07

Como visto em aula

Nesta aula criamos a funcionalidade de buscar alunos próximos integrando Java, MongoDB e a API do Google Maps.

Para uso de estudo, essa API disponibiliza uma quantidade de requisições no plano gratuito. Faça seu cadastro e obtenha sua chave aqui:

https://developers.google.com/maps/?hl=pt-br (https://developers.google.com/maps/?hl=pt-br)

Precisamos da biblioteca Java do Google maps na nossa aplicação:

```
<dependency>
    <groupId>com.google.maps</groupId>
    <artifactId>google-maps-services</artifactId>
    <version>0.1.17</version>
</dependency>
```

Agora com a API já no nosso sistema, podemos fazer uma pesquisa de geolocalização baseada no endereço de um aluno para obter a latitude e longitude:

```
public List<Double> obterLatLongPor(Contato contato) throws Exception {
        GeoApiContext context = new GeoApiContext().setApiKey("SUA-CHAVE-AQUI");

        GeocodingApiRequest request = GeocodingApi.newRequest(context).address(contato.getEnder

        GeocodingResult[] results = request.await();
        GeocodingResult result = results[0];
        Geometry geometry = result.geometry;
        LatLng location = geometry.location;
        return Arrays.asList(location.lat, location.lng);

    }
```

Agora com os dados você já consegue obter os alunos mais próximos usando o banco de dados.

```
public List<Aluno> obterAlunosProximos(Contato contato) {
        criarConexao();
        MongoCollection<Aluno> collectionAlunos = obterCollection();
        collectionAlunos.createIndex(Indexes.geo2dsphere("contato"));
        Point pontoReferencia = new Point(new Position(contato.getCoordinates().get(0), contato
        MongoCursor<Aluno> alunosMaisProximos = collectionAlunos.find(Filters.nearSphere("conta
        List<Aluno> alunos = popularAlunos(alunosMaisProximos);
        fecharConexao();
        return alunos;
    }
```

Agora é só integrar esse resultado com a API do Google Maps na camada de visão.