COMP23412

Week 7
Using external APIs: the Mapbox API

Markel Vigo, Robert Haines, Mustafa Mustafa

Motivation

- Use of external/third party APIs is common
- From requirements to specific development needs
- Understanding what the API can do for us is key
- Sometimes it feels overwhelming
- In Eventlite it is about
  - Finding the right functionalities
  - Fitting the API into our MVC architecture

Where to start?

- Looking for the right API is a search problem
  - What am I looking for?
  - Is there any documentation provided?
  - Is it of a good quality? (ie. examples and updated)
- Which are your needs?
  - Is there a community that will support?
  - Are there any quotas?
- Scenarios:
  - Best case: a lot of good documentation
  - Worst case: no documentation

From requirements to selecting the API

From requirements to specific development needs

- Today's requeriment(s): "*display the location of events on a map, given the street name of the venue*"

1. **Get geographic location from an address.**

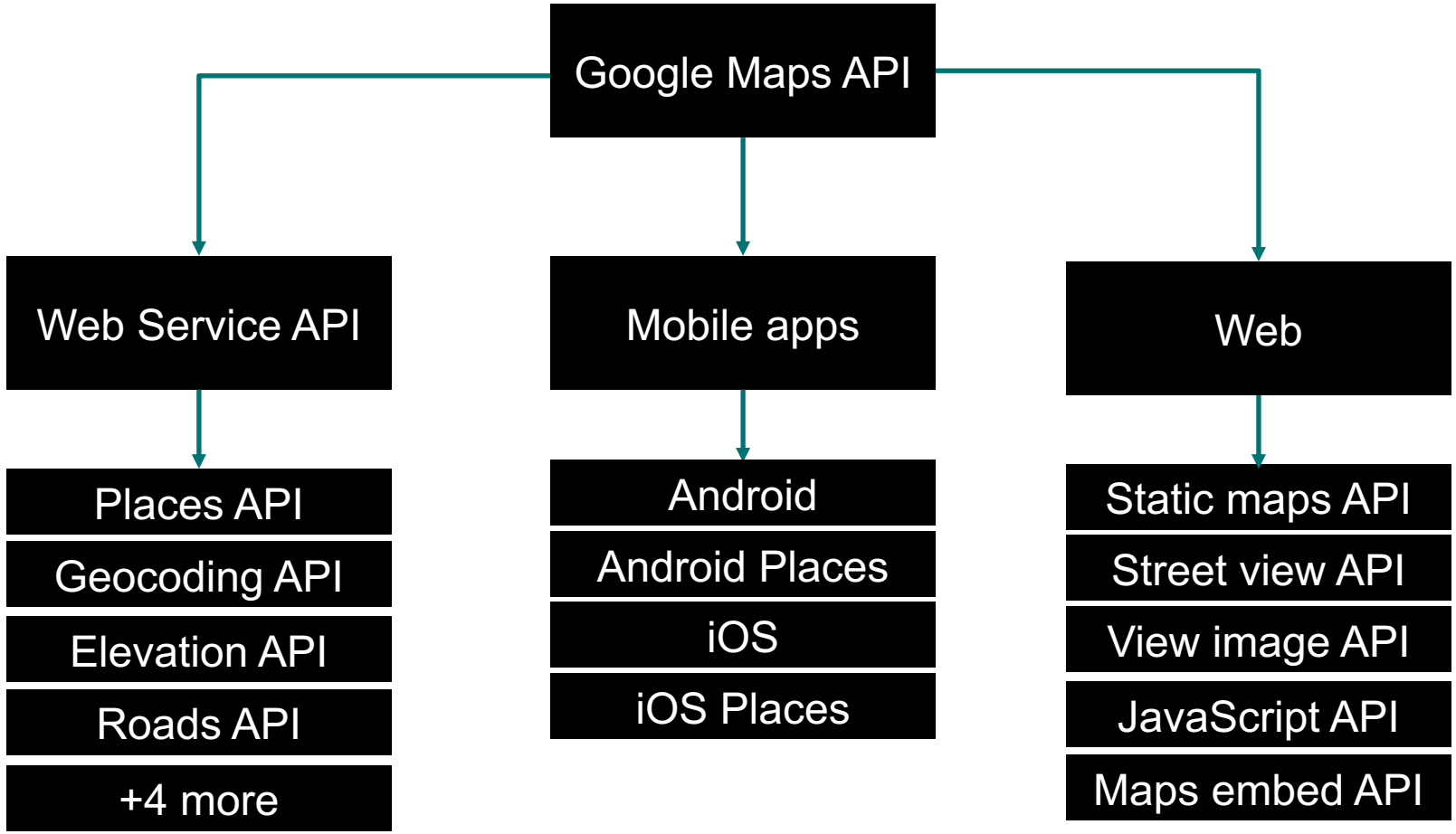
   Getting geographical coordinates from the address is called geocoding

2. **Display on a map.**

   Plots markers on locations if longitude and latitude are given

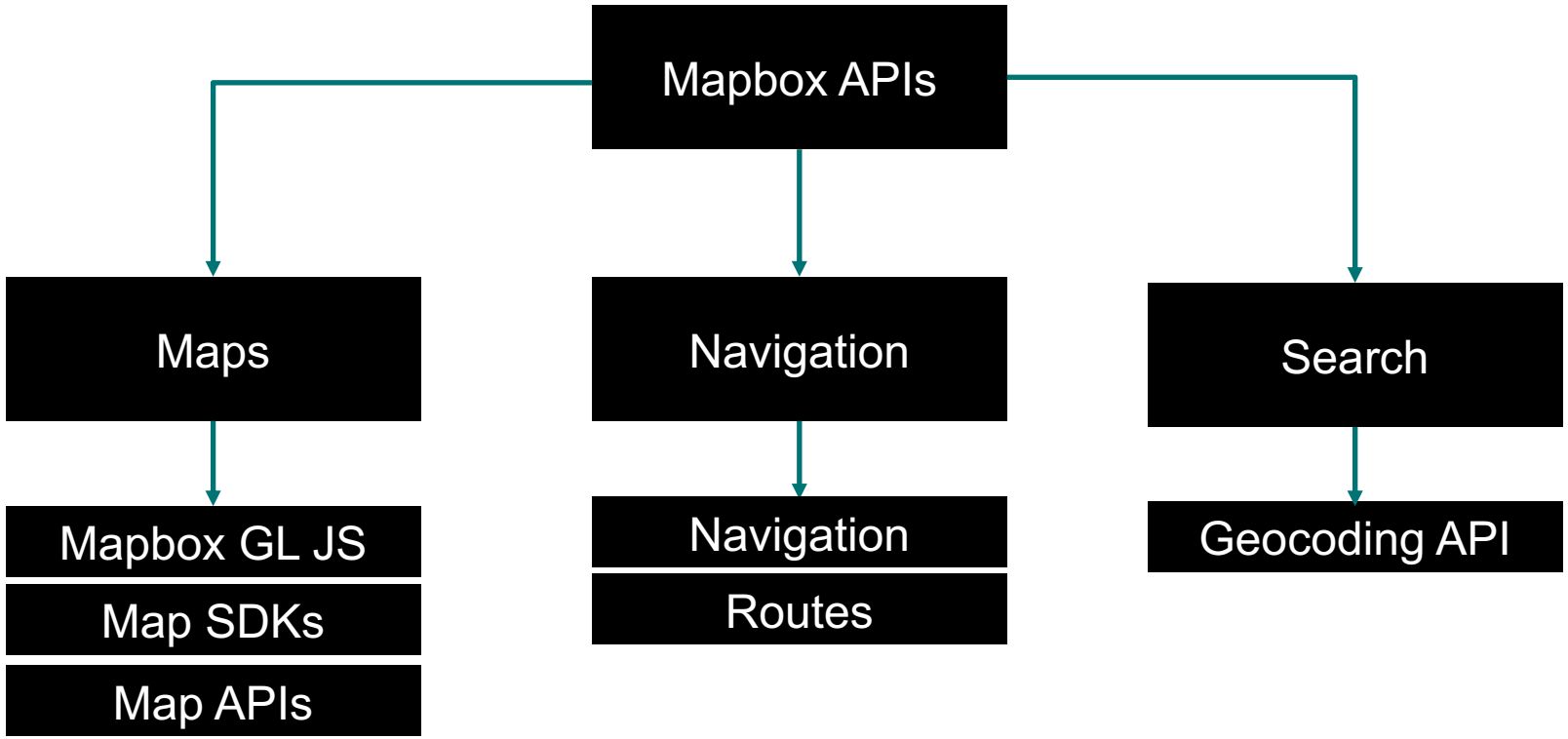
Google Maps API

- Looking for the right functionalities is a search problem



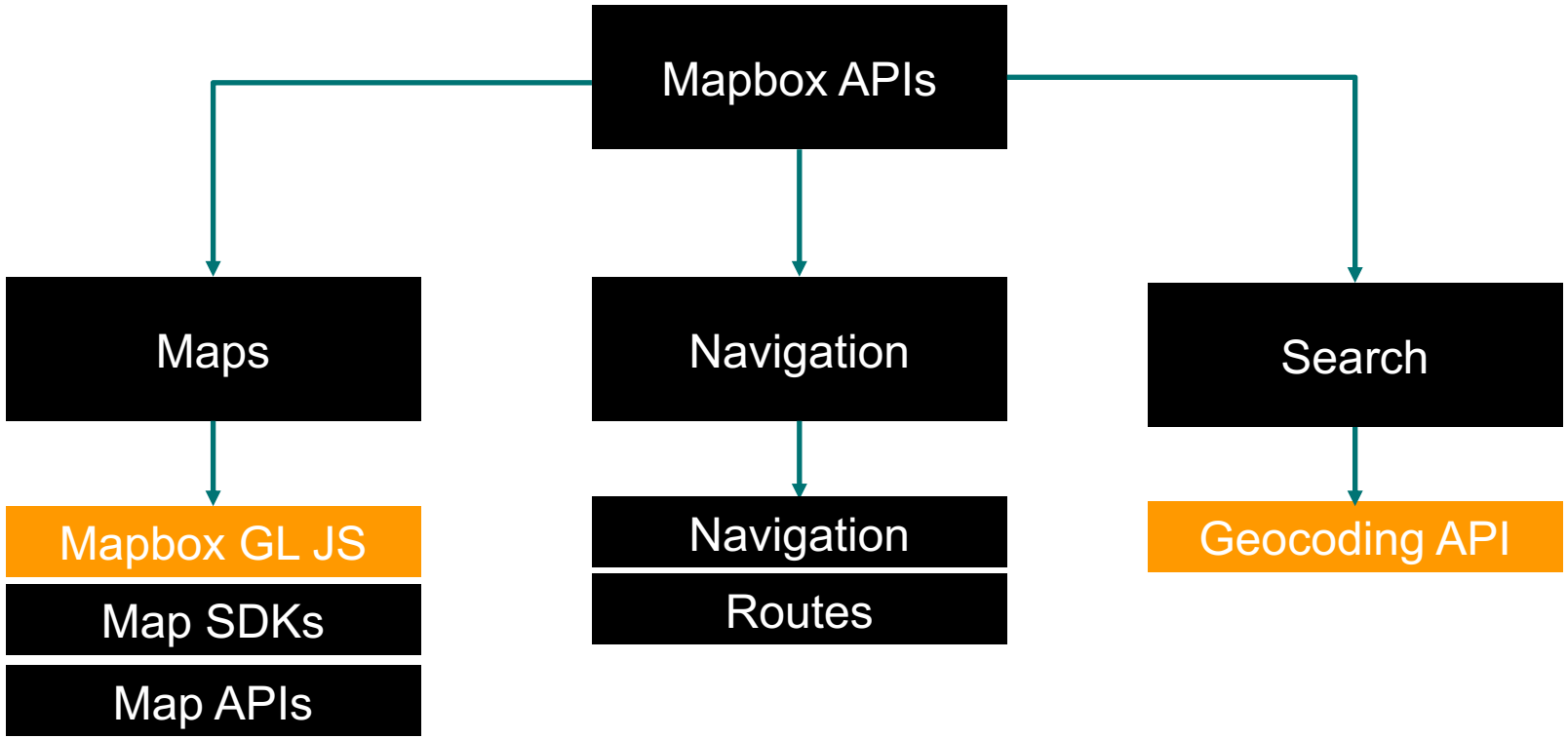
Mapbox API

- Looking for the right functionalities is a search problem

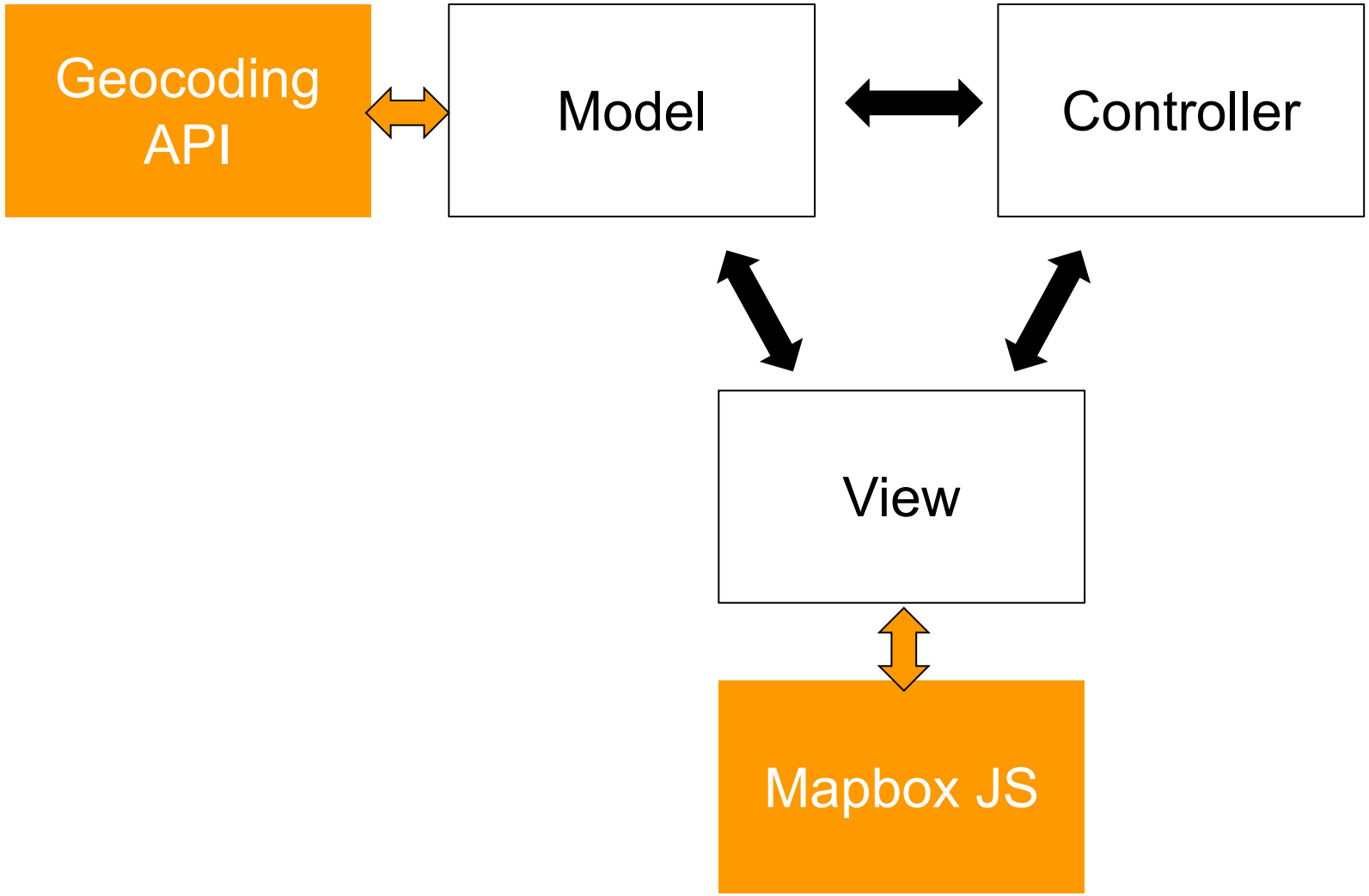


Mapbox API

- Looking for the right functionalities is a search problem



The Mapbox API in Spring boot



Programming by example

- The art of reusing existing code on your code
- Very extended thanks to online communities of programmers
- It's all about finding analogies and see the code as a template
- Two antagonistic strategies
  - Copy and paste and try to make it work until it works.
  - Try to understand the *why*s and *how*s

Programming by example

Geocoding example

```
MapboxGeocoding mapboxGeocoding = MapboxGeocoding.builder()
  .accessToken(                    )
  .query("                         ")
  .build();

mapboxGeocoding.enqueueCall(new Callback<GeocodingResponse>() {
  @Override
  public void onResponse(Call<GeocodingResponse> call, Response<GeocodingResponse>
  response) {

    List<CarmenFeature> results = response.body().features();

    if (results.size() > 0) {

      // Log the first results Point.
      Point firstResultPoint = results.get(0).center();
      Log.d(TAG, "onResponse: " + firstResultPoint.toString());

    } else {

      // No result for your request were found.
      Log.d(TAG, "onResponse: No result found");

    }
  }
```

Programming by example

Plotting events on maps

```
<script>
L.mapbox.accessToken = '[redacted]';
var map = L.mapbox.map('map')
    .setView([redacted], [redacted], [redacted])
    .addLayer(L.mapbox.styleLayer('mapbox://styles/mapbox/streets-v11'));

// L.marker is a low-level marker constructor in Leaflet.
L.marker([redacted], [redacted], {
    icon: L.mapbox.marker.icon({
        'marker-size': 'large',
        'marker-symbol': 'bus',
        'marker-color': '#fa0'
    })
}).addTo(map);
</script>
```

⚠ Deprecated example, but the main principle applies

Lab this week

1. Continue with (previous) requirements
2. On the *Model*:
   - Add `longitude` and `latitude` to the Venue class
   - **Important**: use setters and getters appropriately to prevent problems in the view

     `attribute` → `getAttribute()`, `setAttribute(..)`
   - Use the Geocoding API to retrieve the geographical coordinates of venues given their address

Lab this week

3. On the *View*: plot events on their venues
   - Single event using on a single event view
   - A number of events on the multiple events view

- Talk between yourselves to distribute the work and get the API key
- Write tests
- Use the JavaScript console to debug the View
- More information on Blackboard