Mail App Architecture

version 1.0.0

gabe@seedess.com

Technical Architecture for Native Mail App

UI and UX

The mail app UI will follow Material Design and UX will be developed with React-Native.

The pubsub channel is simply to notify realtime information such as new emails, classification or emails from AI etc.
The HTTP REST channel is for transferring large data such as email body and attachments.

Mail AI

The Mail AI backend can either support pubsub or expose a REST interface.
The AI subscribes to a /new-mail endpoint/ channel.
Each new email is sent to the /new-mail endpoint and the resulting analysis is published to the same endpoint.

Mail App Architecture

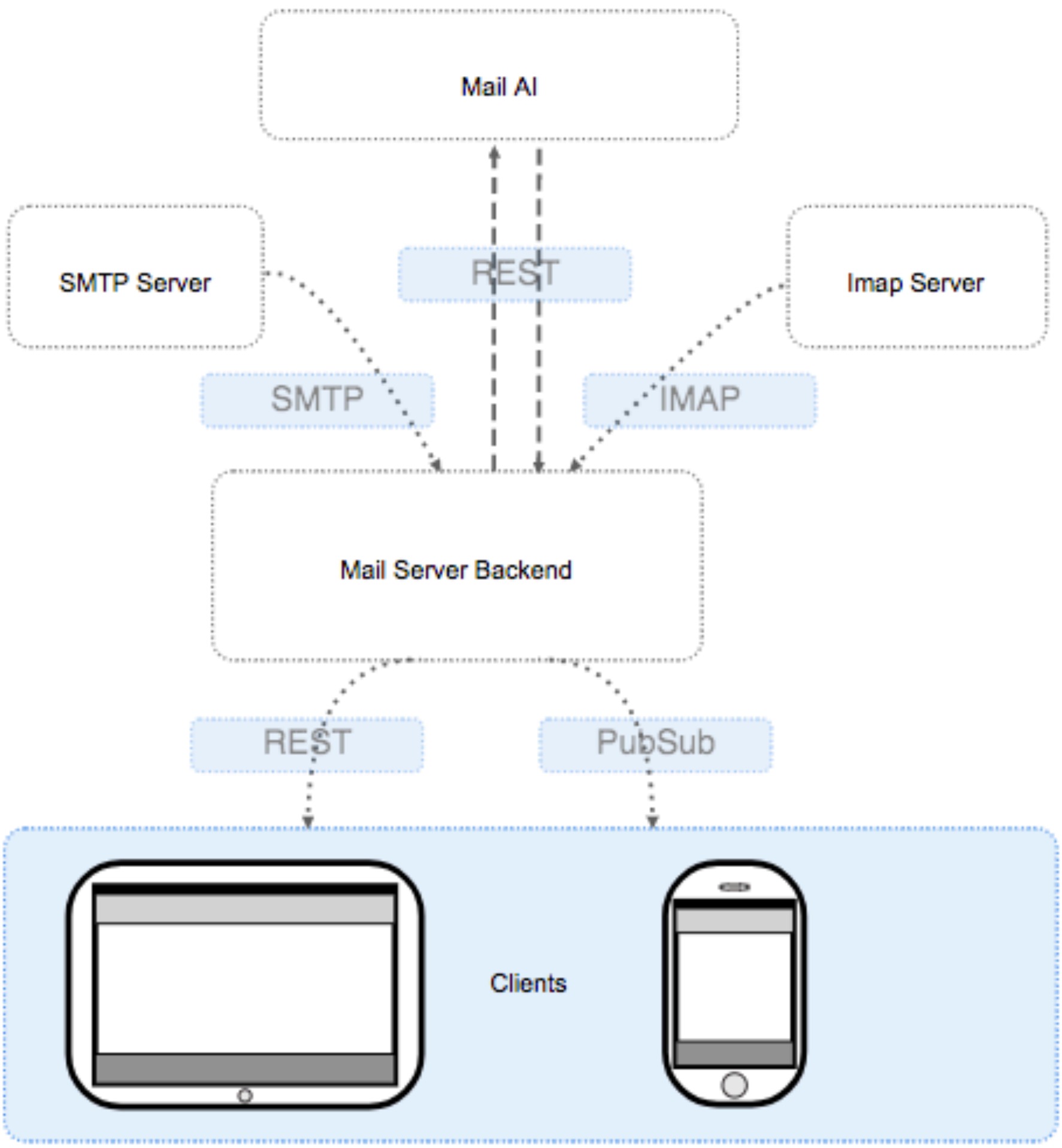
version 1.0.0

gabe@seedess.com

Technical Architecture for Native Mail App

Mail App Communication

Communication overview for each component



Mail App Architecture

version 1.0.0

gabe@seedess.com

Technical Architecture for Native Mail App

Example REST:

Request:

HTTP GET https://mail-ai.com/new-mail

```
{
  id: ‘unique-abc’,
  headers: [...],
  subject: “Test Email”,
  from: “user@example.com”,
  body: [...]
}
```

Response:

```
{
  id: 'unique-abc’,
  categorize: ['test', 'important']
  delete: false
}
```

Mail App Architecture

version 1.0.0

gabe@seedess.com

Technical Architecture for Native Mail App

PubSub Protocols

https://deepstreamhub.com/blog/an-overview-of-realtime-protocols/

MQTT

Highly considering MQTT as the pubsub transport protocol.
http://mqtt.org/faq

* The use of MQTT is because it's a binary pubsub transport predominantly used for low powred IoT devices and thus suitable for mobile apps. Eg: Facebook messgenger http://mqtt.org/2011/08/mqtt-used-by-facebook-messenger
* MQTT is supported by many message queues. Eg: https://www.rabbitmq.com/mqtt.html
* Standard protocol with many implementations https://github.com/mqtt/mqtt.github.io/wiki/Server%20support