I am a father of two lovely kids, a three-years old wildcat girl and a two-months old prince. I love outdoor sports like paintball, trekking and outdoor running. I spend my weekends on my lovely village on Crete and try to travel all over the world (when it is possible). On my free time (if there is any), i try to live new experiences or stay at home playing video games or reading books.

I am a software engineer that loves to be part of challenging and interesting large scale software solutions and like to follow state-of-the-art technologies, best practices and design patterns. Big fun of microservices architecture and of its benefits like loosely coupled services, independent deployables and high maintenance. I love to communicate with people and clients, transform the business requirements into software specifications and finding solution on other’s problems. My short/long term expectation is to become a solution architect.

I have studied Computer Science Department at University of Crete and completed master studies on ubiquitous computing at Hellenic Open University. My master thesis is based on ubiquitous systems simplifying summarization video process using context. My professional experience with software development started as a front end developer focusing on android native development and Angular web framework. At short time, my duties increased with REST API development using .NET C# and undertook a team of four as a team leader following Agile Methodology. Then I moved on microservices architecture with Spring Boot framework and Spring Cloud Netflix OS implementing the middleware of Pancreta Bank. I took place on designing, the architecture and drove the implementation of the open banking solution of the PSD2 regulation. Currently, I am the team leader of the web services team of the bank. I have taken part in solution designs, components integrations and solution presentations on clients.

From my experience, maintainability is a big issue on software development. As a big fun of clean code I am trying to create independent software components in order to be easy maintained and easy to change the integration between components. Creating concrete layers in the source code helps us to keep a clean architecture and separation of concerns. So, the reason I like the article that follows is that the suggested principles agrees with a better code design in order to keep it clean. Suggestions are write code that is easy to delete, layer the code, split the code isolating the hard parts from the easy ones and chasing “one rabbit at a time”. It is a mainstream article but remains topical.

How-to-write-disposable-code-in-large-systems

https://programmingisterrible.com/post/139222674273/how-to-write-disposable-code-in-large-systems