1. What's pastebin?
   User store plain text or images over the network, and generate unique URLs to access the uploaded data.
   Users would just need to pass the URL to let other users see it.

2. Requirements and goals of the system?

   Functional requirements:
   Users should be able to upload or paste their data and get a unique URL to access it.
   Users will only be able to upload text.
   Data and link will expire after a specific timespan automatically. Users should also be specify the expiration time.
   Users should optionally be able to pick a custom alias for their paste.

   Non-functional requirements:
   Read-> access URL
   Write -> store the text into the DB, so the system should be read-heavy.

   Also, the generated URL should not be predictable.
   System should be high reliable, data uploaded should not be lost.
   High available, because if service is down, users will not be able to access their pastes.
   Users should nearly immediately get their paste's URLs.

   分析 paste 的 URL 会被 access 多少次?

3. Some design considerations.
   Max size of paste ?=> can be 10MB.
   If support custom URL => reasonable to impose a size limit on custom URL.

4. Capacity estimation and constraints:

Assume 5:1 ratio. Between read and write.

Traffic estimates:
Every day : 1M write, 5M read.
Per second: 12 paste / s , 58 reads/s.
Storage estimate : size limit is assume 10MB every paste. Assume each paste has size of 10KB.
1M * 10 KB = 10GB / day
存十年的数据，就是 36TB。
1M paste/day => 3.6Billion paste in 10 years= > if base64encoding => 6 digits enough!!!
存 3.6Billion 的 key =>总的大小 3.6Billiion * 6 => 22 GB.

十年的数据， 36TB 的 paste + 22GB 的 key(base 64 encoding)

Bandwidth estimates:
12 paste / s , 58 reads/s.
写：120KB/s,
读：0.6MB/s

Memory estimate:
Cache => follow 80-20 rule,
一天 5M 读， 20% 就是 1M， 1M * 10B = 10GB。

5. System API

   对于用户，paste 了之后需要得到对应 URL 发给别人

   String addPaste(api_dev_key, paste_data, custom_url = None, user_name = None, paste_name = None, expire_data = None)

   Return URL

   对于后端，需要根据 key 拿到对应 paste，给别人看。

   getPaste(api_dev_key, api_paste_key).

6. Database design

   存储 billion 级别的数据

   Metadata 很小，paste object 中等大小。

   记录之间没有关系。

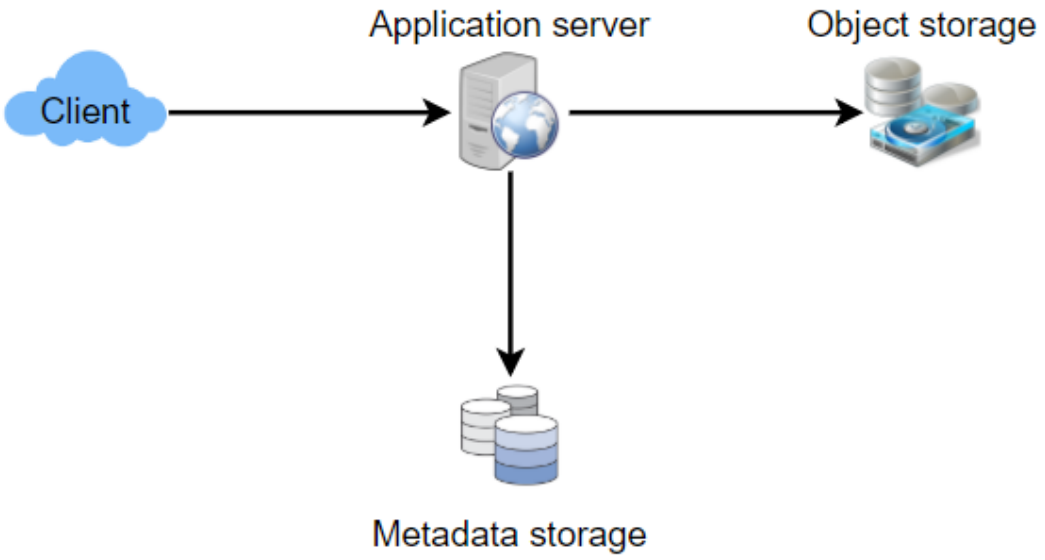
   Read-heavy。

Paste	
PK	**URLHash: varchar(16)**
	ContentKey: varchar(512)
	ExpirationDate: datatime
	UserID: int
	CreationDate: datetime

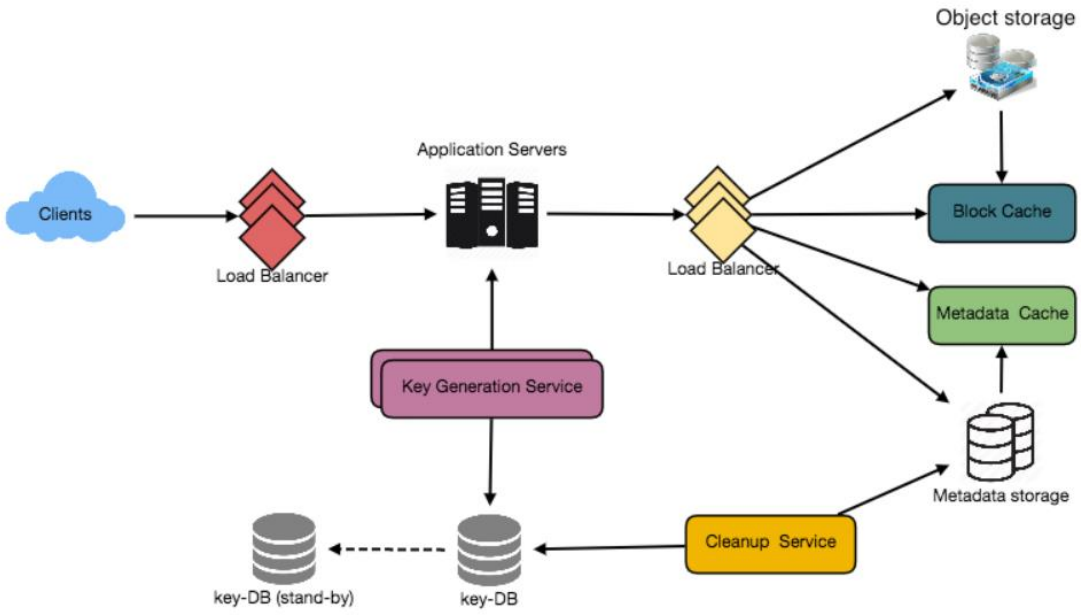
User	
PK	**UserID: int**
	Name: varchar(20)
	Email: varchar(32)
	CreationDate: datetime
	LastLogin: datatime

> Here, 'URlHash' is the URL equivalent of the TinyURL and 'ContentKey' is a reference to an external object storing the contents of the paste; we'll discuss the external storage of the paste contents later in the chapter.

7. High level design



8. Component design



Detailed component design for Pastebin

Application layer：
Receive request=》 generate 6 digits key , store the key and the content into the DB, key 可能重复，那就重新生成。
另一种方法：run KGS , KGS 有两张表，一张可用的 key， 一张用过的 key。
KGS single point failure 怎么办？=》 replica of KGS.
并且 server cache 一些 key, 这样加快获取速度。 不怕浪费 key 数量，因为 6 位 digits 足够。

数据库层：

Metadata 层 都可以
和 object 层：存到 Amazon S3 里。