that calls other functions. If I developed a function in Zimki, whenever that function was called then I could see exactly how much it had cost me. I was charged on the network, storage and compute resources used by that function. This was quite a revelation. It changed behaviour significantly because suddenly in the sea of code that is my application, I could find individual functions that disproportionately cost me more.

As far as I know this pricing per function was unparalleled in the world of IT in 2006 and we didn't see an equivalent pricing granularity until AWS Lambda was launched in 2014. Now, obviously I was also the provider of Zimki and behind the scenes there was a complex array of basket of goods concepts and all manner of financial instruments to be able to provide those cost figures. But this was abstracted from the developer. All they saw was a cost every time their function ran no matter how much it scaled. There was no capital investment and this turned the operational cost of an application into a manageable variable.

Flow

I'm now going to combine the ideas of worth based (outcome) development and pricing granularity to introduce an idea known as flow. To do this, we're going to revisit the LFP project but this time with a map and the knowledge of what a utility platform can bring. Back when we were actually working on the LFP project, I hadn't developed the mapping concept fully and Zimki wasn't released. Hence