MazeMap

INTEGRATION GUIDE

MazeMap in 3rd Party (Mobile) Applications

June 10, 2020

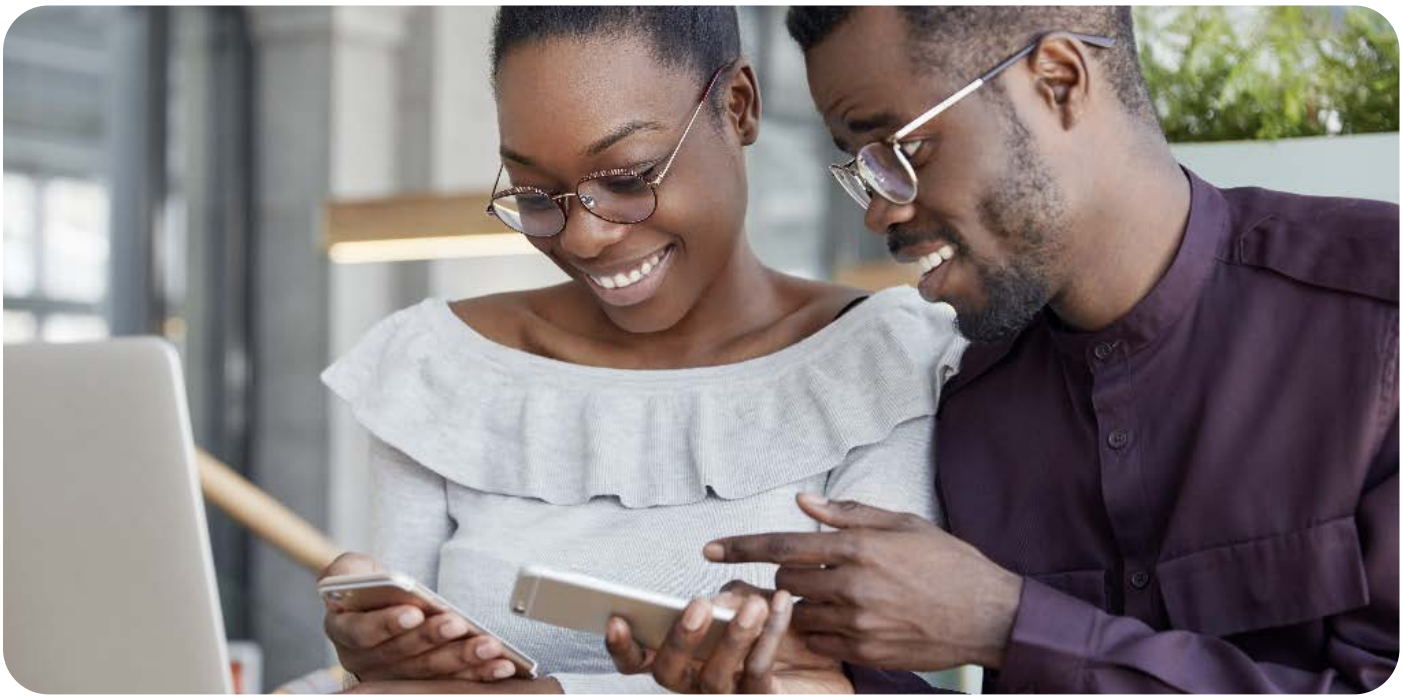
What is described here

This document tries to give a quick description of different ways of integrating MazeMap into 3rd party applications. It's relevant for decision makers and iOS / Android / Web developers.

Contents:

Q: Does MazeMap provide a mobile SDK to integrate maps?

A: Instead of specific iOS/Android SDKs, we provide a common JS API which should be sufficiently easy to integrate with, is more flexible to work with, and is reusable across all your platforms- both mobile and web.

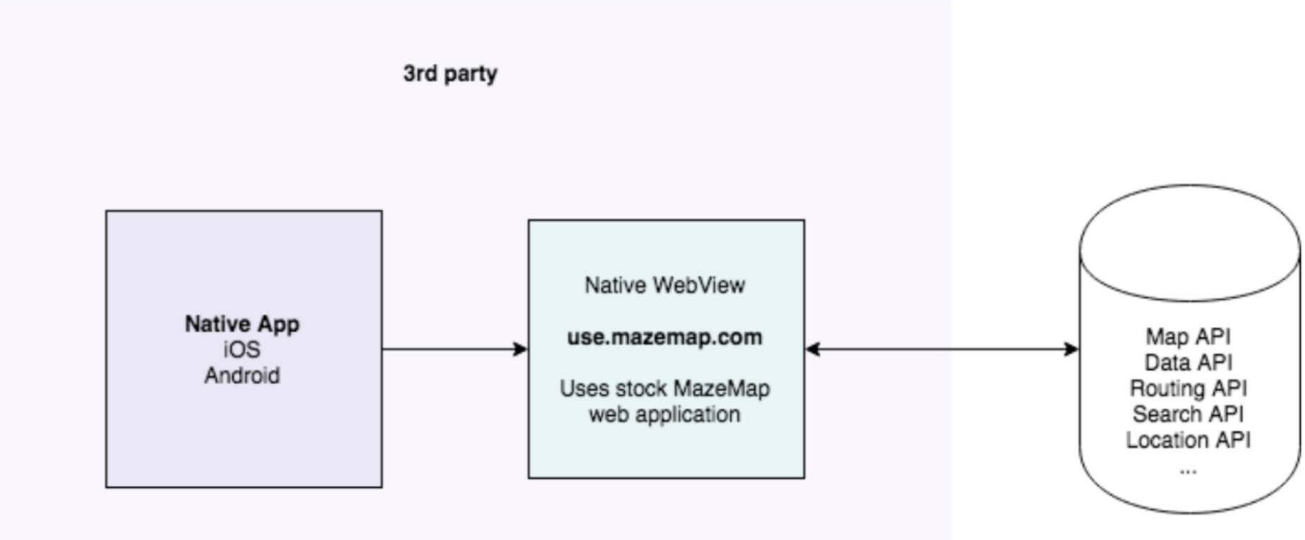


1

Embedding the default MazeMap web app

The easiest way to get quick results

This is by far the easiest way of simply getting indoor maps into your application. The default MazeMap web application located at https://use.mazemap.com can be embedded into web pages or native mobile/desktop applications using standard IFrames for browsers, or WebViews in Android / iOS.



Pros

You get all(*) the basic functionality of the standalone MazeMap web app, such as indoor location, navigation, search and the maps without writing a single line of code. If it works in https://use.mazemap.com, it should work in your embedded version as well. Also, future application updates will automatically be visible in the embedded version, since it would always run the latest version of MazeMap.

*) Some functionalities/features might not be available out-of-the-box for a native integration, such as Room Booking login, due to limitations and restrictions with native web views. Contact us to find out more.

Cons

You don't get to modify the user interface and control which elements you need and don't need, and you don't get to style the interface colors to match your application.

Recommended when

- You need a quick pilot to test how MazeMap would add value to an existing app.
- You want all the default functionality of MazeMap
- You don't need very customised functionality or behaviour
- You don't need a white labelled map interface.

2

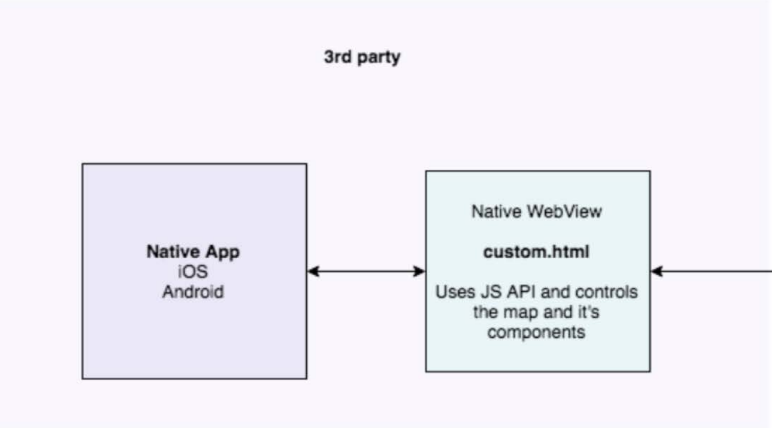
Integrating a map using MazeMap JS API

Implementing a custom map with the MazeMap JS API

This method is used when you want to customize the user experience and control which UI components you want in your map interface.

Basically, you write your own mini map application in a custom html file (i.e. **custom.html**) and use this file in your own WebView component (Android / iOS).

The mini web application will be able to use the JS API components and you can control the behaviour with great detail. The components should be easy to understand and use, and we provide many small examples of how to use each component. See https://www.mazemap.com/js-api-latest/ for details.



Pros

- You get to control which UI elements to show at any moment.
- You get to style the application to fit your mobile app branding

Cons

You need to write some application logic

Recommended when

- You want a more simplified map experience without unnecessary “default MazeMap” components
- You want to customize or limit the user experience
- You want to brand the map interface to better fit the rest of the application

3

Integrating with Indoor Location

MazeMap can work with different options for indoor location, but here are some guidelines and facts that often come up:

Cisco DNA Spaces

Our recommended indoor location technology is the Cisco DNA Spaces integration. When using this technology, indoor location will automatically work in the stock MazeMap web application everywhere it is integrated/embedded. When using the JS API, it is also supported and will work by using built-in location API functionality.

Learn more

To find out more about Indoor Location, please contact us at sales@mazemap.com, or go here: https://www.mazemap.com/indoor-positioning.

Using other Indoor Location providers, such as Bluetooth beacon technology

The MazeMap web application supports injection of generic location updates from an external source. That means that a native app could retrieve a coordinate from a Beacon source and then inject this into the MazeMap web app client.

A similar approach would work with the JS API, where you would be able to control the BlueDot location using a javascript injection command from the native layer into the WebView component.

The main idea here is that the “3rd party provider” is responsible for gathering location data and injecting it into the MazeMap application. The MazeMap map only needs coordinates as Latitudes, Longitude and Floor level to show a blue dot.