With the introduction of Google's new ranking system, Web Vitals, Google aims to measure real-world experience metrics like how fast does the page load, how fast is it interactive and how fast it is stable.

Core web vitals

Core Web Vitals are the subset of Web Vitals that apply to all web pages, should be measured by all site owners, and will be surfaced across all Google tools. Each of the Core Web Vitals represents a distinct facet of the user experience. Currently the metrics that make up Core Web Vitals includes:

- Largest Contentful Paint (LCP): measures *loading* performance
- First Input Delay (FID): measures *interactivity*
- Cumulative Layout Shift (CLS): measures *visual stability*.

Low quality image placeholders(LQIP)

The concept of low quality image placeholders is quite simple, load and show a low quality version of the image that is to be shown and replace it with the full quality one once it finishes loading. With this technique the user can see a low resolution image instead of a blank space, optimising the user experience as well as improving the aforementioned Core Web Vitals. Using LQIP can help with Web Vitals as a low quality image placeholder can reserve the space the full resolution picture will occupy thus improving the Cumulative Layout Shift metric and improving the user experience as text is not shifting around.

Video showing comulative shift (https://i.gyazo.com/d2d70ed6a8a3e78de01669c357233e7f.mp4)

With the LQIP technique we can achieve a better result as the low quality image placeholder gives the user a new experience.

Video showing lqip (https://i.gyazo.com/ba07e438ee48a8691ee33d3501f2efb0.mp4)

LQIP com Next.js

In order to incorporate LQIP in Next.js and in order to generate the low resolution images we used next-optimized-image, a plugin that allows to optimize images. it is of note that this plugin requires the instalation of other dependencies that are needed, in this case lqip-loader. Using this plugin we can create low quality versions of images: `{require('myImage?lqip")}` The creation of the low quality image is done during build time in the server so this low quality image has no extra processing time on the user.

With the low quality image placeholder the only thing left is to display the low quality image and switch it for the high resolution one once it finishes loading. There are several ways of doing this, using absolute positions and placing the images atop eachother, the low quality one will be visible until the high resolution loads. The other solution is to use a react component that changes the image once it loads. One example is react-image-fallback. Using this package we can easily implement this technique:

```
<ReactImageFallback src={require('../public/images/myImage.jpeg')} fallbackImage=
{require('falbackImage.jpeg')} initialImage={<img src={require('myImage.jpeg?lqip')} style=
{{width:"100%"}}></img>} alt="ImageAlt" className="my-image" />
```

This tutorial was composed taking into account the next-optimized-image, however Next.js rolled out it's own image optimization.