Presentación Laboratorio Entrega 3
Grupo 3

Taller de Base de Datos

Integrantes	Matías Cortés Diego Garrido Ian Rickmers Joakin Roa Luis Toro Leo Vergara
Profesores	Manuel Manríquez Luis Veas
Fecha	19-12-2022

Contenidos

- Introducción
- Descripción de estado de proyecto
- Diagrama de arquitectura
- Demostración de funcionalidades
- Conclusiones

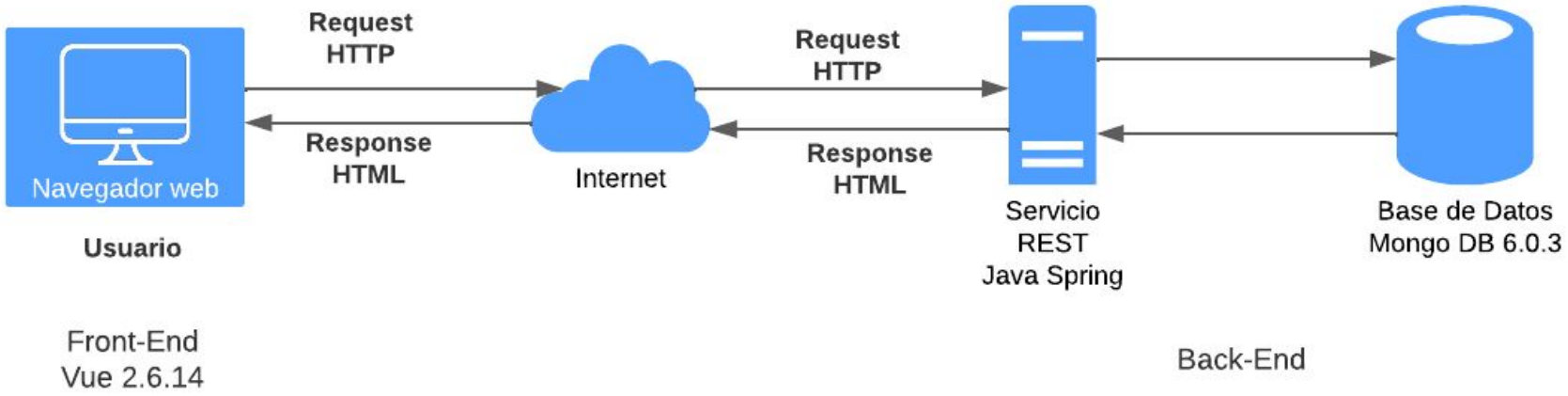
Introducción

- Dificultad para coordinar voluntarios para acudir a una emergencia
- MongoDB: sistema de base de datos nosql.

Descripción de estado del proyecto

Funcionalidad	Porcentaje de avance
F1: Modelar tabla Tareas como una colección de MongoDB.	100%
F2: Modelar tabla Voluntarios como una colección georreferenciada de MongoDB.	100%
F3: Realizar consulta en un servicio REST para obtener todas las tareas activas de la emergencia utilizando aggregate, lookup y unwind.	100%
F4: Mostrar vista de todas las emergencias en un mapa, donde se pueden ver los voluntarios que son afectados por una emergencia. Especificar un radio cuyo centro es el punto (x,y) de la emergencia.	100%

Diagrama de arquitectura



Modelar tabla Tareas como una colección de MongoDB

```
{
    "_id": 0,
    "nombre": "Sacar escombros",
    "descrip": "Sacar escrombros de la zona cero del temblor",
    "cant_vol_requeridos": 20,
    "cant_vol_inscritos": 14,
    "id_emergencia": {
        "_id": 4,
        "nombre": "Evacuación de personas (t)",
        "descrip": "Evacuación de personas ante un tsunami",
        "finicio": "1995-09-14",
        "ffin": "2002-02-25",
        "_id_institucion": {
            "_id": 4,
            "nombre": "27F",
            "descrip": "Institución Humanitaria"
        },
        "geom": {
            "type": "Point",
            "crs": {
                "type": "name",
                "properties": {
                    "name": "EPSG: 4326"
                }
            },
            "coordinates": [
                -70.691528,
                -33.185835
            ]
        },
        "estado": 1
    },
    "finicio": "2006-03-24",
    "ffin": "2011-05-25",
    "id_estado": {
        "_id": 0,
        "descrip": "Requerida"
```

Modelar tabla Tareas como una colección de MongoDB

```java
package com.app.voluntariosbe.models;

import java.util.Date;
import org.springframework.data.annotation.Id;
import org.springframework.data.mongodb.core.mapping.Document;

@Document(collection = "tarea")
public class TaskMongo {
    @Id
    private Integer id;
    private String nombre;
    private String descrip;
    private Integer cant_vol_requeridos;
    private Integer cant_vol_inscritos;
    private Integer id_emergencia;
    private Date finicio;
    private Date ffin;
    private Integer id_estado;

    public TaskMongo(Integer id, String nombre, String descrip, Integer cant_vol_requeridos, Integer cant_vol_inscritos, Integer id_emergencia, Date finicio, Date ffin, Integer id_estado) {
        this.id = id;
        this.nombre = nombre;
        this.descrip = descrip;
        this.cant_vol_requeridos = cant_vol_requeridos;
        this.cant_vol_inscritos = cant_vol_inscritos;
        this.id_emergencia = id_emergencia;
        this.finicio = finicio;
        this.ffin = ffin;
        this.id_estado = id_estado;
    }
```

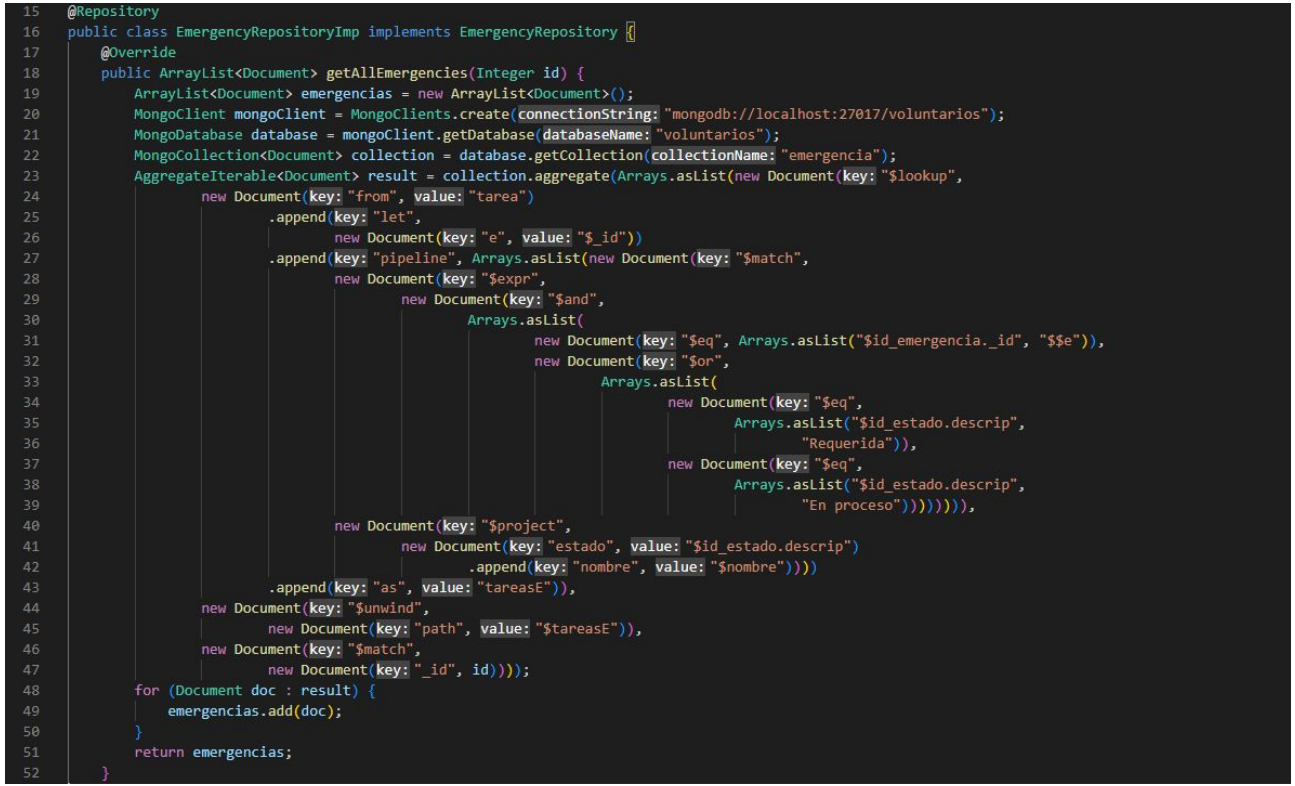
Modelar tabla Voluntarios como una colección georreferenciada de MongoDB


```
[
    {
        "_id": 0,
        "nombre": "Luis Roa",
        "fnacimiento": "1975-05-01",
        "geom": {
            "type": "Point",
            "coordinates": [
                -70.307007,
                -23.621878
            ]
        },
        "rut": "17.019.246-K"
    },
    {
        "_id": 1,
        "nombre": "Leo Vergara",
        "fnacimiento": "1980-12-22",
        "geom": {
            "type": "Point",
            "coordinates": [
                -68.31958,
                -22.843021
            ]
        },
        "rut": "20.426.303-5\t"
    },
    {
```

Modelar tabla Voluntarios como una colección georreferenciada de MongoDB

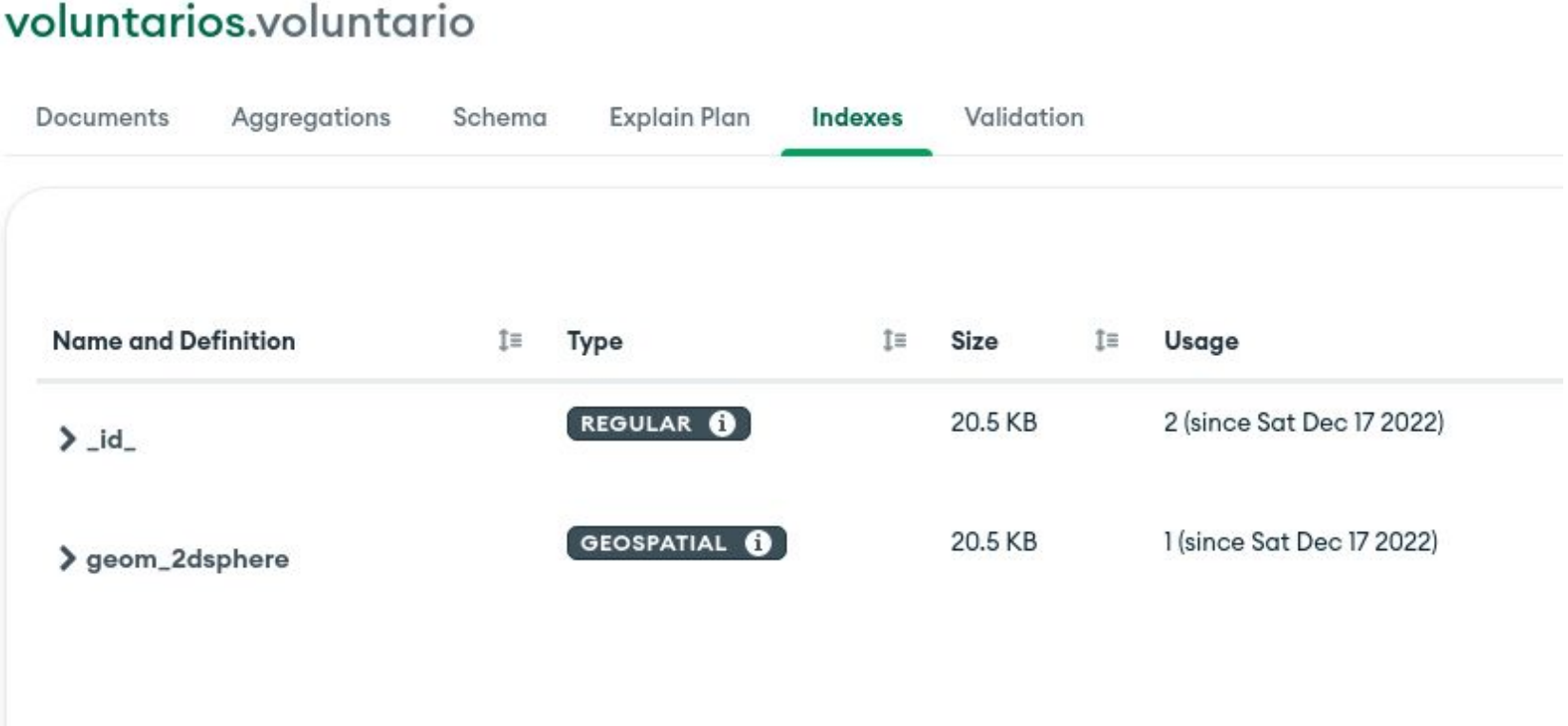
```java
package com.app.voluntariosbe.models;

import java.util.Date;
import org.springframework.data.mongodb.core.mapping.Document;
import org.springframework.data.annotation.Id;

@Document(collection = "voluntario")
public class VolunteerMongo {
    @Id
    private Integer id;
    private String nombre;
    private Date fnacimiento;
    private String rut;
    private double longitud;
    private double latitud;

    //Constructor
    public VolunteerMongo(Integer id, String nombre, Date fnacimiento, double longitud, double latitud) {
        this.id = id;
        this.nombre = nombre;
        this.fnacimiento = fnacimiento;
        this.longitud = longitud;
        this.latitud = latitud;
    }
```

Consulta REST que obtiene todas las tareas activas de una Emergencia



```java
@Repository
public class EmergencyRepositoryImp implements EmergencyRepository {
    @Override
    public ArrayList<Document> getAllEmergencies(Integer id) {
        ArrayList<Document> emergencias = new ArrayList<Document>();
        MongoClient mongoClient = MongoClients.create(connectionString: "mongodb://localhost:27017/voluntarios");
        MongoDatabase database = mongoClient.getDatabase(databaseName: "voluntarios");
        MongoCollection<Document> collection = database.getCollection(collectionName: "emergencia");
        AggregateIterable<Document> result = collection.aggregate(Arrays.asList(new Document(key: "$lookup",
                new Document(key: "from", value: "tarea")
                        .append(key: "let",
                                new Document(key: "e", value: "$_id"))
                        .append(key: "pipeline", Arrays.asList(new Document(key: "$match",
                                new Document(key: "$expr",
                                        new Document(key: "$and",
                                                Arrays.asList(
                                                        new Document(key: "$eq", Arrays.asList("$id_emergencia._id", "$$e")),
                                                        new Document(key: "$or",
                                                                Arrays.asList(
                                                                        new Document(key: "$eq",
                                                                                Arrays.asList("$id_estado.descrip",
                                                                                        "Requerida")),
                                                                        new Document(key: "$eq",
                                                                                Arrays.asList("$id_estado.descrip",
                                                                                        "En proceso")))))))),
                                new Document(key: "$project",
                                        new Document(key: "estado", value: "$id_estado.descrip")
                                                .append(key: "nombre", value: "$nombre"))))
                        .append(key: "as", value: "tareasE")),
                new Document(key: "$unwind",
                        new Document(key: "path", value: "$tareasE")),
                new Document(key: "$match",
                        new Document(key: "_id", id))));
        for (Document doc : result) {
            emergencias.add(doc);
        }
        return emergencias;
    }
```

Configuracion de voluntarios



Voluntarios dentro de una área de una emergencia

```java
public ArrayList<Document> getEmergencyLocations() {
    ArrayList<Document> emergencias = new ArrayList<Document>();
    MongoClient mongoClient = MongoClients.create("mongodb://localhost:27017/voluntarios");
    MongoDatabase database = mongoClient.getDatabase("voluntarios");
    MongoCollection<Document> collection = database.getCollection("emergencia");
    Bson filter = new Document();
    FindIterable<Document> result = collection.find(filter);
    for (Document doc : result) {
        emergencias.add(doc);
    }
    return emergencias;
}
```

Voluntarios dentro de una área de una emergencia

```java
@Repository
public class VolunteerRepositoryImp implements VolunteerRepository {

    public ArrayList<Document> getVolunteers(Double longitud, Double latitud) {
        Bson filter = near("geom", new Point(new Position(longitud, latitud)), 500000d, 0d);

        MongoClient mongoClient = MongoClients.create("mongodb://localhost:27017/voluntarios");
        MongoDatabase database = mongoClient.getDatabase("voluntarios");
        MongoCollection<Document> collection = database.getCollection("voluntario");
        FindIterable<Document> result = collection.find(filter);
        ArrayList<Document> voluntarios = new ArrayList<Document>();
        for (Document doc : result) {
            voluntarios.add(doc);
        }
        return voluntarios;
    }
}
```

Voluntarios dentro de una área de una emergencia

```
methods: {
    mostrarPuntos() {
        if (this.id_emergencia == -1) {
            this.message = "Debe seleccionar una emergencia";
        } else {
            this.coordenadas = this.emergencias[this.id_emergencia].geom.coordinates;
            this.clearMarkers()
            //Se agregan los puntos mediante llamada al servicio
            this.getPoints(this.mymap);
        }
    },
    async getEmergencies() {
        const url = "http://localhost:8090/emergencies/locations";
        await axios.get(url)
            .then((response) => {
                this.emergencias = response.data.sort((a, b) => a._id - b._id)
            })
            .catch((error) => {
                console.log(error)
            })
    },
    clearMarkers: function () { //eliminar marcadores

        this.points.forEach(p => {
            this.mymap.removeLayer(p);
        })
        this.points = [];
    },
```

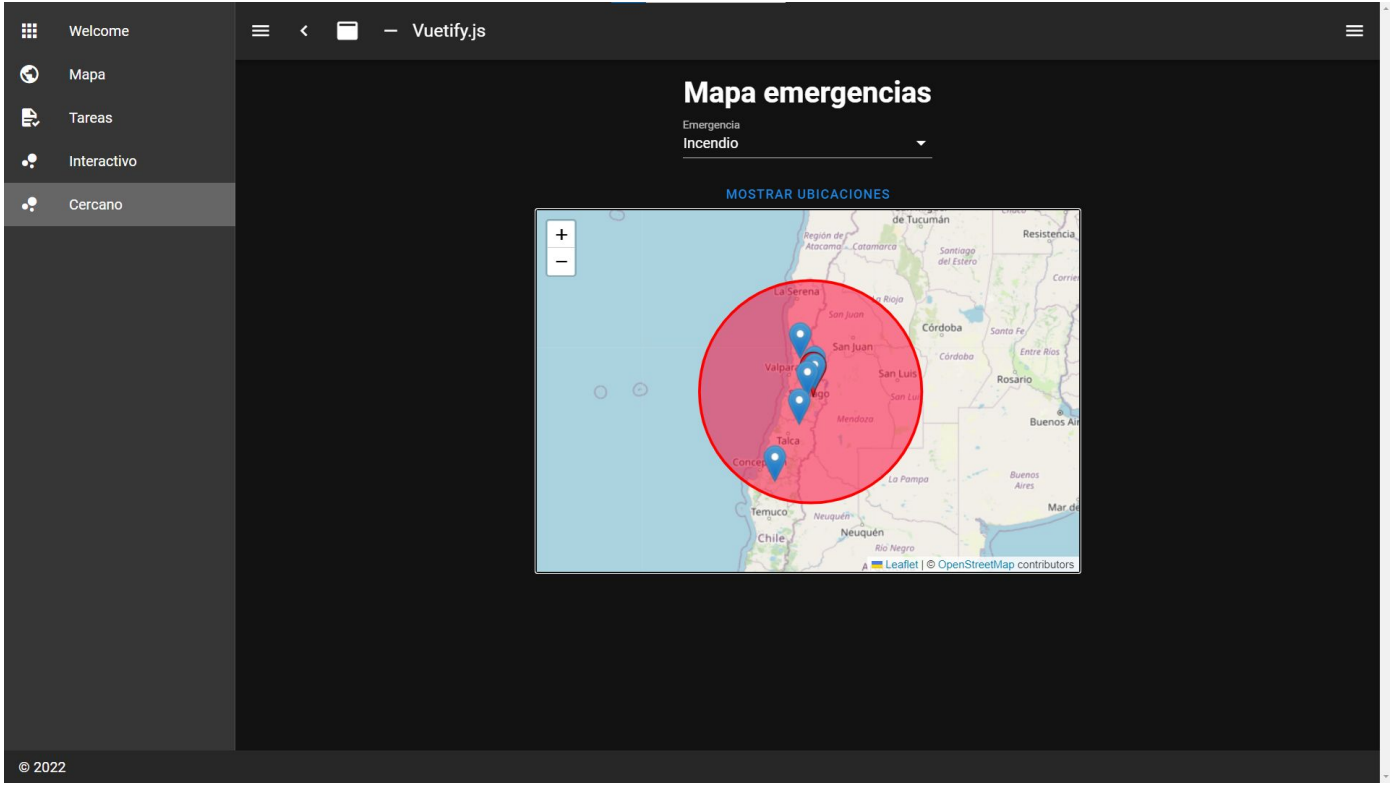
Voluntarios dentro de una área de una emergencia

```
async getPoints(map) {
    try {
        //se llama el servicio
        let lon = this.coordenadas[0]
        let lat = this.coordenadas[1]
        const url = 'http://localhost:8090/volunteers/near/' + String(lon) + '/' + String(lat);
        let response = await axios.get(url);
        let dataPoints = response.data;

        // Punto de la emergencia
        let pE = [lat, lon]
        let infoE = this.emergencias[this.id_emergencia].nombre;
        let markerE = L.marker(pE, { icon: myIcon2 }).bindPopup(infoE);
        this.points.push(markerE);

        // Radio de la emergencia
        var circle = L.circle([lat, lon], {
            color: 'red',
            fillColor: '#f03',
            fillOpacity: 0.5,
            radius: 500000
        })
        this.points.push(circle)
```

Voluntarios dentro de una área de una emergencia

```
    //Se itera por los puntos
    dataPoints.forEach(point => {
        let info = point.nombre + ", " + point.fnacimiento + ", " + point.rut + ", " + point.geom.coordinates[0] + ", " + point.geom.coordinates[1]
        //Se crea un marcador por cada punto
        let p = [point.geom.coordinates[1], point.geom.coordinates[0]]
        let marker = L.marker(p, { icon: myIcon }) //se define el ícono del marcador
            .bindPopup(info); //Se agrega un popup con el nombre

        //Se agrega a la lista
        this.points.push(marker);
    });

    //Los puntos de la lista se agregan al mapa
    this.points.forEach(p => {
        p.addTo(map)
    })
} catch (error) {
    console.log('error', error);
}
```

Voluntarios dentro de una área de una emergencia



Conclusiones

- Funcionalidades completadas al 100%
- Motor de base de datos MongoDB
- Base de datos no relacional