Building a Point of Interest Recommendation App with the Amadeus for Developers Travel APIs

Thursday, July 4, 2024 | By
Jura Gorohovsky (https://portal.draft.dev/writers/recqPggCk9zdjkQkm)

Tutorials (/blog/category/tutorials) **24 -minute read**

> When you're traveling the world, you often only have a few days to grasp the essence of the places you visit, get a local experience, and leave with a lasting impression. People increasingly turn to apps to help them select where they'll visit to make the most of their trips. If you're developing a travel app, implementing features that recommend travel highlights to your customers may seem challenging. However,
> Amadeus for Developers (https://developers.amadeus.com/) offers a
> Points of Interest API (https://developers.amadeus.com/self-service/category/destination-experiences/api-doc/points-of-interest)
> that lets you easily implement a feature for finding the best attractions in a given area, enriched with categories, tags, and scores sourced from millions of online reviews.

Amadeus for Developers is a travel data provider with over thirty-five years of experience (https://amadeus.com/en/about) that lets you build, test, and deliver applications quickly and painlessly, and it provides REST APIs and the convenience of test calls directly from API documentation pages or via a dedicated Postman collection (https://developers.amadeus.com/self-service/apis-docs/guides/developer-guides/developer-tools/postman/). Furthermore, developers can use a wide array of SDKs, including official offerings for Node, Java, and Python, along with community libraries for PHP, .NET, iOS, and Android, complemented by a wealth of code samples (https://developers.amadeus.com/self-service/apis-docs/guides/developer-guides/examples/code-example/) and demo applications (https://github.com/search?q=topic%3Ademo+org%3Aamadeus4dev&type=Repositories) to expedite development.

Amadeus for Developers is

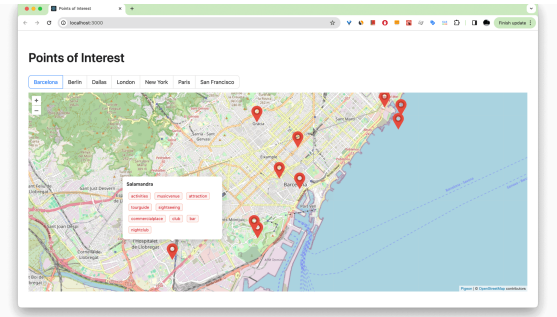
easy to set up (https://developers.amadeus.com/get-started/get-started-with-self-service-apis-335)
and also offers a generous free quota of 200,000 requests per month across forty APIs, which you'll also retain in the transition to production, with additional calls incurring charges.

Read on to learn how to use Amadeus for Developers to create a prototype React application that uses the Points of Interest API to show places worth visiting across a selection of cities.

Building a Point of Interest Recommendation App: Walkthrough

You'll create a simple React application that looks up points of interest in several cities. Normally, you would split an application like this into a backend for sourcing data and a frontend for displaying data, but for the sake of simplicity, let's go with a frontend-only approach.

Here's what the application will look like when you're finished:



Before you start, make sure you have Node installed on your system. If you don't already have a Node installation, please download and install the latest Node LTS release (https://nodejs.org/en).

Setting Up the React Application Boilerplate

You'll use the following libraries in this application:

- React (https://react.dev/) to build the UI and manage state
- Pigeon Maps (https://pigeon-maps.js.org/) to display locations on a map
- Ant Design (https://ant.design/) to use readily available web components for the map and controls
- Sass (https://sass-lang.com/) for better styling control

You'll first need to create a backbone for your React application and add these dependencies.

Create a new directory to host the application and navigate to it:

```
mkdir PointsOfInterestFinder

cd PointsOfInterestFinder
```

If you're going to put the application code under version control, create the following `.gitignore` file (if not, it's safe to skip this step):

```
src/credentials.json

# dependencies
/node_modules
/.pnp
.pnp.js

# production
/build

# misc
.DS_Store
.env.local
.env.development.local
.env.test.local
.env.production.local

npm-debug.log*
yarn-debug.log*
yarn-error.log*
```

To set the basic configuration and dependencies for the application, create a `package.json` file and insert the following code into it:

```
{
  "name": "points-of-interest",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "antd": "^5.13.3",
    "pigeon-maps": "^0.21.3",
    "react": "^18.2.0",
    "react-dom": "^18.2.0",
    "react-scripts": "5.0.1",
    "sass": "^1.70.0"
  },
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
    "eject": "react-scripts eject"
  },
  "browserslist": {
    "production": [
      ">0.2%",
      "not dead",
      "not op_mini all"
    ],
    "development": [
      "last 1 chrome version",
      "last 1 firefox version",
      "last 1 safari version"
    ]
  }
}
```

Now that you have a file that declares the necessary dependencies for the application, go ahead and install them by running the following:

```
npm install
```

You now have a file structure that includes your React application's entry point and the dependencies that it needs to do its job. However, the application has nothing to display. Let's add the parts that the app will actually display in the browser.

Creating and Placing the Application Component

To make React render any data, you need to add a static HTML page that will host your React application in the browser. To do this, create a new directory called `public`. Inside that directory, create an `index.html` file and insert the following markup:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initia
l-scale=1" />
    <meta name="description" content="Looking up city attract
ions using the Amadeus Points of Interest API"
    />
    <title>Points of Interest</title>
  </head>
  <body>
    <noscript>You need to enable JavaScript to run this ap
p.</noscript>
    <div id="root"></div>
  </body>
</html>
```

This is just a canvas for the React application to attach to in the browser. You now need to define exactly what you want to attach. Back at the root of the application, create another new directory, `src`. Inside, create an `index.js` file and add the following code:

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import PointsOfInterest from "./PointsOfInterest";
const root = ReactDOM.createRoot(document.getElementById('roo
t'));
root.render(<PointsOfInterest/>);
```

This file instructs React to attach to the `<div id="root">` element in the HTML file that you created and render a component called `PointsOfInterest`. This is going to be the component encapsulating most of your application, and now that you've referenced the component, it's time to actually create it. Inside the `src` directory, create a new file called `PointsOfInterest.js` and insert the following code:

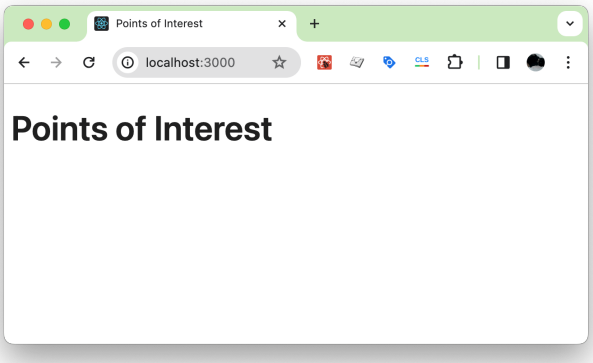
```
import React from "react";
import {Typography} from 'antd';
const {Title} = Typography;
function PointsOfInterest() {
    return (
        <div className="content">
            <div className="title">
                <Title>Points of Interest</Title>
            </div>
        </div>
    )
}
export default PointsOfInterest;
```

You're going to extend this React function component until your application is complete. Even though it only contains the application's title so far, the upside is that you can now actually launch it in the

browser. Return to the root directory and start the application:

```
npm start
```

This command should open the application in the browser at `http://localhost:3000/`. Here's what the application looks like at this point:



Adding a Map

If you're going to fetch points of interest from the Amadeus for Developers API, you'll need a map to plot them on. Your project already has a map library installed called Pigeon Maps (https://pigeon-maps.js.org/).

Back in `PointsOfInterest.js`, insert the following `import` statements:

```
import {Map, ZoomControl} from "pigeon-maps";
import './styles.scss';
```

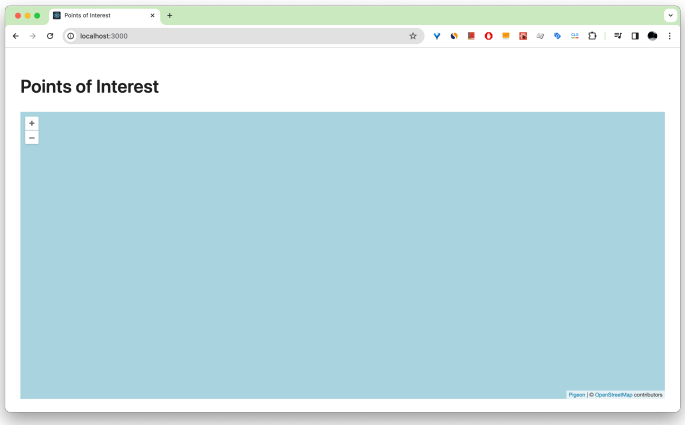
Inside the `<div className="content">` element, after the `title` block, add the following code:

```
<div className="map">
    <Map id="map"
         defaultZoom={13}>
        <ZoomControl/>
    </Map>
</div>
```

Back inside the `src` directory, create a new file called `styles.scss` and add the following styles. These cover all the styles you need right now and in all subsequent steps.

```
$space: 6px;
.content {
  display: flex;
  flex-direction: column;
  flex-wrap: wrap;
  align-items: start;
  padding: $space*4;
  .title {
    width: 100%;
    margin-bottom: $space*2;
  }
  .controls {
    display: flex;
    flex-direction: column;
    flex-wrap: wrap;
    align-items: start;
    padding-right: $space*4;
    width: 100%;
    height: auto;
    > * {
      margin-bottom: $space*2;
    }
  }
  .map {
    width: 100%;
    height: 600px;
  }
}
.ant-popover {
  max-width: 300px;
}
.ant-tag {
  margin: $space
}
```

As a result, you should be able to see a map canvas with prebuilt zoom in and zoom out controls:



The map currently points to a location somewhere in the Atlantic. It's unlikely that ocean waters have a lot of meaningful attractions, so you need to provide some coordinates of popular tourist destinations.

Adding Predefined Locations

In a production application, you would either ask the user to allow access to their current location or let them enter a location in an input box. You won't go beyond the Amadeus for Developers test environment in this tutorial, and the test environment only provides point of interest data for a

selection of cities (https://github.com/amadeus4dev/data-collection/blob/master/data/pois.md)
. This means you'll need to hard-code coordinates for some of these cities.

Inside the `src` directory, create a new subdirectory called `data`. Inside `data`, create a new file called `regions.js` and insert the following code:

```
const regions = [
    {
        name: "Barcelona",
        location: {
            latitude: 41.38,
            longitude: 2.168365,
            square: {
                north: 41.42,
                west: 2.11,
                south: 41.347463,
                east: 2.228208
            }
        }
    },
    {
        name: "Berlin",
        location: {
            latitude: 52.519171,
            longitude: 13.406091,
            square: {
                north: 52.541755,
                west: 13.354201,
                south: 52.490569,
                east: 13.457198
            }
        }
    },
    {
        name: "Dallas",
        location: {
            latitude: 32.779167,
            longitude: -96.808891,
            square: {
                north: 32.806993,
                west: -96.836857,
                south: 32.740310,
                east: -96.737293
            }
        }
    },
    {
        name: "London",
        location: {
            latitude: 51.509865,
            longitude: -0.118092,
            square: {
                north: 51.520180,
                west: -0.169882,
                south: 51.484703,
                east: -0.061048
            }
        }
    },
    {
        name: "New York",
        location: {
            latitude: 40.730610,
            longitude: -73.99,
            square: {
                north: 40.792027,
                west: -74.058204,
                south: 40.697607,
                east: -73.942847
            }
        }
    },
    {
```

```
        name: "Paris",
        location: {
            latitude: 48.864716,
            longitude: 2.349014,
            square: {
                north: 48.91,
                west: 2.25,
                south: 48.80,
                east: 2.46
            }
        }
    },
    {
        name: "San Francisco",
        location: {
            latitude: 37.773972,
            longitude: -122.431297,
            square: {
                north: 37.810980,
                west: -122.483716,
                south: 37.732007,
                east: -122.370076
            }
        }
    },
]
export default regions;
```

This file defines several popular tourist destinations, along with their geocodes, that you'll use to center the locations on the map and request points of interest from the Amadeus for Developers API. Let's now use the file to show one of the predefined locations on the map.

Open `PointsOfInterest.js` and add an `import` statement for the new file:

```
import predefinedLocations from "./data/regions"
```

Modify the existing `import` statement for `react` to include the `useState` hook:

```
import React, {useState} from "react";
```

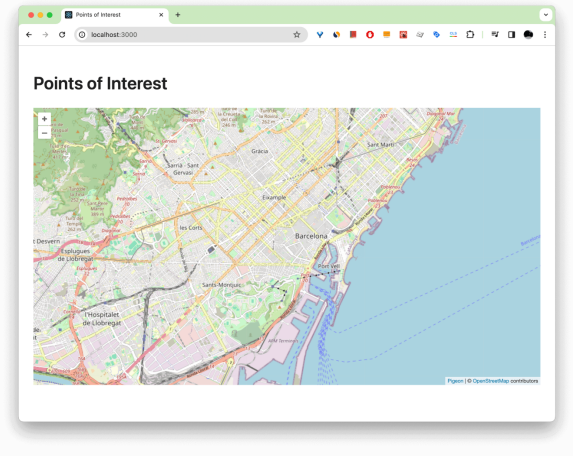
At the very start of the `PointsOfInterest()` function component, insert the following code:

```
const [city, setCity] = useState(predefinedLocations.find(x =
> x.name === "Barcelona"));
```

Scroll down to the `return` statement, locate the `Map` element, and add two new attributes to it to go along with the existing `defaultZoom` attribute:

```
key={city.name}
defaultCenter={[city.location.latitude, city.location.longitu
de]}
```

The `PointsOfInterest` component now has a state object `city`, and every time the application starts, it looks up geocode data for the city of Barcelona. Barcelona's latitude and longitude are then used to center the map on the city. Here's what the updated map should look like in the browser:



Signing Up with Amadeus for Developers and Saving Your API Credentials

Now that the map points to Barcelona, let's use the Amadeus for Developers API to see what points of interest there are in this beautiful city.

To do this, you first need to register with Amadeus for Developers to get access to its Self-Service APIs. Here are the steps that you need to take:

1. Fill out the account registration form (https://developers.amadeus.com/register) .
2. Activate your account using a link that you should receive via email.
3. Sign in to your account, then in the drop-down menu with your username at the top right, click **My Self-Service Workspace**.
4. Click **Create new app**, give it a name, and click **Create**.

Once the app is created, scroll down to the **App keys** area. From there, you can copy the key and the secret that will work with any Amadeus for Developers Self-Service API in the test environment. (If you want, you can read more (https://developers.amadeus.com/blog/test-and-production-environments-for-amadeus-self-service-apis-) about the differences between Amadeus for Developers test and production environments.)

The next step is to provide the API key and secret to your React application. When developing a production application, you would probably do this on the backend using something like dotenv (https://www.npmjs.com/package/dotenv). Since this is a simplified client-only application, you can read API credentials from a file.

In the `src` directory, create a new file called `credentials.json`, and paste the following code into this new file:

```
{
 "amadeus": {
   "key": "YOUR_API_KEY",
   "secret": "YOUR_API_SECRET"
 }
}
```

Replace `YOUR_API_KEY` and `YOUR_API_SECRET` with your app's actual API key and API secret, which you can copy over from your Amadeus for Developers self-service workspace (https://developers.amadeus.com/my-apps)

.

Note that if you're using Git while going through this tutorial, your `.gitignore` file already contains an entry for `credentials.json` to make sure your API credentials don't leak into version control.

Calling the Amadeus for Developers Points of Interest API, Then Saving and Displaying Data

Now that your React app knows your API credentials, you can add the code that calls the API to get the list of points of interest for a particular location.

Inside the `src/data` directory, create a new file called `amadeus.js` and paste the following code into the file:

```
import credentials from "../credentials.json"
export const getPointsOfInterestBySquare = async (square) =>
{
    const baseUrl = "https://test.api.amadeus.com/v1";
    const tokenRequest = await fetch(`${baseUrl}/security/oau
th2/token`,
        {
            method: "POST",
            body: `grant_type=client_credentials&client_id=
${credentials.amadeus.key}&client_secret=${credentials.amadeu
s.secret}`,
            headers: {"Content-Type": "application/x-www-for
m-urlencoded"}
        });
    const tokenResponse = await tokenRequest.json();
    const token = tokenResponse.access_token;
    const pointsOfInterestRequest = await fetch(`${baseUrl}/r
eference-data/locations/pois/by-square?north=${square.north}&
west=${square.west}&south=${square.south}&east=${square.east}
`,
        {
            headers: {"Authorization": `Bearer ${token}`}
        })
    const pointsOfInterestResponse = await pointsOfInterestRe
quest.json()
    return pointsOfInterestResponse.data;
};
```

This file contains one function, `getPointsOfInterestBySquare()`, that makes two API calls:

1. The first call exchanges your API credentials for an access token that you'll need to make the second call.
2. The second call goes to a Points of Interest API endpoint that takes four coordinates defining the borders of a location as a

"square" and receives the list of points of interest found inside that square.

For this demo, you'll request a new access token for each API call, though in a real-world scenario, it's better to introduce additional logic (https://developers.amadeus.com/self-service/apis-docs/guides/developer-guides/API-Keys/authorization/#using-the-token) to reuse a token and only refresh it around the end of its lifetime. If you're using Amadeus for Developers SDKs (https://github.com/amadeus4dev), they'll handle tokens for you.

Note that you can play around with the Points of Interest API before you even start calling it in your code. To do this, go to the Points of Interest API reference page (https://developers.amadeus.com/self-service/category/destination-experiences/api-doc/points-of-interest/api-reference) and explore the three endpoints that this API provides.

Let's now get back to `PointsOfInterest.js` and use the API calling code you just added.

First, add a new `import` statement:

```
import {getPointsOfInterestBySquare} from "./data/amadeus";
```

Replace the existing `import` statement for `react` with the following:

```
import React, {useEffect, useState} from "react";
```

Replace the existing `import` statement for `pigeon-maps` with the following:

```
import {Map, Marker, ZoomControl} from "pigeon-maps";
```

Then, inside the `PointsOfInterest()` function component, right after the `const [city, setCity]` declaration, insert the following code:

```
    const [pointsOfInterest, setPointsOfInterest] = useState
();
    const getPointsOfInterest = async () => {
        const poi = await getPointsOfInterestBySquare(city.lo
cation.square);
        return setPointsOfInterest(poi);
    }
    useEffect(() => {
        getPointsOfInterest()
    }, [city])
```

The code above:

- Introduces another state object to hold and update an array of points of interest
- Adds a `getPointsOfInterest()` function that calls the code inside `amadeus.js` and updates the state object holding points of interest
- Adds a `useEffect` hook wrapping `getPointsOfInterest()` in such a way that it's called every time the `city` state object

changes

Scroll down to the `PointsOfInterest` component's `return` statement and find the following in the very first line:

```
return (
```

Replace this with the code below:

```
return ( pointsOfInterest &&
```

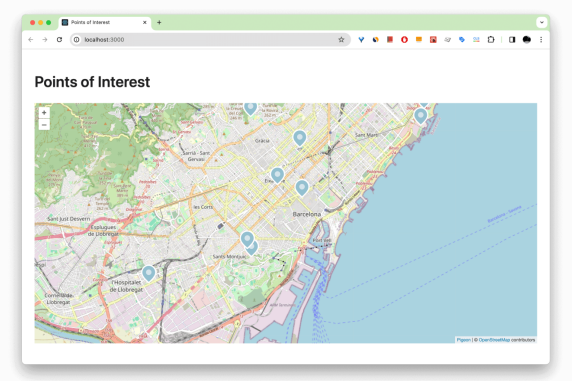
This makes sure that the React application doesn't try to render the map until `pointsOfInterest` is populated with the results of the Points of Interest API call.

Scroll down and insert the following code right after the `ZoomControl` element:

```
{pointsOfInterest.length > 0 && pointsOfInterest.map(location
=> {
  return (<Marker
          width={50}
          anchor={[location.geoCode.latitude, location.geoCod
e.longitude]}
       />
  );
})}
```

This piece of code takes the points of interest found in the current city and visualizes them as map markers.

At this point, your application in the browser should start displaying points of interest in Barcelona:



Adding a City Selector

The map currently displays points of interest in Barcelona, but there's a whole set of cities that Amadeus for Developers provides data for in the test environment. You can add a function that lets you display all these cities and enable switching between them.

Inside `PointsOfInterest.js`, replace the existing `import` statement for `antd` with the following:

```
import {Radio, Typography} from 'antd';
```

Between the `useEffect()` call and the start of the `return` statement, insert the following code:

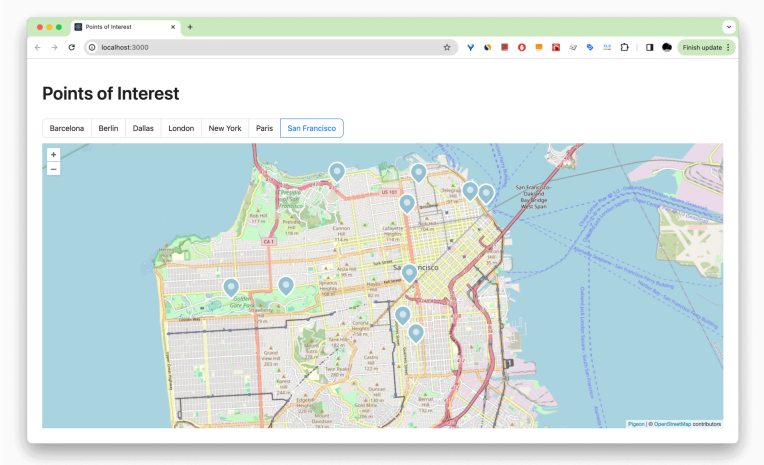
```
    const handleLocationChange = e => {
        e.preventDefault()
        setCity(predefinedLocations.find(x => x.name === e.ta
rget.value));
    };
```

Every time a different city is selected, this event handler function will set the `city` state object to a new value, which will trigger a new API call to fetch the list of points of interest in the new city.

Inside the `return` statement, between the two `div` blocks with the class names `title` and `map`, insert a new `div` with a group of radio buttons for selecting a city from the predefined list:

```
<div className="controls">
    <Radio.Group
        className="locations"
        value={city.name}
        onChange={handleLocationChange}
        size="large">
        {predefinedLocations.map(x => (
            <Radio.Button key={x.name} value={x.name}>{x.nam
e}</Radio.Button>
        ))}
    </Radio.Group>
</div>
```

If you open the browser now, you can see a row of radio buttons above the map that serve to switch between cities on the predefined list. Clicking any of these results will display a map of the city along with markers indicating points of interest obtained from the Amadeus for Developers API.



Enriching Point of Interest Markers

The application is almost ready, but let's enrich the features before wrapping up. The data that comes from the Amadeus for Developers Points of Interest API is not limited to location coordinates. It also includes the location name, a predefined category, a relative rank, and a set of tags that are free-form strings:

```
{
   "type": "location",
   "subType": "POINT_OF_INTEREST",
   "id": "9CB40CB5D0",
   "self": {
     "href": "https://test.api.amadeus.com/v1/reference-data/
locations/pois/9CB40CB5D0",
     "methods": [
       "GET"
     ]
   },
   "geoCode": {
     "latitude": 41.39165,
     "longitude": 2.164772
   },
   "name": "Casa Batlló",
   "category": "SIGHTS",
   "rank": 5,
   "tags": [
     "sightseeing",
     "sights",
     "museum",
     "landmark",
     "tourguide",
     "restaurant",
     "attraction",
     "activities",
     "commercialplace",
     "shopping",
     "souvenir"
   ]
 }
```


You can modify the way the application displays locations on the map by adding location names and the list of tags associated with each location.

Inside `PointsOfInterest.js`, replace the existing `import` statement for `antd` with the following:

```
import {Image, Popover, Radio, Tag, Typography} from 'antd';
```

Replace the existing `import` statement for `pigeon-maps` with the following:

```
import {Map, Overlay, ZoomControl} from "pigeon-maps";
```

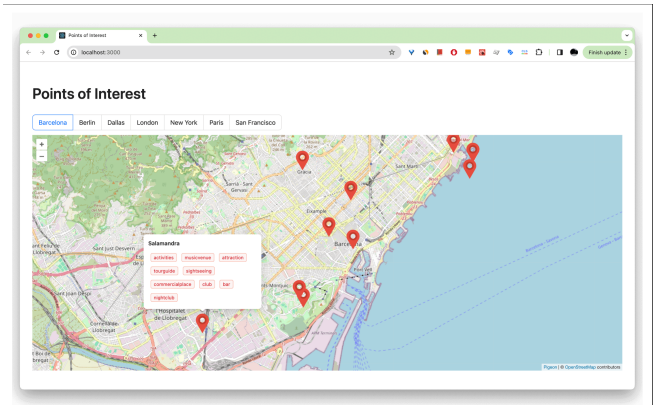
In the `return` statement, replace the entire `Marker` element with the following code:

```
 <Overlay
     anchor={[location.geoCode.latitude, location.geoCode.lon
gitude]}
 >
     <Popover
         title={location.name}
         content={location.tags && location.tags.map(x => <Ta
g color="red">{x}</Tag>)}
     >
         <Image
             src="./marker.svg"
             height={50}
             preview={false}
         />
     </Popover>
 </Overlay>
```

Finally, go to the `public` directory and create a new file called `marker.svg`. Open the file and insert the following SVG markup:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<svg xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns="http://www.w3.org/2000/svg" height="24" width="24" ver
sion="1.1" xmlns:cc="http://creativecommons.org/ns#" xmlns:dc
="http://purl.org/dc/elements/1.1/">
 <g transform="translate(0 -1028.4)">
  <path d="m12 0c-4.4183 2.3685e-15 -8 3.5817-8 8 0 1.421 0.3
816 2.75 1.0312 3.906 0.1079 0.192 0.221 0.381 0.3438 0.563l6
.625 11.531 6.625-11.531c0.102-0.151 0.19-0.311 0.281-0.469l
0.063-0.094c0.649-1.156 1.031-2.485 1.031-3.906 0-4.4183-3.58
2-8-8-8zm0 4c2.209 0 4 1.7909 4 4 0 2.209-1.791 4-4 4-2.2091
0-4-1.791-4-4 0-2.2091 1.7909-4 4-4z" transform="translate(0
1028.4)" fill="#e74c3c"/>
  <path d="m12 3c-2.7614 0-5 2.2386-5 5 0 2.761 2.2386 5 5 5
2.761 0 5-2.239 5-5 0-2.7614-2.239-5-5-5zm0 2c1.657 0 3 1.343
1 3 3s-1.343 3-3 3-3-1.3431-3-3 1.343-3 3-3z" transform="tran
slate(0 1028.4)" fill="#c0392b"/>
 </g>
</svg>
```

Instead of the default marker that comes with `pigeon-maps`, every point of interest now displays using a custom marker defined in the SVG file. When you hover over the marker, you can see a popover card displaying the name of the current point of interest, as well as a list of associated tags to help you get a general idea of what to expect from that location:



Congratulations! You now have a working React application that sources points of interest in various cities from the Amadeus for Developers API and displays them on a map along with additional

details.

If you were just skimming through without following, feel free to check out
this GitHub repository (https://github.com/gorohoroh/PointsOfInterestFinder)
with the completed application.

Summary

After following this tutorial, you learned about the
Amadeus for Developers Points of Interest API (https://developers.amadeus.com/self-service/category/destination-experiences/api-doc/points-of-interest)
and how it can be used in a simple React application.

In addition to Points of Interest, Amadeus for Developers has lots of useful travel APIs for you, including APIs for
flight booking (https://developers.amadeus.com/self-service/category/flights/api-doc/flight-offers-search)
,
hotel ratings (https://developers.amadeus.com/self-service/category/hotels/api-doc/hotel-ratings)
, and
tours and activities (https://developers.amadeus.com/self-service/category/destination-experiences/api-doc/tours-and-activities)
.

Whether you are a developer, a startup, or a leading travel brand, Amadeus for Developers APIs (https://developers.amadeus.com/self-service)
enable you to create useful, high-quality applications and quickly deliver them to market.