Replace the build-in Server-Side Rendered pages with Redis

If you use the cluster mode in the PM2 your build-in cache is created separately for each cluster instance. In the situation that you use, for example, 8 nodes of Node.JS in PM2, 8 consecutive requests for a home page can be added to the cache main page 8 times. This kind of cache is inefficient because of:

- the cache is kept in memory so you use 8 times more memory
- you can't rely on that cache because the traffic split in PM2 is unpredictable and you can 8 times hit node 1, or hit 8 different nodes.

To avoid this situation we recommend modifying `server.ts` to use Redis cache instead of the build-in `lru-cache`.

To use Redis cache you need to install Redis on your server and modify access to cache.

```
async function saveToCache(req, page: any) {

  if (!isUserAuthenticated(req)) {
    const key = getCacheKey(req);
    // Avoid caching "/reload/[random]" paths (these are hard refreshes after logout)
    if (key.startsWith('/reload')) { return; }

    // If bot cache is enabled, save it to that cache if it doesn't exist or is expired
    // (NOTE: has() will return false if page is expired in cache)

    if (isRedisEnabled()) {
      const expireTime = getCacheExpireTime(req, environment.cache.serverSide.redis.timeToLive);
      await redisPCG.SET(key, page, {
        PX: expireTime
      });
      if (environment.cache.serverSide.debug) { console.log(`REDIS CACHE SAVE FOR ${key} in anonymous cache.`); }
    } else  {
      if (botCacheEnabled() && !botCache.has(key)) {
        botCache.set(key, page);
        if (environment.cache.serverSide.debug) { console.log(`CACHE SAVE FOR ${key} in bot cache.`); }
      }

      // If anonymous cache is enabled, save it to that cache if it doesn't exist or is expired
      if (anonymousCacheEnabled() && !anonymousCache.has(key)) {
        anonymousCache.set(key, page);
        if (environment.cache.serverSide.debug) { console.log(`CACHE SAVE FOR ${key} in anonymous cache.`); }
      }
    }
  }
}
```

We recommend creating a cache key with language prefix: `${lang}:${req.url}` to avoid caching the wrong language and reduce the flickering issue on the preloaded page.

```
function getCacheKey(req): string {
  // NOTE: this will return the URL path *without* any baseUrl
  const lang = req.cookies[LANG_COOKIE] || req.acceptsLanguages(...environment.languages.map(el => el.code)) ||
environment.defaultLanguage || 'pl';

  return `${lang}:${req.url}`;
}
```

We also recommend splitting cache times to specific pages to make a cache more efficient, for example:

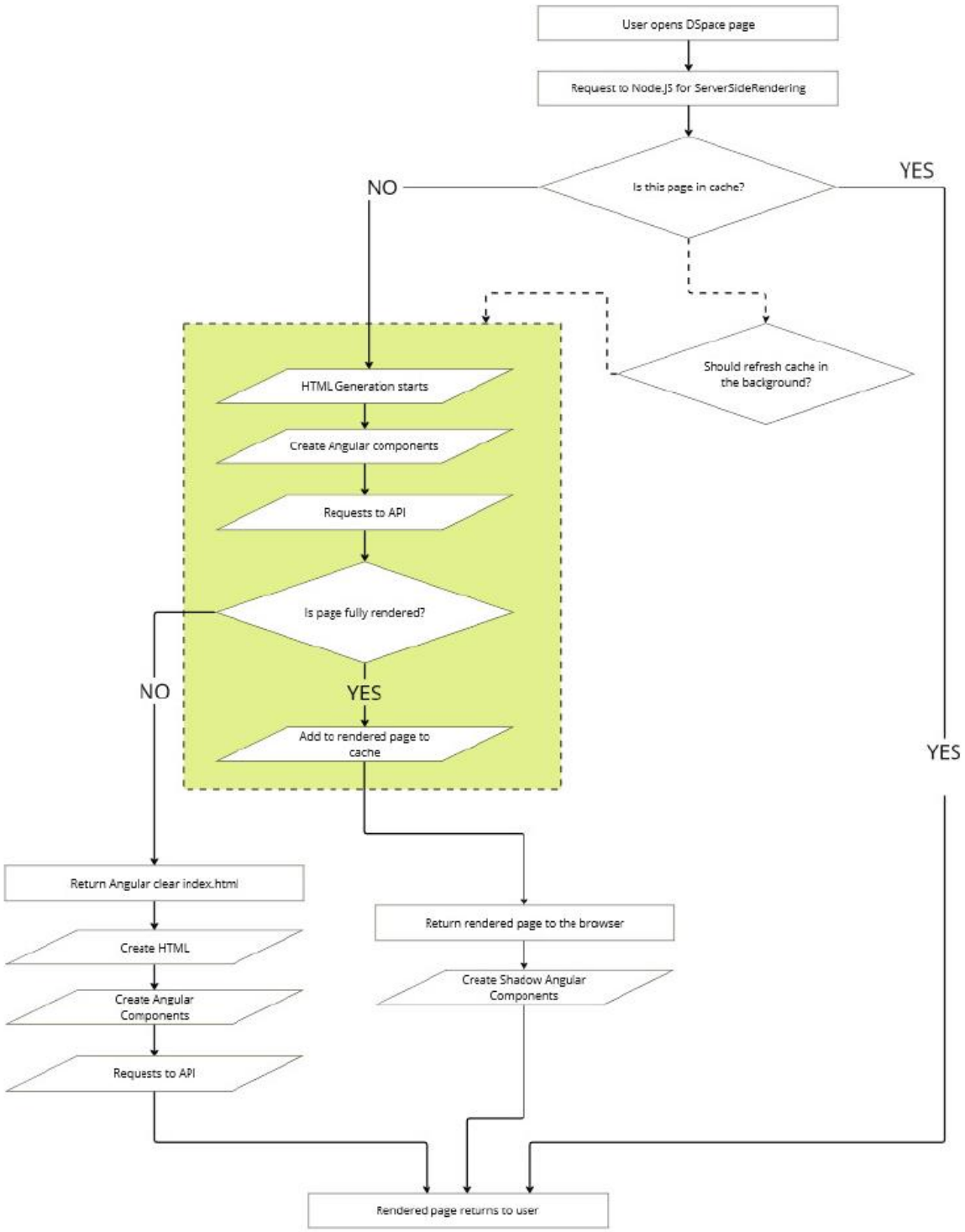
- home page
- publication - (short cache)
- bitstream - (short cache)
- search
- all others pages

```
function getCacheExpireTime(req, redisTime: ReidsTimeConfig): number {
  if ((req.url === '/' || req.url === '/home') && redisTime.homePage) {
    return redisTime.homePage;
  } else if ((req.url.includes('/entities/publication/') || req.url.includes('/handle/item/')) && redisTime.publication) {
    return redisTime.publication;
  } else if ((req.url.includes('/bitstream/handle/')) && redisTime.bitstream) {
    return redisTime.bitstream;
  } else if ((req.url === '/search' || req.url.includes('/search?')) && redisTime.search) {
    return redisTime.search;
  }

  return redisTime.default;
}
```

And configure that cache for your needs.

If you look for another optimization, please consider refreshing the cache in the background. In this approach, you can return the page from the cache, and refresh a cache version of the page in the background. That gives you a relatively fresh page in the cache system, and you can provide a quick response to the customer.


User opens DSpace page
Request to Node.JS for ServerSideRendering
Is this page in cache?
NO
YES
Should refresh cache in the background?
HTML Generation starts
Create Angular components
Requests to API
Is page fully rendered?
NO
YES
Add to rendered page to cache
YES
Return Angular clear index.html
Create HTML
Create Angular Components
Requests to API
Return rendered page to the browser
Create Shadow Angular Components
Rendered page returns to user