Christoffer Nilsson

I am the kind of person who really loves my job. I spend a lot of my free time programming applications och puzzles just for fun and reading blogs. A part from my job I like to spend time with my family, my girlfriend who is a recently graduated teacher and our 3 year old son. I also follow a lot of sports for example golf, which I also (try to) play myself, hockey and some football.

It is important to me to have a good relationship with the people I work with. I like to collaborate daily with my colleagues. In teams where I work for longer time periods I tend to end up in a role where I handle much of the communication with stakeholders, architects and users. It is a role I like to be in since it is a good compromise between writing code and having a higher order of understanding.

The project that I am most proud of having been a part of is the one at Thomas Cook. I worked in a very inspiring team and was trusted with a lot of responsibility even though I was the only consultant on the team. The journey we did, from a monolithic .Net application to more of a "microservice" architecture with modern tech stack and continuous delivery, has shown me what is possible to achieve if you set your mind to it.

Experience

Global Connect/IP-Only - Digitization

2018-12-01 - Ongoing

Global Connect/IP-Only builds and operates one of Sweden's largest and neutral fiber networks. Their business model is to empower digitization by owning, developing and offering the best infrastructure and low level services for the digital society.

The focus of the project was to develop both customer facing and internal it-systems. Before the project started, different parts of the company worked mostly in their own defined processes and tool. The tools could be bought systems but mostly excel and email was used to handle sales and deliveries. The aim was to to document all core processes through the different parts of the company and set up integrations and build new systems to make the work faster, easier and less error prone.

System developer .NET
I started the project as a backend .Net developer working with business sales support. IP-Only had just merged with another company called DGC and had inherited some .Net systems for searching connectivity options for addresses and some tools used for deliveries in the Network acquired from DGC. The main issues with this project was that these systems where in bad shape and lacked documentation. Quickly after starting the project I lifted the issues and proposed building new requirements in the IP-Only stack and develop a new module in the legacy system to benefit from everything done in the IP-Only tech stack. This finally led to so much of the functionality already being in the IP-Only stack that we decided to build the UI and missing logic there as well and deprecate the old system. The value gained from this was a more reliable system and now the entire IT development department have the competence needed to fix bugs and monitor the system instead of having a single person responsible for all of this like before.

Full stack developer Java/AngularJS
As the project progressed the role became more and more a Java/AngularJS fullstack developer continuing

to build a customer portal. The goal of this was to have enterprise customers log in by themself and search for network options for their locations. This new customer facing feature was built upon the engine for search described in the previous role. It has a angular frontend for managing addresses, products and so on. This system was integrated with a new self developed product catalog and a back-office UI to manage all products. The integration led to a faster way for customers to get quotes, often automatic replies, on buying new products.

Full stack developer Java/AngularJS
When the first version of the customer portal was released I felt that I wanted to try something new and as a result the Client agreed to move me from the sales support team to the operational support team. There I started working in a developing team with a lot of work dependent on just one team member. I worked close together with the project manager to develop I working development process and implement most of the scrum model. This new role had a few different goals, set up a back-office application for reserving and finding network identifiers of different forms. Providing data for a new Service Now integration as well as setting up API:s to fetch data from Service now API:s. The other prioritized goal was unifying the stack used to monitor the 2 recently merged networks. This meant merging data sources, set up new services to handle both networks and so on. What I take away from this role os more of the soft skills I developed. Coaching new team-members. Introducing to stake holders what a sprint demo is and doing a lot of them and so on.

Java | Spring Boot | DataGrip | Gitlab | IntelliJ | RabbitMQ | Elasticsearch | Kibana | MS SQL | .Net | C# | AngularJS | Archi | ArchiMate | MariaDB | Nginx | Docker Swarm | Grafana | Octopus | Teamcity

Thomas Cook NE - Responsive website redesign

2015-01-01 - Ongoing

Thomas Cook Group is one of the world's largest organizers of charter trips. The group includes i.a. the tour operators Ving, Spies, Tjäreborg, Globetrotter, the airline Thomas Cook Airlines Scandinavia and the hotel chain Sunwing Family Resorts. Thomas Cook Group has 22,000 employees in 15 countries, 20 million customers and a turnover of almost SEK 90 billion.

Thomas Cook Northern Europe wanted reinforcement for their project group that worked on redesigning Ving.se, Ving.no, Spies and Tjäreborg to make the website more mobile-friendly. They wanted to make it responsive, more user-friendly and easier to understand for the convenience of their users.

Full stack developer
I was originally intended to be a back-end .Net developer but I was put in a team that had just built a Node.JS API with a AngularJS front-end. Before starting their next project I was tasked with comparing the new angular version and ReactJS. To do this I tried building a small application in both frameworks and talking to my Netlight colleges at SVT who had used React for a while. I proposed to switch to React which we did and it worked out great. I still feel that I like working in React better than Angular even with the newer versions. We then continued to build smaller services for their webpage using Node/Express API:s and React front-end apps.

Workshop leader
When we had started to build more of the React applications for our external web site. We started to notice the need for a component library and common styles to use in our applications. I started pushing

for us taking after e.g. Airbnb and make a style guide with components that can be reused and to publish all of these both as a npm package and as a webpage with examples and code snippets. Me and a college decided to boot strap this with a workshop for all teams working on the web. We set up a basic repository, had a short introduction, divided up groups with people from different teams and assigned them with building one component each and also an example page where they could be used. They then spread out and started working from their own ideas and we as workshop leaders moved around to each groups and checked if anyone needed help or input from us. The package was well received in all teams and every developer was included in an early stage in creating the package and components. It was important to us that all web developers where comfortable making changes and adding new components in the library since it would otherwise become more of an obstacle than a helpful tool. When I left thomas cook this project was still going strong and had a large impact on the pace new applications could be developed and still keep a coherent ui.

Full stack developer GraphQL
I was trusted with more and more responsibilities at Thomas Cook and when a colleague and me started to discuss the potential of setting up a GraphQL API for the web we got an ok to test it out on a small application. The reason to try it was that we had seen that there was much time going in to writing API:s for similar API:s again and again across the company e.g. text data from our CMS and facts for hotels. We also had a need to figure out how to handle caching for our API:s to make our API calls run faster when possible.

We started of as a team of 3 who worked on this and we started to set up our own framework for fetching data from different sources such as cache, api:s and databases. This was not available out of the box in Apollo server at the time this was built, which it to a large portion is today. Our first version of caching was set up using a MS SQL in memory database as a key value store. We wanted to use Redis but there was some hosting issues needing sorting out before hand. We therefore used our SQL database in using the same kind of methods available on Redis to make it easier to swap them out later. Towards the end of the project we also got access to a Redis instance and could make the switch.

The new API made using data provided by services built in other teams easier as well as providing us with useful data on response times for different types of data. We also tried setting up Apollo client directly on top of React but found it was not easy to build the same functionality that we were used to build using redux and redux-saga. So we ended up just using it in out data access layer at the client and keeping our shared application data in redux as it was previously done.

Elasticsearch Express Github Jenkins Kibana Node React Redux Redux-saga Apollo-client GraphQL .Net C# MS SQL RabbitMQ Redis TFS

Belivia - E-health

2014-11-01 - Ongoing

Belivia is a company working on risk assessment for insurance companies. Part of their product is a electronic health declaration. The health declaration can be tailored to fit the need for different insurance policies by uploading scripts using their own format and the dynamically render the questionnaire to the user.

Upon selling their solution for health declarations to a large swedish insurance company they had to increase security on their application. Data had to be encrypted on disk and participants had to be able to sign their forms using BankId. They therefore decided to rewrite their application in .Net instead of the current PHP.

System developer .Net
I was one of two developers on the project and worked on all parts of the new application. What I learned during this short project was how to integrate with the BankId API, how to handle encryption with rotating keys in the data access layer and how to go through a external security audit.

.Net Entity Framework MVC javascript TFS BankId Azure

Education

Royal institute of technology - Master of science program for Computer science

2008-09-01 - 2014-09-01

Technical University of Munich - Exchange semester, computer science program

2013-09-01 - 2013-12-20

Amf1 Berga - Military service, combat boat driver

2009-09-01 - 2010-06-20