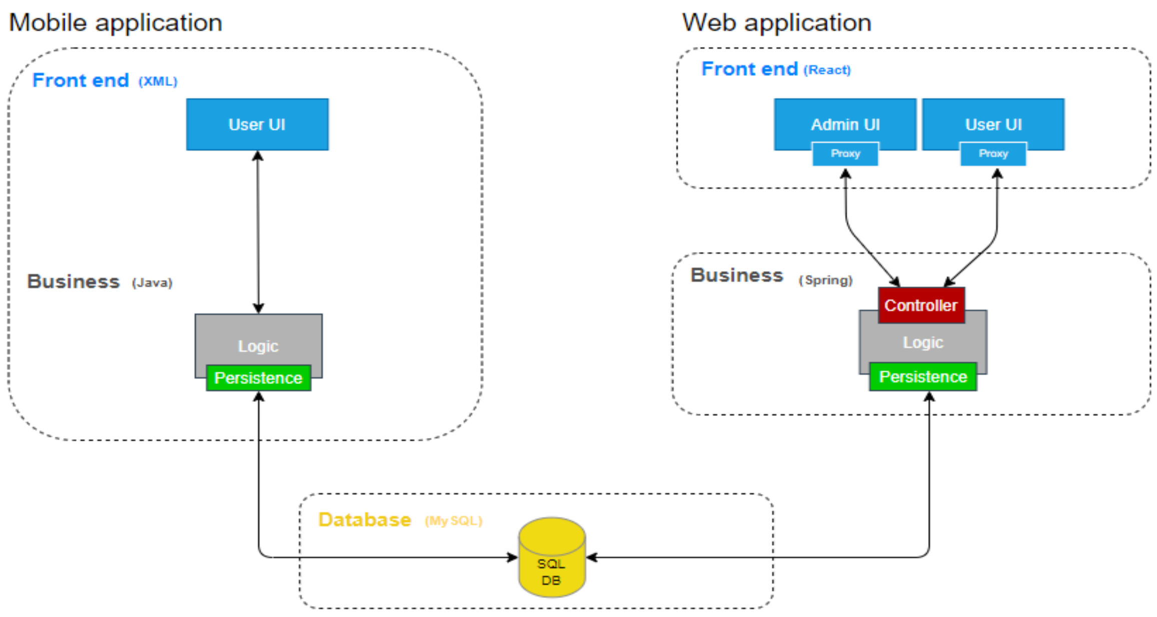
Guidance Architecture Document

Short summary

The Guidance Application is a web application made to ease maneuvering throughout a building for a user in both normal situations and in case of potential disasters. Once a user's mobile device is connected to a building's WiFi network, floor plans that the administrator has access to edit are shown to them, displaying their location and the population in different rooms throughout the building. As well as that, the user has the ability to choose a start and end point and the app will create a path to then end and display it for the user to find the least way to the end with meeting as few people as possible.

Table of Content

High Level Architecture



Disclaimer: Our team follows the C4 model of architecture, but this diagram was created before this decision. While our PO's have shown preference towards C4, this is a school project and this diagram is satisfactory for the range of our project and with prolonged discussion inside the development team, we have decided to leave it as it is, because it would take extended resources and research to rework into the C4 model.

- **View** This layer shows the data to the user and sends requests to the proxy layer

- **Proxy** The proxy layer receives requests and ensures that they will be sent via the right port to the API

- **Controller** The controller maps the fetches to send multiple requests to the logic layer such as: get, set and save.

- **Logic** Here the data is processed to data that is usable for the Persistence/View layers

- **Persistence** Here the data is picked up from the database and this layer stores data on the database

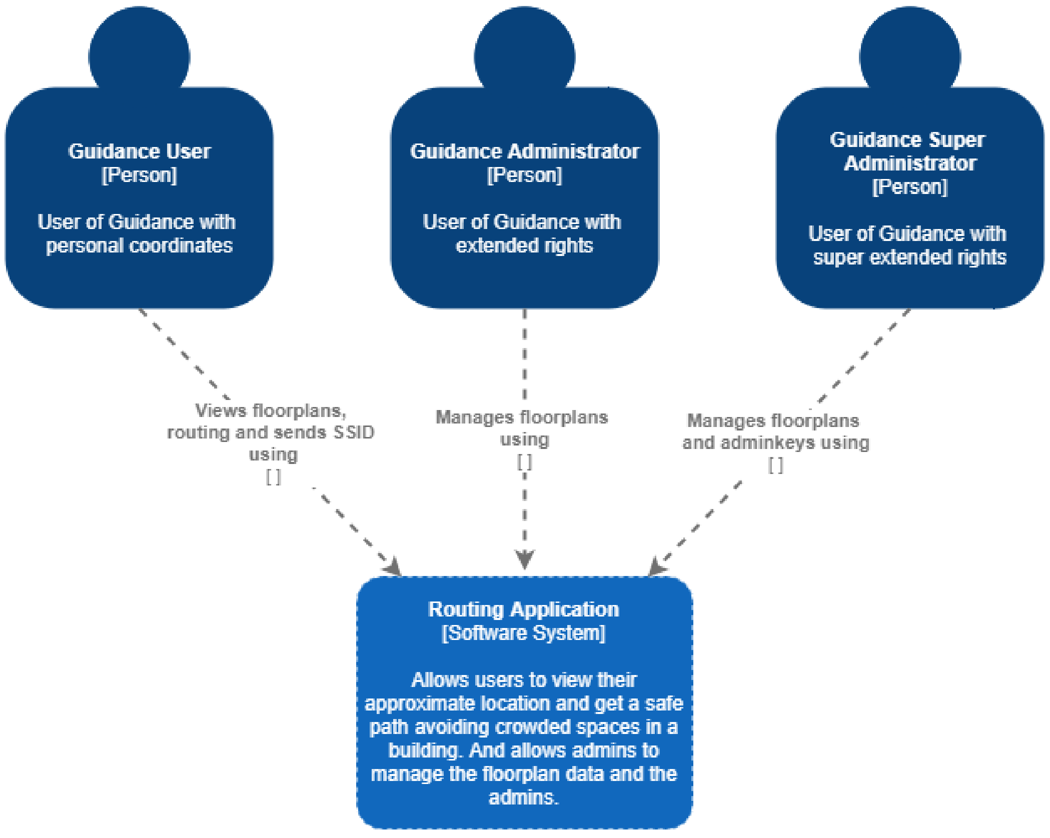
- **Database** This is the MySQL database where the floorplans and the coordinations are stored

Architecture motivation

The mobile application is a seperate application from the web application. Its purpose is to find Ibeacons and calculate the location of the user, this will be sent to the same database as the web application is using. The web application's purpose is to show the correct data to the right user. It has 2 front ends for the admin and the normal users, this is because an admin should be able to manage the floorplans. The web application’s frontend and backend are separately deployable. The mobile application’s back- and frontend however are not. We went for this design and not, for instance, a microservices architecture, because all of the software we implement needs to be compatible with each other. We wanted to keep the mobile application’s architecture as simple as possible so it is a really low demanding application in order for it to easily run on every user’s mobile phone.

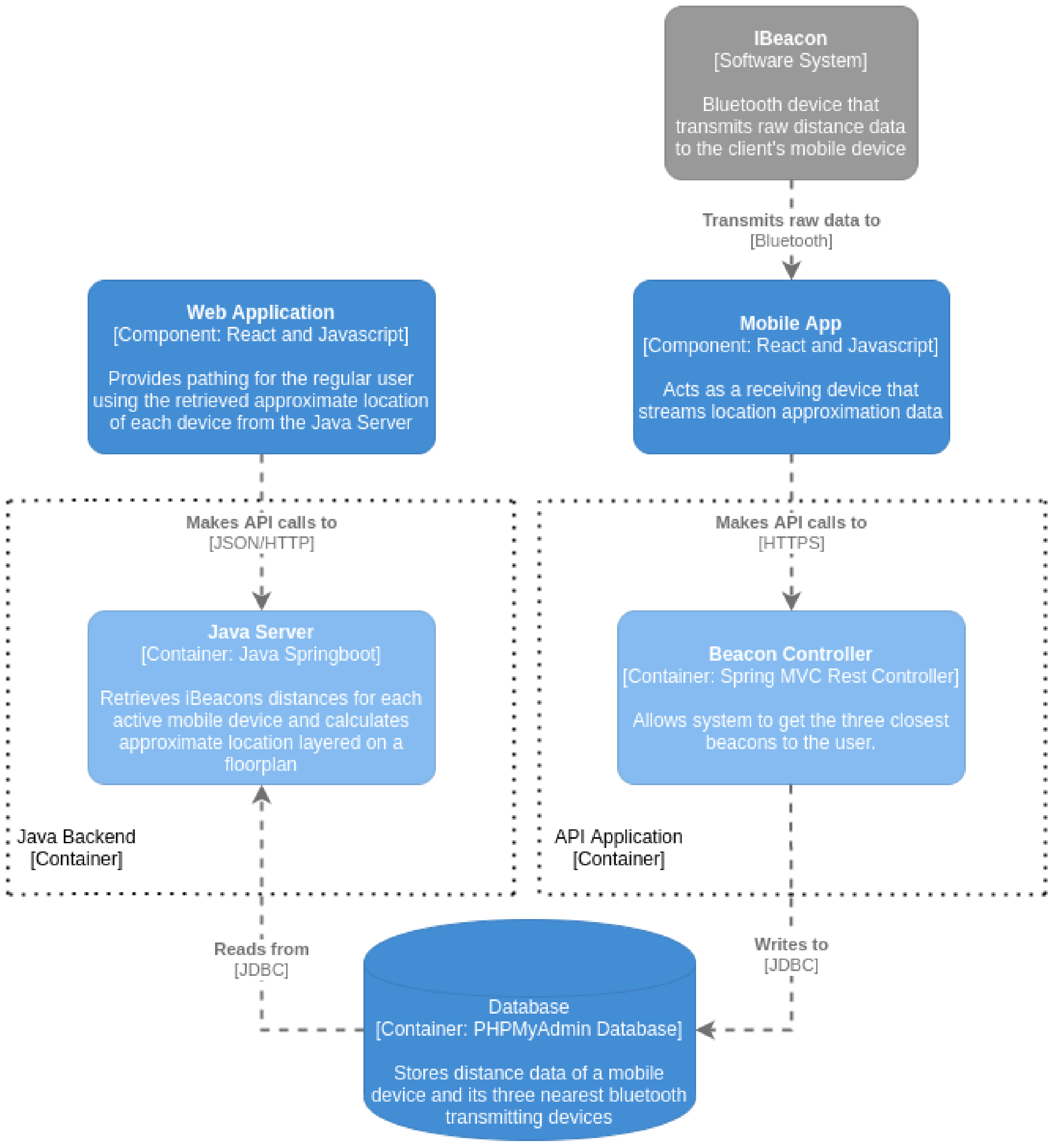
Context Diagram

Guidance users will be able to see their location and if asked for their routing on the floorplan. This way users are able to safely move through a building. Admins will be able to manage floorplans on a admin page and super admins will be able to manage the admin keys on a super admin page.



Guidance User
[Person]
User of Guidance with personal coordinates
Guidance Administrator
[Person]
User of Guidance with extended rights
Guidance Super Administrator
[Person]
User of Guidance with super extended rights
Views floorplans, routing and sends SSID using
[ ]
Manages floorplans using
[ ]
Manages floorplans and adminkeys using
[ ]
Routing Application
[Software System]
Allows users to view their approximate location and get a safe path avoiding crowded spaces in a building. And allows admins to manage the floorplan data and the admins.

Triangulation


IBeacon
[Software System]
Bluetooth device that transmits raw distance data to the client's mobile device
Transmits raw data to
[Bluetooth]
Web Application
[Component: React and Javascript]
Provides pathing for the regular user using the retrieved approximate location of each device from the Java Server
Mobile App
[Component: React and Javascript]
Acts as a receiving device that streams location approximation data
Makes API calls to
[JSON/HTTP]
Makes API calls to
[HTTPS]
Java Server
[Container: Java Springboot]
Retrieves iBeacons distances for each active mobile device and calculates approximate location layered on a floorplan
Beacon Controller
[Container: Spring MVC Rest Controller]
Allows system to get the three closest beacons to the user.
Java Backend
[Container]
API Application
[Container]
Reads from
[JDBC]
Writes to
[JDBC]
Database
[Container: PHPMyAdmin Database]
Stores distance data of a mobile device and its three nearest bluetooth transmitting devices

We need to pinpoint the location of the user. To do that without bothering the user too much, we decided to use triangulation. The only thing the user will need to do is a single action on his/her phone. What we need for triangulation are 3 distances between the user's phone and another object.

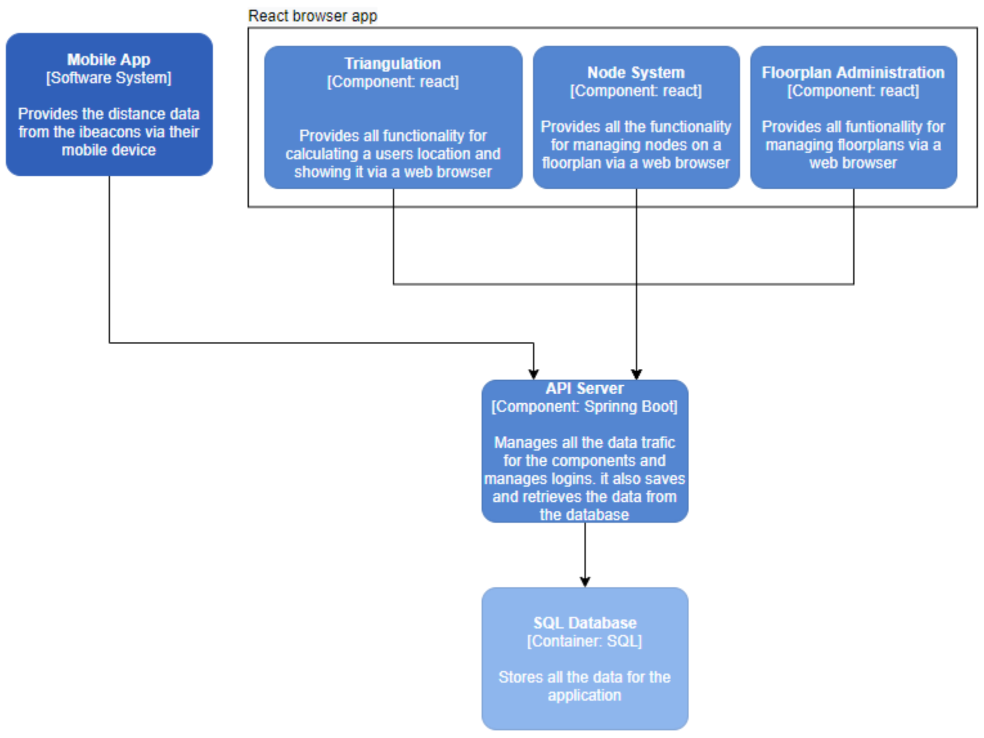
We were able to obtain those 3 distances in 2 ways:

- We could use access points and then obtain the distances from the access points to the user.
- We can use iBeacons and intercept the signals on the user's phone. The signal strength is incorporated in the signals. We can convert the signal strength back to a distance range where the user must be at.

We chose the iBeacons because the first option includes too much infrastructure and it becomes too much work for us with the time we have available.

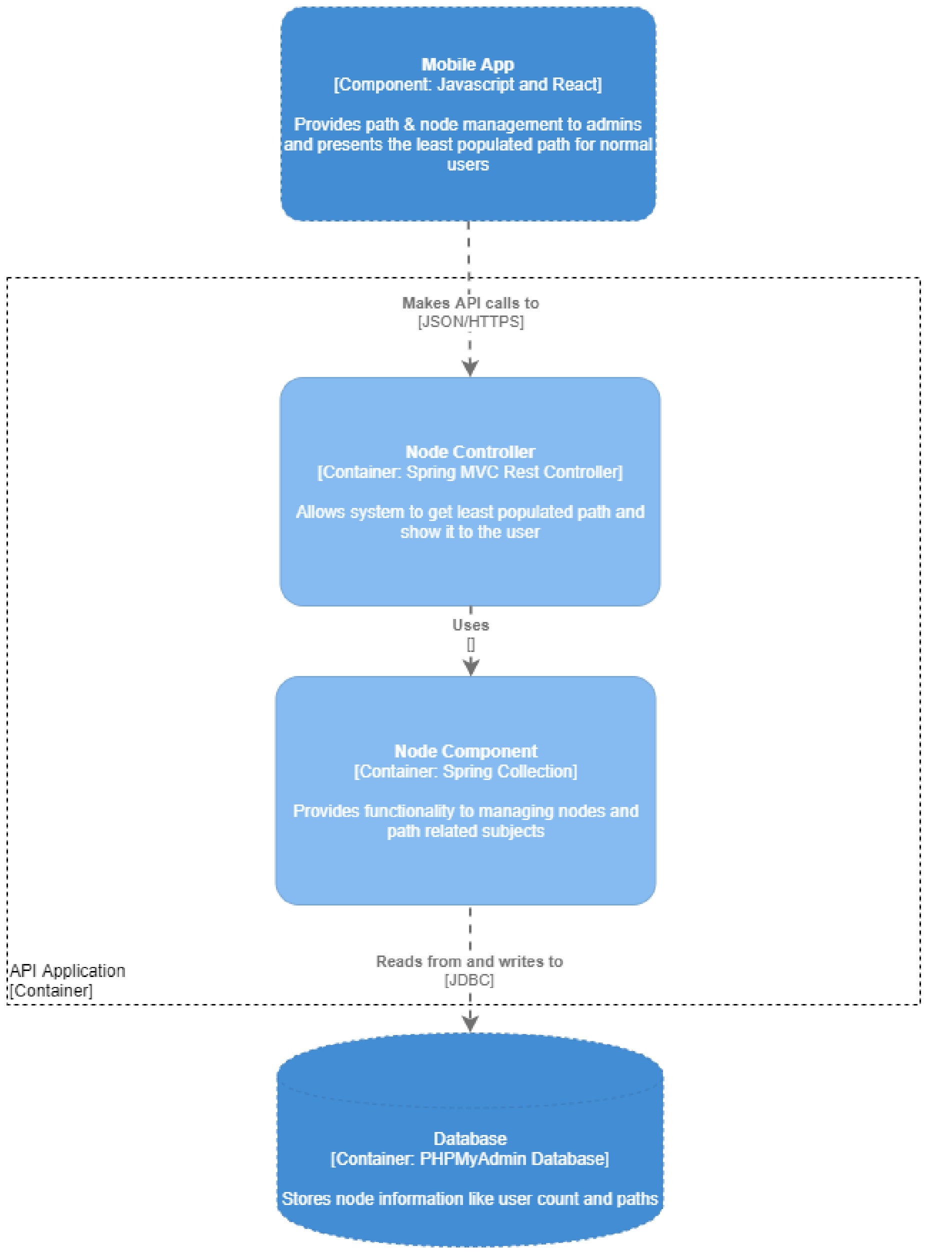
All the user has to do is install an app on his/her phone and open the app. The app will be able to run in the background.

Node System


React browser app
Mobile App
[Software System]
Provides the distance data from the ibeacons via their mobile device
Triangulation
[Component: react]
Provides all functionality for calculating a users location and showing it via a web browser
Node System
[Component: react]
Provides all the functionality for managing nodes on a floorplan via a web browser
Floorplan Administration
[Component: react]
Provides all funtionallity for managing floorplans via a web browser
API Server
[Component: Sprinng Boot]
Manages all the data trafic for the components and manages logins. it also saves and retrieves the data from the database
SQL Database
[Container: SQL]
Stores all the data for the application

In the start our team decided we wanted to use a grid system for pathfinding after working on it we found out that it was best to calculate the path in our client side.
After the first delivery we got feedback from an expert that this system wasn't feasible for this project and that we were better off creating a simpler system which uses pre created routes. That's how we came up with the node system this time we adapted it to fit our project better. The idea is that a map admin creates nodes and places them on an intersection after the map admin places all the needed nodes he/she can link them this will create the pre set paths. The map admin can also define if a node is a destination, intermediary or stair node. Then the map admin is supposed to add the ibeacons to the location they are in the actual building. The routing Algorithm will be built for working with this system.

Path System


Mobile App
[Component: Javascript and React]
Provides path & node management to admins and presents the least populated path for normal users
Makes API calls to
[JSON/HTTPS]
Node Controller
[Container: Spring MVC Rest Controller]
Allows system to get least populated path and show it to the user
Uses
[]
Node Component
[Container: Spring Collection]
Provides functionality to managing nodes and path related subjects
API Application
[Container]
Reads from and writes to
[JDBC]
Database
[Container: PHPMyAdmin Database]
Stores node information like user count and paths

In the start we wanted to use a pathfinding system that used an algorithm like the Dijkstra pathfinding algorithm. This turned out to be too much to implement in the short time period we had. We chose this system in consultation with Philips because this system would eliminate the implementation of a big algorithm. The system has an admin page where you can create, edit and delete paths. In here you can add and remove nodes, this way the path will be created between those nodes. Then a user on the landing page when wanting to go in or out the building can retrieve the least populated route through the building and this will be displayed on the floorplan on the landing page.