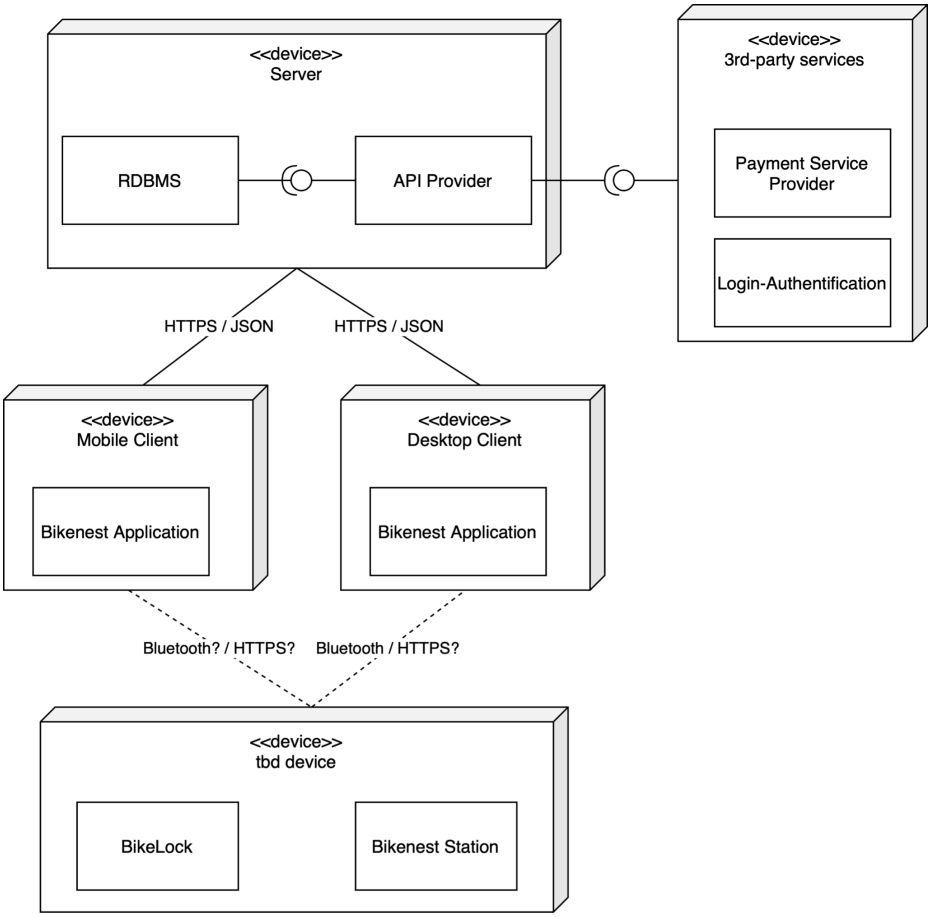
Software architecture description

AMOS SS2021

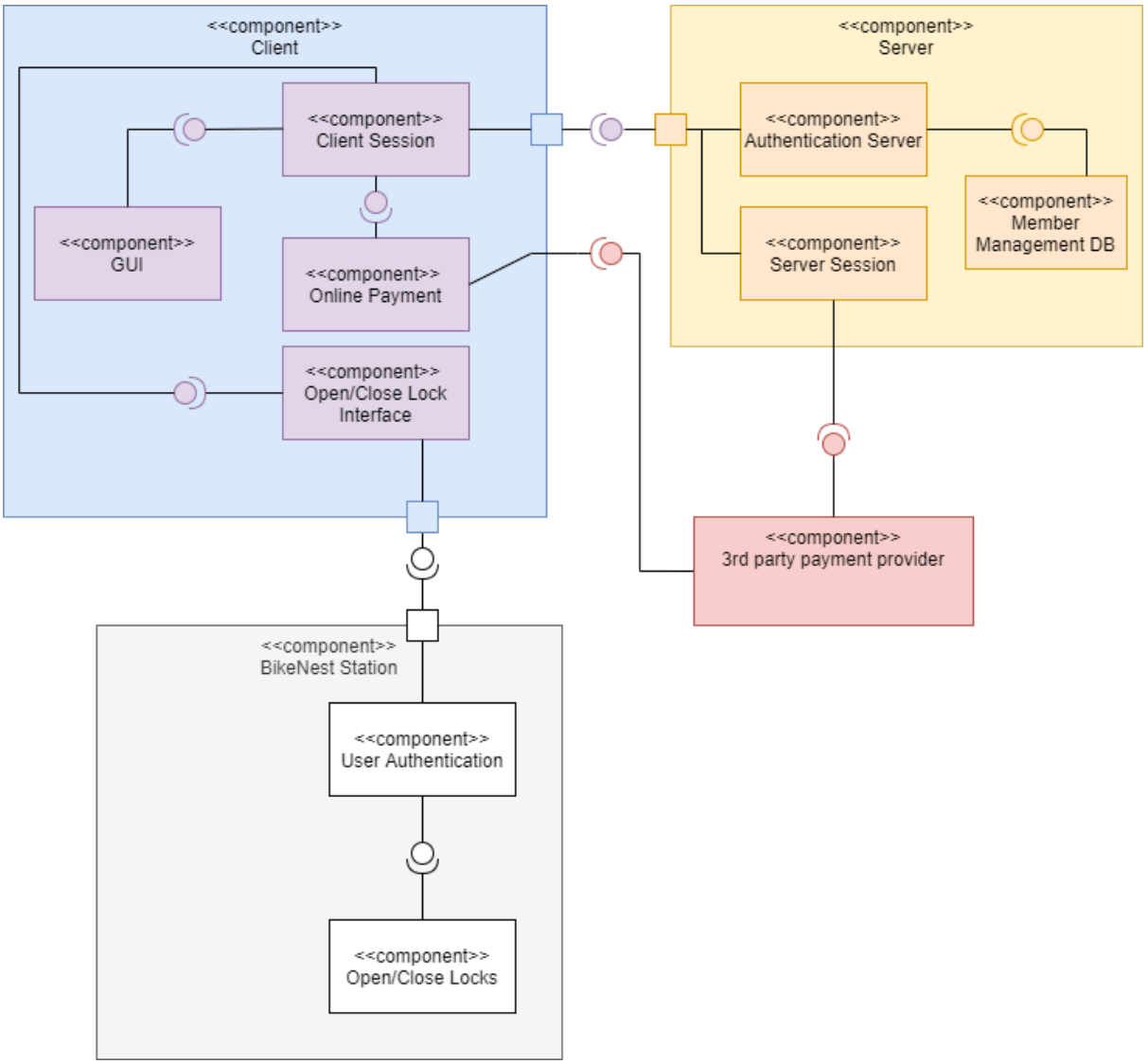
Project 7

Runtime Components



Based on the current status we do not have all necessary requirements and thus not a running environment. Therefore, another runtime perspective via activity diagrams can be given at a later point.
Currently, we have different components which interact with each other during Runtime.
The user's mobile client communicates on the one hand with the Server via HTTPS/JSON to Login, get Bikenest Station Locations and finally to get unlocking credentials. The exact (un)lock mechanism is not set. It could be based on QR-Codes or PINs generated by the Open/Close Lock Interface (see Code Components). To (un)lock the Bikenest Station or BikeLock the Mobile Client might communicate via Bluetooth or HTTPS. However, the exact mechanism depends on the Lock, which is not specified yet.
Besides the Mobile Client there is the Desktop Client. Its main purpose is also the reservation of BikeLocks, but also for administrative purposes. Therefore, the Desktop Client can also communicate with the Server and possibly with the Bikenest Station and BikeLock. Thus, it is able to get the current used locks and might be able to (un)lock the Lock from a distance and further to be used for to be defined functions.
The Server is used to store data in the RDBMS and it provides an API which handles and coordinates the different requests, like database requests, user and lock authentication and so on. It is therefore connected to 3rd-party services, which are also not yet defined but include Payment Service Providers and maybe Login-Authentication via e.g. Google, PayPal.

Code Components



We opted for a component diagram to show an overview of all components we need to integrate in our code. Our Basic components will be the Client, this will be a portable user device, the Server and the Bike Nest Hardware. The Bike Nest Hardware consists of a Cage which is locked with a “Smart” Lock and an inner lock to lock up your bike in place. The “Smart” Lock will have an interface to authenticate the Client against our Bike Nest and open it for him/her. The technology is not set yet but we assume that the lock-interface will have no internet connection, so the authentication has to work offline as well. At this point we consider Bluetooth as the best choice.
To communicate between Server and Client we will have a Web-Interface. The Client has to log in to start a session. He/She will have to authenticate to our server to use our service. Our Server will start a Server Session to communicate the Member Data to the Client. For the reservation of a Slot we use a 3rd party payment provider.

Underlying Technology Stack

Client

- Mobile: iOS, Android
- Web browser

Developing

- UI frameworks and libraries: React Native, Node.Js
- Backend: Express / Flask (still under consideration)
- Programming languages: JavaScript, TypeScript
- Other frameworks and platforms: Expo
- Developing tools: Visual Studio Code or Intelliji recommended but not mandatory
- Databases: SQlite (current idea for the prototype, options like MongoDB Realm and Firebase under consideration)
- APIs: Third party providers for login and payment

Description about the technologies

React Native:
React Native is a framework that builds a hierarchy of UI components to build the JavaScript code. It has a set of components for both iOS and Android platforms to build a mobile application with a native look and feel.
We choose it because it has all the needed requirements we have so far and is a popular framework with a lot of community support. Components are reusable, single code as it is a cross platform. In case of need we could also build native code.

Express:
Lightweight Node.js web application framework providing a robust set of features for web and mobile applications.

Flask:
Lightweight WSGI web application framework in Python. It doesn't enforce any dependencies or project layout, leaving the option to choose suitable tools and libraries.

Developing tools:
Visual Studio Code extension that provides a development environment for React Native projects is recommended but any other IDE that can work with React Native is also accepted.

Expo:
Expo is a toolchain built around React Native to help you build native iOS and Android projects using JavaScript/TypeScript and React. Expo enables you to build cross-platform native apps using only JavaScript. Benefits of using Expo is no need to use Xcode or Android Studio. You can build managed and bare workflows, Apps are built with the managed workflow using the expo-cli, the Expo Go app available for mobile devices and access to SDK. In case of need we can also opt to go bare workflow that will build native code.

SQlite:
SQLite is a lightweight software library providing a relational database management system. SQLite's characteristics are:

- self-contained: requires minimal support from the the OS or external libraries
- serverless: does not require a server to run, is integrated with the application which accesses it
- zero-configuration
- transactional: ACID-compliant -> Atomic, Consistent, Isolated, and Durable