Contiene clases y métodos que ayudan a definir qué columnas (campos) se deben incluir o excluir en las consultas SQL generadas, basándose en la reflexión de los métodos de los modelos de datos.

import org.traccar.storage.QueryIgnore

Propósito: QueryIgnore es una anotación utilizada para marcar métodos o campos de un modelo que deben ser ignorados al generar consultas a la base de datos. Esto es útil cuando ciertos datos no deben ser considerados en las operaciones de consulta, por ejemplo, campos sensibles o irrelevantes.

import org.traccar.storage.query.Condition;

```
package org.traccar.storage.query;

import org.traccar.model.GroupedModel;

import java.util.List;

public interface Condition {

    static Condition merge(List<Condition> conditions) {
        Condition result = null;
        var iterator = conditions.iterator();
        if (iterator.hasNext()) {
            result = iterator.next();
            while (iterator.hasNext()) {
                result = new Condition.And(result, iterator.next());
            }
        }
```

```
        return result;
    }

    class Equals extends Compare {
        public Equals(String column, Object value) {
            super(column, "=", column, value);
        }
    }

    class Compare implements Condition {
        private final String column;
        private final String operator;
        private final String variable;
        private final Object value;

        public Compare(String column, String operator, String variable, Object value) {
            this.column = column;
            this.operator = operator;
            this.variable = variable;
            this.value = value;
        }

        public String getColumn() {
            return column;
        }

        public String getOperator() {
            return operator;
        }

        public String getVariable() {
            return variable;
        }

        public Object getValue() {
            return value;
        }
    }

    class Between implements Condition {
        private final String column;
        private final String fromVariable;
        private final Object fromValue;
        private final String toVariable;
        private final Object toValue;

        public Between(String column, String fromVariable, Object fromValue, String
toVariable, Object toValue) {
            this.column = column;
            this.fromVariable = fromVariable;
            this.fromValue = fromValue;
            this.toVariable = toVariable;
            this.toValue = toValue;
        }
```

```
    public String getColumn() {
        return column;
    }

    public String getFromVariable() {
        return fromVariable;
    }

    public Object getFromValue() {
        return fromValue;
    }

    public String getToVariable() {
        return toVariable;
    }

    public Object getToValue() {
        return toValue;
    }
}

class Or extends Binary {
    public Or(Condition first, Condition second) {
        super(first, second, "OR");
    }
}

class And extends Binary {
    public And(Condition first, Condition second) {
        super(first, second, "AND");
    }
}

class Binary implements Condition {
    private final Condition first;
    private final Condition second;
    private final String operator;

    public Binary(Condition first, Condition second, String operator) {
        this.first = first;
        this.second = second;
        this.operator = operator;
    }

    public Condition getFirst() {
        return first;
    }

    public Condition getSecond() {
        return second;
    }

    public String getOperator() {
```

```
            return operator;
        }
    }

    class Permission implements Condition {
        private final Class<?> ownerClass;
        private final long ownerId;
        private final Class<?> propertyClass;
        private final long propertyId;
        private final boolean excludeGroups;

        private Permission(
                Class<?> ownerClass, long ownerId, Class<?> propertyClass, long propertyId,
boolean excludeGroups) {
            this.ownerClass = ownerClass;
            this.ownerId = ownerId;
            this.propertyClass = propertyClass;
            this.propertyId = propertyId;
            this.excludeGroups = excludeGroups;
        }

        public Permission(Class<?> ownerClass, long ownerId, Class<?> propertyClass) {
            this(ownerClass, ownerId, propertyClass, 0, false);
        }

        public Permission(Class<?> ownerClass, Class<?> propertyClass, long propertyId) {
            this(ownerClass, 0, propertyClass, propertyId, false);
        }

        public Permission excludeGroups() {
            return new Permission(this.ownerClass, this.ownerId, this.propertyClass,
this.propertyId, true);
        }

        public Class<?> getOwnerClass() {
            return ownerClass;
        }

        public long getOwnerId() {
            return ownerId;
        }

        public Class<?> getPropertyClass() {
            return propertyClass;
        }

        public long getPropertyId() {
            return propertyId;
        }

        public boolean getIncludeGroups() {
            boolean ownerGroupModel =
GroupedModel.class.isAssignableFrom(ownerClass);
            boolean propertyGroupModel =
```

```
GroupedModel.class.isAssignableFrom(propertyClass);
            return (ownerGroupModel || propertyGroupModel) && !excludeGroups;
        }
    }

    class LatestPositions implements Condition {
        private final long deviceId;

        public LatestPositions(long deviceId) {
            this.deviceId = deviceId;
        }

        public LatestPositions() {
            this(0);
        }

        public long getDeviceId() {
            return deviceId;
        }
    }

}
```

org.traccar.model.GroupedModel:

GroupedModel es probablemente una interfaz o clase en Traccar que define un modelo que puede estar asociado con un grupo. Se utiliza en la clase Permission para determinar si el modelo pertenece a un grupo y si los grupos deben ser considerados o excluidos en las condiciones de las consultas.

import org.traccar.storage.query.Request;

```
package org.traccar.storage.query;

public class Request {

    private final Columns columns;
    private final Condition condition;
    private final Order order;

    public Request(Columns columns) {
        this(columns, null, null);
    }

    public Request(Condition condition) {
        this(null, condition, null);
    }

    public Request(Columns columns, Condition condition) {
```

```
        this(columns, condition, null);
    }

    public Request(Columns columns, Order order) {
        this(columns, null, order);
    }

    public Request(Columns columns, Condition condition, Order order) {
        this.columns = columns;
        this.condition = condition;
        this.order = order;
    }

    public Columns getColumns() {
        return columns;
    }

    public Condition getCondition() {
        return condition;
    }

    public Order getOrder() {
        return order;
    }

}
```

Propósito:
Esta clase representa una consulta a la base de datos, combinando columnas, condiciones y órdenes de clasificación en una única solicitud.
Permite construir consultas de manera flexible, especificando solo las partes que son necesarias para una operación particular.

El paquete org.traccar.storage.query se utiliza para encapsular y estructurar las consultas a la base de datos en Traccar. Aquí te explico brevemente cómo funciona:

import org.traccar.model.Group;
Propósito: Esta es una clase de modelo en Traccar que representa un "grupo" dentro del sistema. Un grupo puede ser una colección de dispositivos o entidades relacionadas.
Función en el código: Group es el tipo de objeto que se manipula en GroupResource. Este recurso API permite realizar operaciones sobre los grupos en Traccar, como agregar un nuevo grupo o modificar uno existente.

```
/*
```

```
 * Copyright 2016 - 2018 Anton Tananaev (anton@traccar.org)
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */
package org.traccar.model;

import org.traccar.storage.StorageName;

@StorageName("tc_groups")
public class Group extends GroupedModel {

    private String name;

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

}
```

Es una clase Java llamada GroupResource, que es parte de un proyecto que utiliza el framework Traccar. Traccar es un sistema de rastreo GPS de código abierto. La clase GroupResource extiende de SimpleObjectResource<Group>, lo que sugiere que se trata de un recurso RESTful para manejar entidades del tipo Group.

import org.traccar.model.Group;: Importa la clase Group, que representa el tipo de objeto que esta API manejará.

Documentación Técnica: Componentes y Secuencia de Funcionalidades

Licencia y Derechos de Autor: El encabezado del archivo especifica los derechos de autor y las condiciones de uso del software, asegurando que los desarrolladores que utilicen o modifiquen este código entiendan las restricciones y permisos bajo la licencia Apache 2.0.

Paquete y Organización: La declaración del paquete org.traccar.api.resource ayuda a organizar el código dentro de la estructura del proyecto, haciendo que sea más fácil de encontrar y mantener.

Importaciones: Las clases y anotaciones importadas de otros paquetes y bibliotecas permiten que la clase GroupResource funcione correctamente como un recurso RESTful, proporcionando una API que puede manejar solicitudes HTTP para la gestión de objetos Group.

Anotaciones JAX-RS: @Path, @Produces y @Consumes configuran la API RESTful:

@Path: Define el endpoint para las operaciones HTTP.
@Produces: Especifica que la salida será en formato JSON.
@Consumes: Define que la entrada será en formato JSON.
Herencia de SimpleObjectResource: Al extender SimpleObjectResource, GroupResource hereda métodos para manejar operaciones estándar de una API RESTful (como GET, POST, PUT, DELETE) aplicadas a objetos del tipo Group.

Constructor y Super Clase: El constructor de GroupResource inicializa la clase base con el tipo de objeto Group, configurando SimpleObjectResource para manejar este tipo específico de entidad. Esto permite reutilizar lógica genérica para CRUD en objetos Group, reduciendo la duplicación de código y mejorando la mantenibilidad.

import org.traccar.storage.StorageName;
Propósito: Esta es una clase de modelo en Traccar que representa un "grupo" dentro del sistema. Un grupo puede ser una colección de dispositivos o entidades relacionadas, permitiendo organizar y gestionar estas entidades de manera estructurada dentro del sistema de rastreo GPS.

Función en el Código: La clase Group es el tipo de objeto que se manipula en la clase GroupResource. Este recurso API permite realizar operaciones sobre los grupos en Traccar, como agregar un nuevo grupo, modificar uno existente o eliminarlo. La clase Group define los atributos y métodos básicos necesarios para representar y manipular un grupo en el sistema.

```
/*
 * Copyright 2016 - 2018 Anton Tananaev (anton@traccar.org)
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */
package org.traccar.model;

import org.traccar.storage.StorageName;

@StorageName("tc_groups")
public class Group extends GroupedModel {

    private String name;

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

}
```

Documentación Técnica: Componentes y Secuencia de Funcionalidades

Inicialización de la Clase Group: La clase Group se inicializa y configura para representar un grupo dentro del sistema Traccar. Gracias a la herencia de GroupedModel, hereda funcionalidades básicas para la gestión de entidades agrupadas.

Asignación y Recuperación de Datos:

Asignación de Nombre: Se utiliza el método setName(String name) para establecer el nombre de un grupo. Esto es crucial al crear nuevos grupos o modificar los existentes.
Recuperación de Nombre: El método getName() permite acceder al nombre del grupo. Esto es útil en varias operaciones, como listar grupos o mostrar detalles específicos del grupo.
Persistencia en la Base de Datos:

La anotación @StorageName("tc_groups") conecta la clase Group con la tabla tc_groups en la base de datos, lo que permite a Traccar almacenar y recuperar información de grupos de manera eficiente.

import org.traccar.model.Group

La librería import org.traccar.model.Group se importa para poder utilizar la clase Group que se encuentra en el paquete org.traccar.model. Esta clase Group probablemente se utiliza para representar un grupo de objetos o dispositivos dentro del sistema Traccar, que es una plataforma de rastreo GPS.

Componentes y Secuencias

Licencia: Define los términos bajo los cuales se puede usar el archivo.

Paquete: Organiza la clase dentro del paquete org.traccar.model.

Importaciones: Importa dependencias necesarias para la clase.

Anotación @StorageName: Especifica la tabla de almacenamiento para los datos de la clase.

Clase Group: Define la estructura y comportamiento de un grupo.

Atributo name: Almacena el nombre del grupo.

Método getName: Proporciona acceso al nombre del grupo.

Método setName: Permite modificar el nombre del grupo.

```
/**
 * Clase que representa un grupo en el sistema Traccar.
 * Los datos de esta clase se almacenan en la tabla "tc_groups".
 */
@StorageName("tc_groups")
public class Group extends GroupedModel {

    /**
     * Nombre del grupo.
     */
    private String name;

    /**
     * Obtiene el nombre del grupo.
     *
     * @return el nombre del grupo.
     */
    public String getName() {
        return name;
    }
```

```
    /**
     * Establece el nombre del grupo.
     *
     * @param name el nombre del grupo.
     */
    public void setName(String name) {
        this.name = name;
    }
}
```

import org.traccar.model.Group

La clase StorageName es una anotación personalizada en el proyecto Traccar. Su principal propósito es servir como un marcador para indicar el nombre de la tabla en la base de datos donde se almacenará una clase de modelo específica. Esta anotación facilita el mapeo de clases de modelo a tablas de base de datos dentro del sistema, asegurando que los datos sean persistidos y recuperados correctamente.

Función en el Código

StorageName es utilizada para vincular clases de modelo con sus correspondientes tablas de base de datos en un sistema de almacenamiento persistente. Al aplicar esta anotación a una clase, el desarrollador define explícitamente el nombre de la tabla en la base de datos que almacena las instancias de esa clase. Esto es esencial en el contexto del Object-Relational Mapping (ORM), donde las clases de objetos se mapean a tablas de bases de datos relacionales.

```
/*
 * Copyright 2022 Anton Tananaev (anton@traccar.org)
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */
package org.traccar.storage;

import java.lang.annotation.ElementType;
import java.lang.annotation.Retention;
```

```
import java.lang.annotation.RetentionPolicy;
import java.lang.annotation.Target;

@Target(ElementType.TYPE)
@Retention(RetentionPolicy.RUNTIME)
public @interface StorageName {
    String value();
}
```

Documentación Técnica: Componentes y Secuencia de Funcionalidades

Definición de Anotación: StorageName se define como una anotación personalizada con un único elemento value de tipo String. Esto permite al desarrollador especificar un nombre de tabla al aplicar la anotación a una clase.

Especificación de Uso con @Target: La anotación @Target(ElementType.TYPE) se utiliza para restringir el uso de StorageName únicamente a nivel de clase. Esto asegura que la anotación no se pueda aplicar incorrectamente a métodos, campos u otros elementos.

Retención en Tiempo de Ejecución con @Retention: La anotación @Retention(RetentionPolicy.RUNTIME) garantiza que StorageName esté disponible en tiempo de ejecución, lo que es esencial para que los sistemas de persistencia que usan reflexión puedan leer la anotación y aplicar las reglas de mapeo correspondientes.

Aplicación en Clases de Modelo: En el contexto del proyecto Traccar, StorageName se aplica a las clases de modelo (por ejemplo, Group) para indicar a qué tabla de base de datos deben mapearse estas clases. Esta configuración permite que el sistema de persistencia sepa cómo almacenar y recuperar instancias de la clase.

La anotación StorageName proporciona un mecanismo para vincular clases de modelo a tablas de base de datos específicas dentro del sistema Traccar. Utilizando elementos de la biblioteca de anotaciones de Java (ElementType, Retention, RetentionPolicy), la anotación se define de manera que sea segura y eficaz para su uso en el mapeo ORM. Esta anotación es fundamental para la persistencia de datos, permitiendo a Traccar almacenar y gestionar datos de manera eficiente y mantenible.

La importación de import org.traccar.model.Group; en el archivo de código, como el que has proporcionado, se utiliza para hacer referencia a la clase Group del paquete org.traccar.model. Esta clase representa un grupo dentro del sistema Traccar y se utiliza para manejar y gestionar grupos de dispositivos o entidades.

Documentación de la Importación import org.traccar.model.Group;
Funcionalidad General de la Clase Group:

Propósito: La clase Group en Traccar representa una colección de dispositivos o entidades relacionadas. Es utilizada para agrupar y gestionar estas entidades dentro del sistema.
Función en el Código: Group es un tipo de modelo que se utiliza para definir las características de los grupos que pueden ser creados, modificados o eliminados en la plataforma Traccar.
Uso de la Importación en Diferentes Archivos
En los archivos proporcionados en tus imágenes, la clase Group se utiliza de la siguiente manera:

ExtendedObjectResource.java:

Uso de Group: La clase Group es utilizada para verificar los permisos relacionados con los grupos. En este contexto, se valida si un usuario tiene los permisos adecuados para acceder o modificar un grupo específico.
Ejemplo de Uso: En la sección de código proporcionada, permissionsService.checkPermission(Group.class, getUserId(), groupId); verifica si el usuario actual tiene permiso para acceder al grupo especificado por groupId.
CommandResource.java, GroupResource.java, PermissionsService.java, y otros:

Uso de Group: En estos archivos, Group se utiliza para realizar operaciones CRUD (Crear, Leer, Actualizar, Eliminar) sobre objetos Group, gestionar permisos de acceso, y realizar otras operaciones específicas relacionadas con los grupos en el sistema Traccar.
Ejemplo de Uso: Estos archivos pueden usar métodos como addGroup, updateGroup, deleteGroup, y otros, que operan directamente sobre instancias de la clase Group.
CacheManager.java y DatabaseStorage.java:

Uso de Group: La clase Group podría utilizarse en el contexto de manejo de caché o almacenamiento en base de datos para optimizar el acceso y la persistencia de datos relacionados con los grupos.

Ejemplo de Uso: En DatabaseStorage.java, la clase Group puede ser utilizada para mapear objetos de grupo a registros de la base de datos y viceversa, facilitando la persistencia y consulta de estos objetos.

En resumen, la importación import org.traccar.model.Group; es esencial en estos archivos para proporcionar acceso a la funcionalidad y estructura de la clase Group, permitiendo que las diferentes partes del código interactúen y manipulen los datos de los grupos dentro del sistema Traccar.