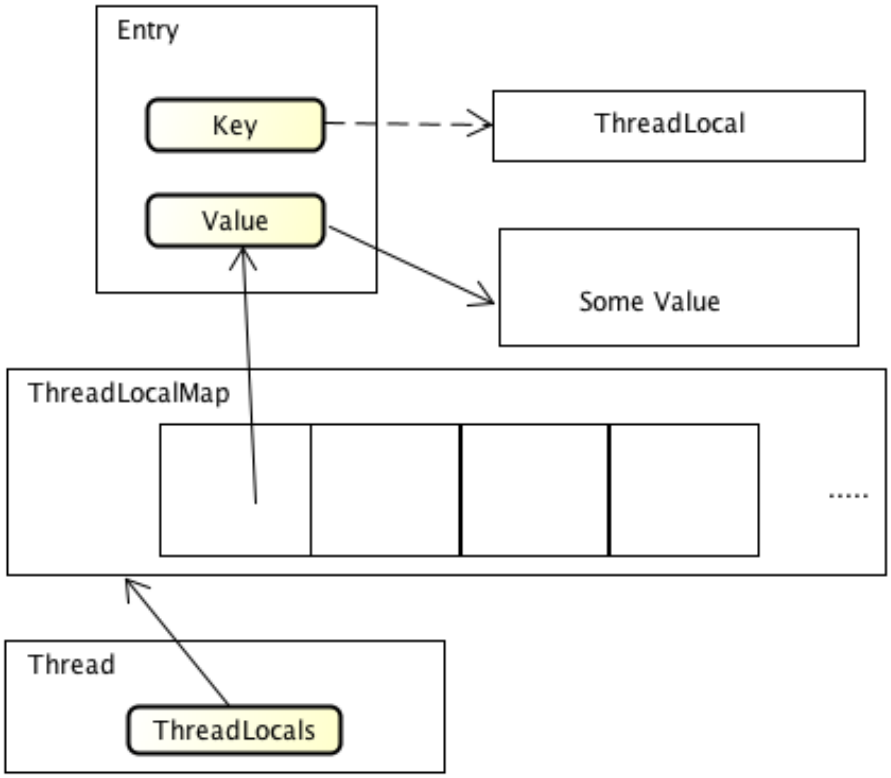
We using “Thread Local Storage” pattern to store the details of who is authenticated. To indicate a user is authenticated is to set the SecurityContextHolder directly which uses a ThreadLocal to store these details. In order to get the current username, we first getting a SecurityContext, which is obtained from SecurityContextHolder. This SecurityContext keep the user details in an Authentication object, which is obtained by calling the getAuthentication() method.
When request comes to back-end, authentication filter intercepts a user request, extracts user details from JWT token and sets it into SecurityContextHolder. This class is basically a wrapper for thread local storage, so each thread accosted with a requests and all child threads have access to currently authenticated user via static method getAuthentication(). Also with a help of this pattern there is no need to pass user details for all methods because SecurityContextHolder provides a single global access point to authenticated user.



Message Queue

Our application is sending a lot of mails to landlords and tenants about different events, like new application, approves, declines etc. For instance, when user sends an application to rent a property front-end sends a request to back-end to create this application and this is a part of request processing to send email to landlord about this action.
We had a problem with it, because Mail Sending Server can be unavailable and slow sometimes which is increasing processing time of request. That was a big issue for the application performance.

To solve this problem we decided to use **asynchronous** approach. As sending of email is not a required action to get request done, we used **Message Queue to decouple heavyweight processing** (call a notification-service to send an email). A message queue is a form of asynchronous service-to-service communication widely used in microservices architectures. Messages are stored on the queue until they are processed and deleted. Each message is processed only once, by a single consumer (in our case - notification-service).