Requirements:

Functional Requirements:

1. Should paste text and it should be stored and return a unique url to access it
2. Should upload text or paste
3. It should have a default expiration time and also a provision to mention the time
4. Users should optionally be able to pick a custom alias for their paste.
5. Length of the text that can be uploaded – Design limitation

Non-Functional Requirements:

1. Should be available always
2. Should be reliable, should not lose the text

Capacity Requirements:

Read: Write 🡪 5:1

So, if we have a 1Million pastes per day, 5M reads per day

10MB per paste

Scale:

1. 5M/24 \* 60 \* 60 = 58 Reads per second
2. 1M/ 24 \* 60 \*60 = 12 Writes per second

Storage:

1. 1M \* 10KB = 10GB per day
2. 1M \* 10KB \* 30 \* 12 \* 10 = 36TB for 10 years
3. 64^6 = 68.7 billion unique urls

API Routes and Methods

Create Paste:

1. Username
2. Text
3. Paste Name
4. User token if authenticated
5. Expiration time
6. Custom url
7. Return: short url

Get Paste:

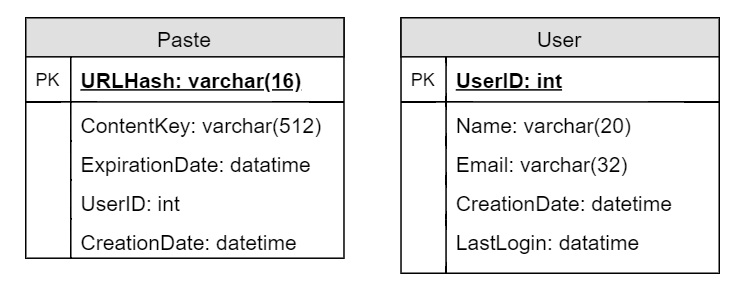
1. Short url
2. Returns: text stream

Delete:

1. Short Url

Define database models:

1. Key value pair, we can store in a document database like mongodb
2. We can store the meta data in a relational database and actual document in a NoSQL database



System Design:

Create:

1. We first generate a unique short url for this request using Short URL generator service (SUG)
2. SUG can have a database where it can store the prepopulated set of urls
3. App server can cache certain keys to speed up the process
4. Stores the paste text on an object database like S3
5. Stores the metadata in a relation table, with the reference to S3

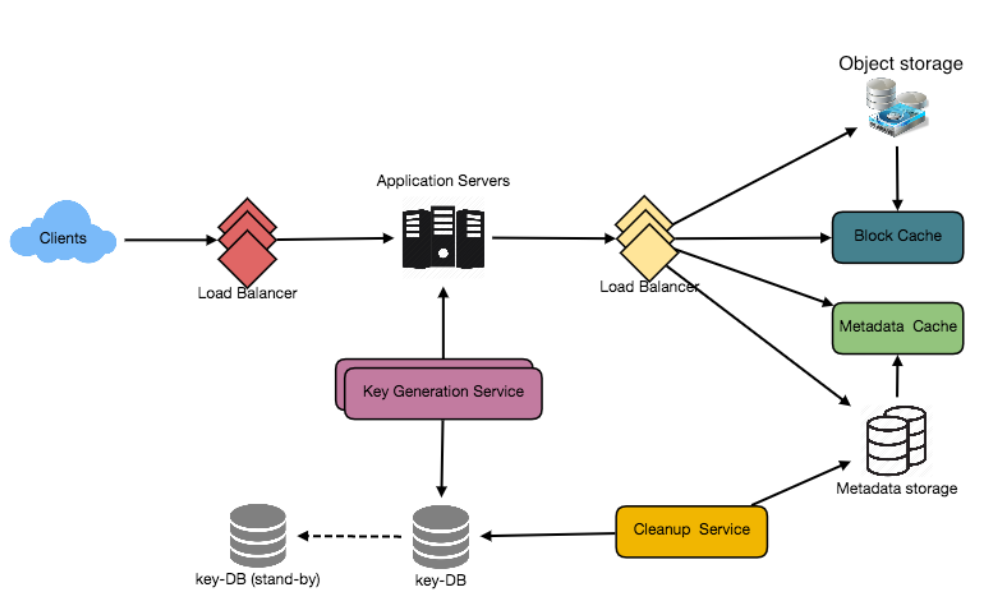
Single point of failure:

1. We can have multiple app servers which can be load balanced
2. If one of the servers dies the call can be routed to another app server
3. We can also monitor the metrics on the load balancer to circuit break the call and give it to a different server

Read:

1. Read by userid or unique key
2. If found returns the contents with 200 else 404

Block Diagram:



Caching:

1. We can cache 20% of the hot urls
2. We can use LRU cache, which evicts the least recently used value

Cleanup:

1. We can create a cleanup service to delete the references and metadata of the expired data

Security:

1. We can allow only valid users to create data more than certain amount
2. We can check the username and retrieve the paste if it is private