Я люблю свою работу, посещу ее в субботу.

Для начала внесем немного смысла в задачу об урлах.

Задача, видение и основной концепт (предположительная, придуманная вводная).

Люди обмениваются ссылками в мессенджерах, твиттерах и лентах соц. сетей, однако не все эти необработанные ссылки удобны. Место на экране ограничено и не всегда пользователю нужно знать весь url, чтобы подгрузить ресурс. При этом можно считать, что переписка и лента новостей, хотя и хранятся бесконечно долго, со временем устаревают. Поэтому умные люди предложили сервис-словарь, который укорачивает url, и затем восстанавливает исходный по короткому, при этом не храня страницы словаря вечно. Это экономит место на экране, ресурсы соц. сетей, нервы людей и не требует значительных затрат (вычислительных, дисковых и человеческих).

Подход к решению.

Будем решать задачу методом последовательных прояснений аспектов на моделях, реализации прототипов, выбора технологий, тестирования, кодирования и деплоя на AWS. (на google бесплатный аккаунт подошел к концу, а на изучение Azure нет ни времени ни сил). Постараемся справиться за неделю, если не получится - возьмем еще неделю сверху.

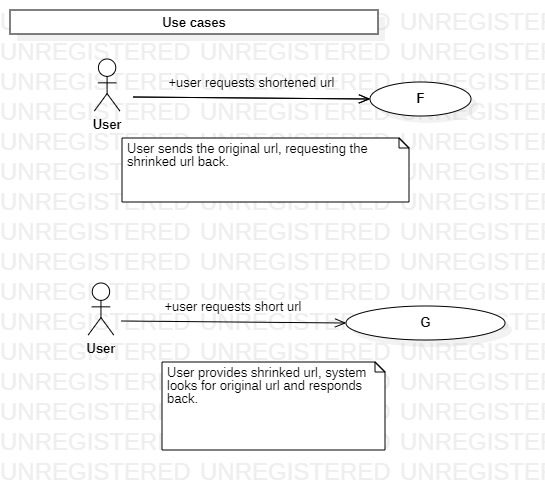
Начнем с математической модели.

Задача распадается на два процесса:

1. Преобразование строки подлиннее в строку покороче.

2. Обратное преобразование строки покороче в исходную строку.

Изобразим случаи использования на диаграмме.



Опишем эти два процесса языком теории множеств.

Дано:

1) входной url - в формате http[s]:\\[protocol]:<имя сайта>/[resource].

2) формат выходного урла - строка в формате http:\\www.<our url>\<строка>.

3) выходной урл должен умещаться в строку не более K символов длиной. Будем считать,

Что K – степень двойки. K = 2^l.

Решение:

Пусть si (входной урл) - строка из множества C8 (бесконечное множество строк произвольной длины).

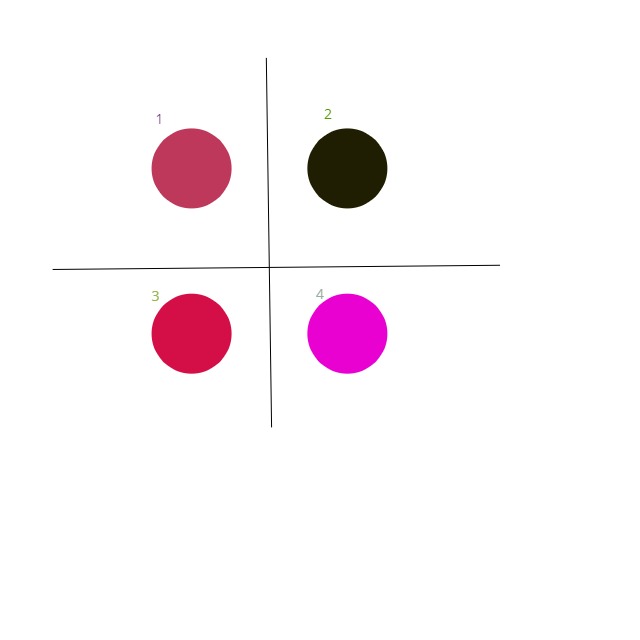
Пусть so (выходной урл) принадлежит множесту CK ( конечно множество строк длины не больше K).

Пусть F: C8 -> CK - отображение множеств, функция, укорачивающее url.

Пусть G: CK -> C8 - обратное отображение, функция, восстанавливающая по укороченному url оригинальный.

Поскольку мощности множеств C8 и CK не совпадают, F необратима в общем случае. По короткому урлу можно восстановить произвольный набор прообразов. Какой же из них мы должны выбрать, чтобы решить нашу инженерную задачу хотя бы как-то? Можем брать самый последний. Что означает “самый последний”? Самый последний вычисленный. Будем для простоты считать, что вероятности появления слов из C8 равны, тогда можем выбирать соответствие между CK и C8 случайно. Тогда F : C8->CK = random(CK). Как же тогда построить G?

Изобразим множество CK.



Множество конечно и все его элементы можно пронумеровать алгеброй A.. Назовем Ri – множество состояний множества соответствий C8 и A, множества C8 x A, где R0 - пустое множество.

Ai = F(sin). X(R, A, sin) = {r из R: r.a = a, r.sin = sin}, R i+1 = Ri - X(Ri, F(sin), sin) + (sin, F(sin))

Теперь чтобы достроить F и G достаточно закодировать и декодировать число A в строку sout длины K. Как это лучше сделать? Будем считать что строка sout кодируется символами английского алфавита (26 символов) и 6 арабскими цифрами. Итого – 32 символа. Тогда алгоритм преобразования используем такой:

Codec: S = 0, s = A mod 32, S = S плюс справа T(s), A = A div 32 – s, где T – таблица соответствий между числами от 0 до 31 и выбранными 32 символами.

Decodec: тот же алгоритм только в обратную сторону.

Итого получаем несколько функций:

1. F: C8 -> A
2. Codec : A -> CK
3. Decodec : CK -> A
4. G: A -> C8
5. D: Ri -> R i+1

Проведем амортизационный анализ. Пусть множество R реализовано через массив. Тогда добавление элемента и вычитание предыдущего сводится к операции присваивания: R[i] = e.

Codec O(K)

Decodec O(K)

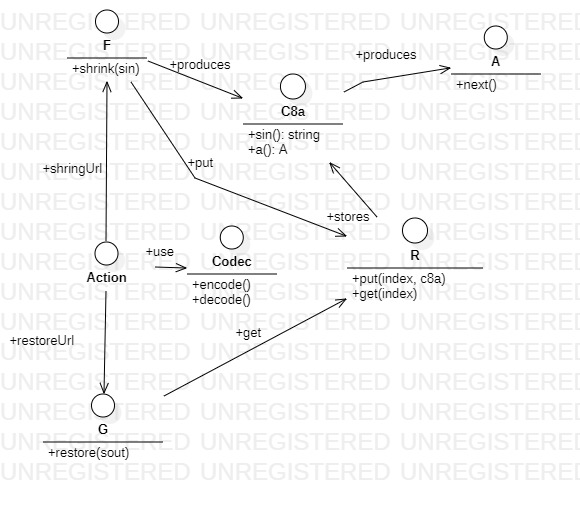
F O(1)

G O(1)

D O(1)

Итого: Укорачиваем ссылки вычисляя случайное число от 0 до K, строим словарь на основе массива, преобразуем номер элемента в массиве в строку и обратно преобразованием формата записи числа.

Изобразим сущности на диаграмме классов.

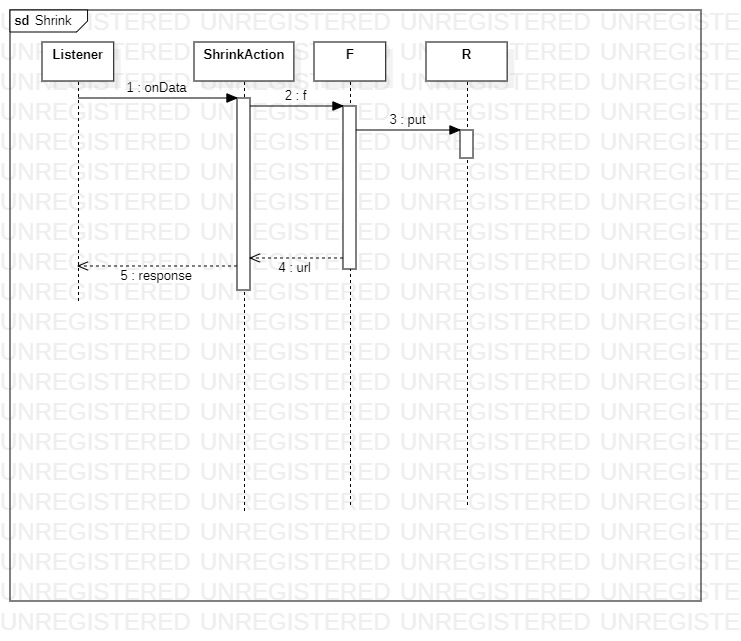


Для проверки модели разработаем прототип на самом близком автору по опыту языку Java.

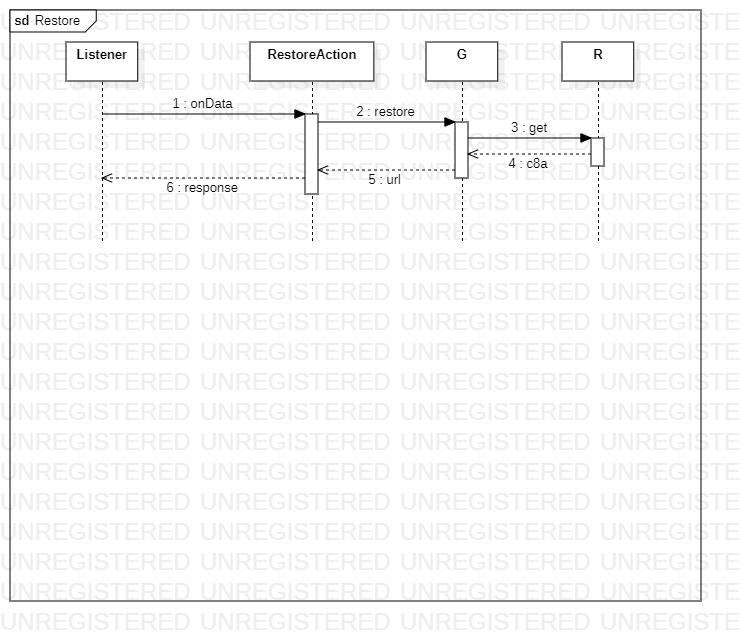
Синхронизация асинхронных событий.

Поскольку запросы на конвертацию вливаются в обе стороны асинхронно, стоит распределить их обработку по нескольким потокам. Операция присвоения ссылки и считывания ссылки на множестве R – атомарны, поэтому операции записи и чтения не будут мешать друг другу.

Изобразим потоки записи-чтения на графике.



И



Построим приложение упаковки и распаковки урлов на базе http сервера. Для этого надо подобрать платформу разработки приложения. Нам нужна шустрая платформа с простым http сервером и поддержкой ооп парадигмы.

|  |  |  |  |
| --- | --- | --- | --- |
|  | C++ | Java | NodeJs |
| Простота wbsrv | +- | + | + |
| парадигма | все | все | все |
| Скорость работы | +- | +- | + |

Напишем сервер на NodeJS. В качестве интерфейса выберем два rest эндпойнта. Put на <http://www.server.com/shrink> с урлом в теле запроса возвращает короткий url в ответе. и get на [http://www.server.com/<shorturl](http://www.server.com/%3cshorturl)> возвращает исходный url.

Все.