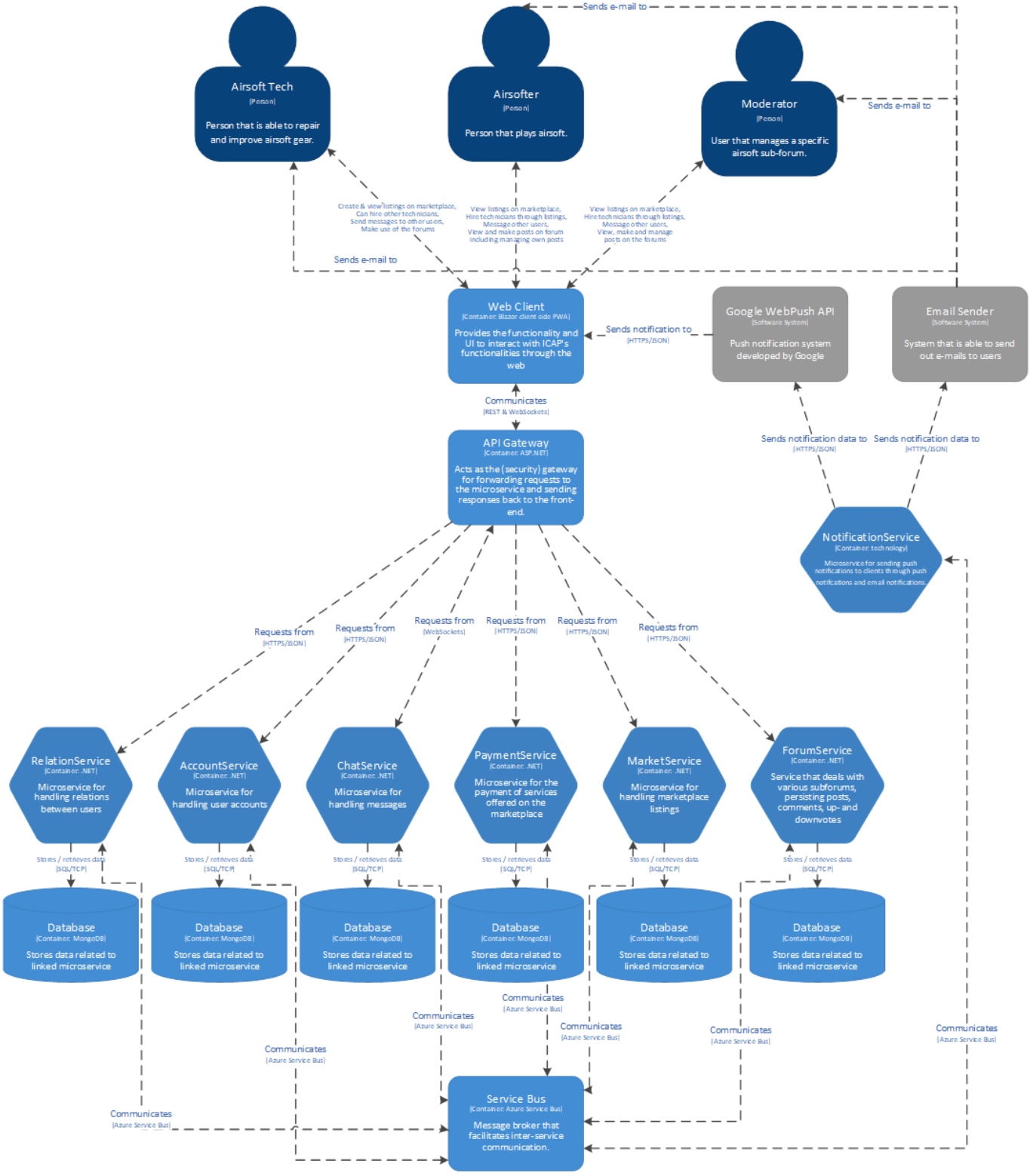
Messaging in ICAP

ICAP is distributed application that makes use of the microservices architecture. The microservices within ICAP have to be able to communicate with each other using a certain messaging system. The platform used to facilitate this messaging will be Azure Service Bus. This document will show the events that will be consumed and produced by the various microservices of ICAP. The general message structure that will be re-used across all microservices will also be described here.

Messaging Structures

Within ICAP there are various microservices with various use-cases. If we look at the image below you'll get a better idea of what those services are.



Most of these services provide data and functions that do not require real-time updates that can be pushed from the service to the web client, except for the actual messaging feature. All these services will therefore be using a simple form of RESTful request-response communication that holds open the connection when that function needs to acquire information from a different part of the application that is not contained within its own business logic.

When it comes to real-time messaging however, it is better to make use of websockets, because there is less latency overhead with the use of websockets and does not require an open HTTP connection in the same way that a long-polling request might need. Compared to short-polling it also puts less strain on the server and

client because there is no constant need to send a message from the client to the server, asking if there is any new data available.

When a request is made to a service that requires information from another service, we can hold open that request, attach an asynchronous process to it that sends an HTTP request to the required endpoint and keeps the original request open. When the response is given the original request can then be responded to.

ICAP also uses the PUB/SUB pattern in order to update any services asynchronously.