HSL Interactive Database for Trees and Stones:

1. The Map: The map used in this project is one of the default map styles from MapBox. The data is loaded into map from the template and is fetched from a function within the view. Within the template, the GeoJSON data stored in the database is parsed and stored within a FeatureCollection as separate features. GeometryCollections are supported. FeatureCollections are split into individual Features with the same properties. For stones, the start and end date are processed and loaded as map information in the form of integers, with BCE/BC dates being represented by negative numbers. The data is loaded as four separate layers: two for stones, with one highlighted, and two for trees, with one highlighted. Zoom, pan, tilt, etc. are supported in the map by default.

    - A number of interactive elements are displayed over the map with HTML. These map overlays are defined and styled in the template, and their functions are defined through javascript.
    - Loading data: The functions for loading information into the map are split between the template "map.html" and the static file map-load.js. Some of mapbox's built in functions can only be used in the same file with the map container (the html file), but most functionality was moved to map-load.js.

2. Models: This project uses a number of models to store information. The most important models are the Trees and Stones models, which hold information about the database's trees and stones, respectively. The TreeImages and StoneImages classes hold images associated with trees/stones, using a ForeignKey field in order to connect back to their respective material types.

3. Queries: Results are found by using built in Mapbox GL JS functions combined with Turf functions used to aid in creating geographic shapes. The query returns all results either beneath a point or within a bounding box depending on the user's selection for search radius. The query is unable to return results outside of the currently loaded tilesets, which means that certain items may not be found based on the current zoom level and position of the map. The original intention was to use circular raycasting, but this is not currently supported by MapBox, so the bbox is more convenient with an acceptable margin of error (about 1.4x the search radius at most, with the area in which the result can be found being shown to the user).

    - File location: Most of this functionality is defined in the static file "map-scripts.js".
    - Item information: When a user clicks on a result, an event fires that requests the item's database entry from the server. The item's information is separated in a list of dictionaries that contain the

attribute name and value. This is returned to the template in JSON format and displayed in the "results-box" div.

- Geographic area: When an item is selected, the appropriate "-highlighted" (either stones-highlighted or trees-highlighted) layer is filtered by its item name, and the "-highlighted" layer is set to visible. When any of the map overlays are closed, the "-highlighted" layer is set to not visible.
- Images: Along with the item information, the attributes returned to the template also include a list of image URLs that are associated with the item in question. These images are displayed in the "picture-box" div and can be toggled through via tags. The HTML is generated when a user initially clicks on a search result.

4. Filtering: There are a number of options the user can choose between in order to limit their results. The main code for these features is in the static file map-scripts.js, though some preprocessing happens in the view when the data is given to the template.

    - Filtering by type: The user can select whether they want to look for stones and/or trees, as specified by a checkbox in a map overlay. When one of the items is selected, when the user clicks. queryRenderedFeatures() is called for the layer specified (either "trees" or "stones"). If one (or both) of the types is not selected, the layer is not queried.
    - Filtering by age: This is only relevant for stones, as only stones have a start and end date specified. The user can specify start and end dates through text fields for the numerical year and radio buttons for a choosing between BCE/CE. If the dates are invalid (a year or era is specified but not both or the start date is after the end date), the user is shown an error message and the date filters are not applied. If the dates are valid, then before the query, a filter (according to Mapbox's style specifications), is created and applied that filters stones for a start date greater than the user's specified start bound (translated to negative/positive rather than BCE/CE) and/or an end date less than the user's specified end bound.