**2. Overall Description**

* 1. **Product perspective**

As a result, an entirely new mobile app will be developed that uses Yandex maps as additional service. Tracking the user's location will be performed with the help using the built-in sensors of the mobile device. Yandex Map API is responsible for displaying available stores and building the path to them.

The detailed architecture of the program and its interaction with third-party API will be described below.

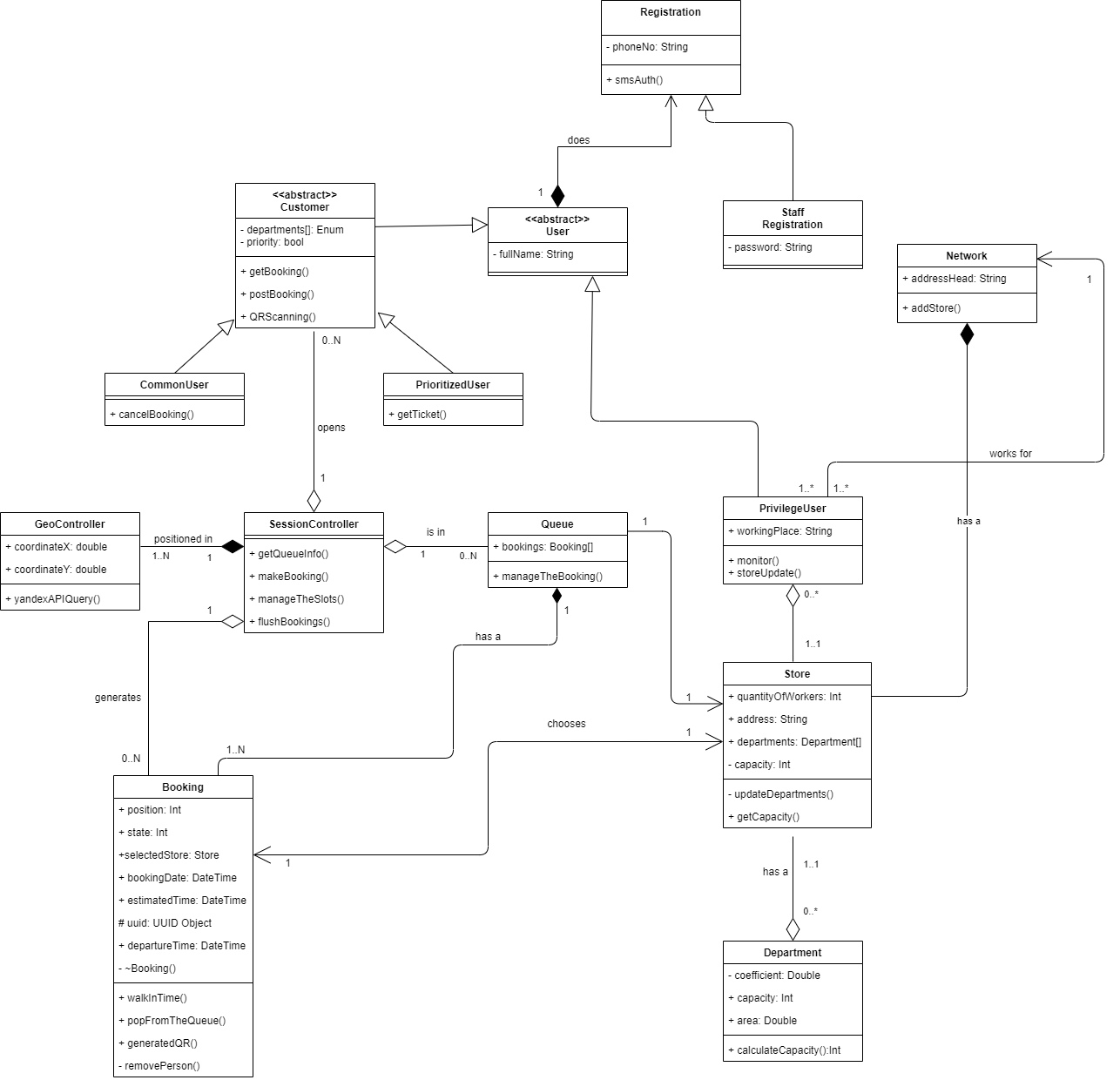


Figure 1 Class diagram

As can be seen in the screenshot, the entry point of the program can be considered as the *Registration*, which is divided into two branches: the end-user registration and staff registration. Class *Customer*, after inheriting the qualities of the *User* Class, also can be divided into two parts: the one group of people who will use the application - *CommonUser*, with the consequent possibility of canceling reservations online, and the second group *PrioritizedUser*, which will receive its ticket to the queue upon arrival at the store. The *SessionController* class, which controls the process of Queuing in general, uses the *GeoController* class, which provides functionality for tracking the user position and building a route to the selected store. The *YandexAPI* library is responsible for the rest of the map functionality. The *SessionController* class contains a key *makeBooking* method that implements the logic of introducing a new user to the queue, storing more detailed information, such as the selected store and the capacity of its departments, the user’s state (going to the store, making purchases, leaving the store), and generating a QR code that is necessary needed for going to the store. User’s position data is transmitted via a mobile device; therefore, it is important to grant the appropriate rights to use the app. The Queue class stores an array of all bookings for the store. The purpose of the *Store* and *Department* classes is to prevent congestion and create a more efficient distribution of incoming people, to do this, we need to have additional information about the store itself that the owner provides. Let us say the user selects priority departments, and we know their capacity, which will help us calculate the approximate number of customers for each department. The *PriviligeUser* class is the store staff that can access the app and its advanced settings. The Network class groups stores of a single registered trademark. It is worth noting that the above functionality is performed on the server-side.

Using the status diagram, the typical purchase lifecycle is described below: from the start of registration to the end of the session.

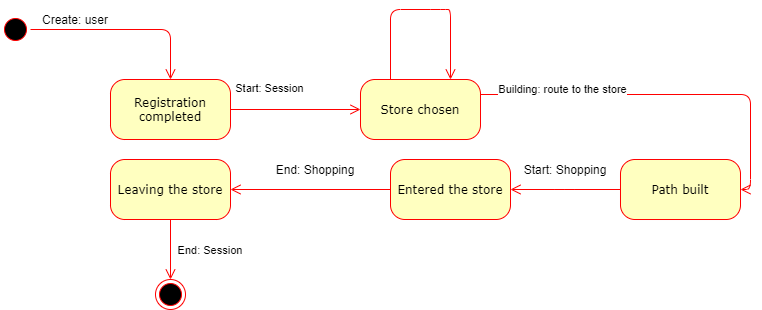


Figure 2 State diagram