Interview transcription - 2021-03-31- 13:00 over Zoom (remote)
Speaker 0 = interviewer
Speaker 1 = Interviewee

- Speaker 0 00:00:01 Okay, so we are recording. Uh, so let's see. Well, let's start off with, uh, some, uh, demographic questions. So in terms of you, uh, what is your previous experience, uh, related to it? And, uh, how many years have you worked at your company for example, or in total? Within the it sector?
- Speaker 1 00:00:30 Yeah. Hello. I can first start to say, my name is [NAME OF INTERVIEWEE] and I'm a web developer and I have worked with it since I was like 15. Uh, so I have approximately 18, 19 years of development experience, mainly with PHP, backend language, uh, database systems, infrastructure servers, uh, and some fronting technologies like react, angular view, et cetera. So at my current job right now, I work at, uh,[COMPANY NAME] and we sell a booking system to, um, ski destination in [CUSTOMER LOCATION]. Uh, and we take care of reservations for skiing, ski pass, uh, commendations packages, tickets, um, and also take care of the cleaning process with, uh, cleaning up, uh, on each customer sites. So I have been working here for one and a half year. So I'll say back-end and front-end web developer.
- Speaker 0 00:01:50 Uh, if we dive a little bit deeper into the topic, which we will discuss, uh, the topic is something we call user story splitting the, some parts surrounding that process, uh, or that, uh, way of working. Uh, so when we mentioned user stories, we don't necessarily think of it as a artifact with a, uh, syntax, uh, such as, as a role. I want to a goal in order to have some benefits. So that's a common, uh, uh, common way of writing it in natural language to kind of facilitate and describe the requirements necessary. So in terms of this conversation, user story could also be, uh, it could, it could be an Epic, so something bigger, it could also be a feature. It could also be a small task. So the interesting thing which we are diving is the way of splitting something that's big into smaller pieces and how that is done technically and practically, but also the purposes behind it and the impact, uh, of the, uh, of that way of working. So in terms of trying to paint a picture, uh, of the way that you structured these artifacts, is there a, can you describe how the hierarchy within your, either your team or your company in terms of user stories and work items, how that hierarchy looks,
- Speaker 1 00:03:44 Uh, right now for [COMPANY NAME], uh, we have recently started to restructure the whole company, how we work with, uh, development tasks and bugs and features and everything within both teams. Uh, so the way we work right now is like we have a support team and the support team reports, bugs and feature requests to developers. Uh, and then if we receive feature requests that we want to use, then we saved them, um, to the next meeting where we can see in a big column for sprint meetings every two weeks. Um, for example, if we have promised to build a feature for a customer like bike rentals, then it's very suitable now to have bike rental as an Epic in JIRA.
- Speaker 1 00:04:43 And then we want the separate meeting just for the bike rental projects. So we can go through all the steps to separate the big project into smaller tasks. Um, we don't necessarily create real user stories. Uh, like, like you said, that, um, I'm a manager or a customer and I want to do this. And my benefit should be that it's more like, um, our customer just wants to be able to do one

function and then we try to create smaller tasks for that. Uh, so the bike rental system is actually one of the first bigger projects that we have, uh, started with during our new structure. Uh, and that was recently finished last week. So now we are testing it and we also have one more project, uh, to restructure and rebuild our checkout summary page. Um, so it's a large, large project to redesign one of the pages before the customer goes to checkout.

- Speaker 0 00:06:09 So when you said that you recently changed this way of working, do you have any insight or, uh, reasons for that change?
- Speaker 1 00:06:21 Oh, yes. Um, before we can say that almost all the years since this company started, like the, uh, the, the product was created in 1995 or 1996. And for many years there were so few people, maybe one developer and one or two for support and that's it. So they only had one guy working. So they, they haven't really needed a structure to work if they only was one person. Um, but now that was like 25 years ago. So recently we got a new boss and, uh, we, they wanted some features and the customers asked for features, uh, and we still didn't have a really good structure. So it was, it was chaotic. Uh, we didn't have the structure and a developer to not focus on one task until someone else thought that their task was more important than the one currently working on. So we were like switching tasks all the time, so we didn't finish what we started.
- Speaker 1 00:07:46 That's, uh, we could end up with like 10 started bugs or features that were never finished. So, it was also a lot of stress because this happened during the high season period from October to April, where people use the system a lot everyday on the destinations, like [CUSTOMER LOCATIONS]. Then they use the system every day. So they find bugs and they just need them fixed because they need to save time when they have a queue of customers. So it's a combination of stress and a lack of structure that has made it very hard to focus. But now when we have a new people in the team and new leaders, a new project team leaders, now we have decided to use the new structure to have like a sprint meetings every two weeks. And then we try to find, all right, which task do we think that we can finish in two weeks, one or many, for example, last period we have around 30, 30 or 32 projects or minor bugs or features for the web project. That's the reason why it was, it needed a restructure to be able to work maybe one or two months ago in January.
- Speaker 0 00:09:34 So you mentioned the, uh, a bike rental system, that's an Epic. So I'm guessing a big, uh, a big thing that needed to be done when you started to work with this. How did you, uh, how did you do the user story split thing of this Epic or the screening feature?
- Speaker 1 00:09:58 Yeah, first we, excuse me. Um, we decided to just split it in like organization parks. For example, we have three systems, so we need three teams just to build one feature and twin. We have the, [COMPANY NAME], which develops the booking system in Delfi. It's a windows application. They have to build a support in the database. They have to build an API. Uh, so the web products, it can speak to this API and then we have to implement a caching layer, uh, and also abstract away the API in [COMPANY NAME]. So we can have like a web backend, a separate web backend with a separate API. And then we have the third product team, which works with front-end development, building a client that can speak to the web server that sits in between of the booking system and the front end system. So we, we split the Epic, we kept one Epic, but then we S we split it so that we had three

sections, one for each team. So for example, my task was to build a back part. Uh, we had, uh, we hire a consultant to do the front-end part, and then a booking system part. We also had one employee that worked with us before, as a consultant in house. So that was the primary way of finding functions and features for each team. Okay.

- Speaker 0 00:12:01 Would you say, is it safe to assume that this is a, um, the way that it usually works this approach?
- Speaker 1 00:12:12 I would think so, make it, yeah. I think it's very logical to, uh, to split a very large product, to make it logical for each individual team. So for example, uh, the front-end team can do their tasks and finish them before the backend stuff is finished, because you can always work with mock data. So you don't need API to be able to test the front-end stuff. And for example, in the backend, they can build a backend without the front-end and tests, test the API and communication and stuff like that. So I think I haven't worked with so many product development companies before that actually do it like this. Uh, I have been working as a consultant before, like two years ago, but it was very unstructured and you were very free. You can do whatever you want, just follow the spec. You could write stories for yourself if you wanted to, uh, and they encouraged you to write stories, but it was only for yourself as a developer because it was normal. Uh, the normal way of working was usually one developer that worked with one product, and then they had some bigger products working with many developers, and they had more requirements for, for the sprint planning, et cetera, and to split it into tasks. So this is actually one of the first product companies I've been working with that is actually working the way a thought that product companies were always working with.
- Speaker 0 00:14:03 Uh, so then when you, apart from when you have a entirely new type of projects, such as this bike rental system, are there other occasions when you, or which other occasions do you use, or do you perform user stories splitting?
- Speaker 1 00:14:26 Um, we usually have it for, for our own internal team, for example, our support team and us, ourselves as developers. For example, we have caching and caching is incredibly hard to do and to implement. And if you have a bug, it's probably the caching layer that has the bug. So for example, we have one mode. The previous tasks that we had last year was that we needed to fix the caching layer. That was one of the big projects. And then we needed to find like the stories that were required for, for ourselves. The support team wants to help the customers to clear the cache by doing this. They want to clear the product cash. They want to care the pricing cash, maybe the product images, image cash. Uh, so then we have to split it into smaller parts for every individual requirements. Okay. So that's why that is a good example, I think.
- Speaker 0 00:15:40 And who are, uh, which roles or who is involved in these user stories, plating sessions or in its process?
- Speaker 1 00:15:52 Um, before we restructured how we work, uh, it was actually ourselves as developers. And if we wanted some input, we just went to the support team or colleague and just asked, how do we want to work with this? So we didn't have like a team leader. Uh, we only had one boss for the whole company with a vision. So, um, it was ourselves who could decide how to split it and how to work with it. Uh, and some tasks require that the person working with have some experience. So it's different from a junior perspective, for example, when developing stuff like for person that have seen your experience, how they see the problems, how they see

the solutions to their problems and how to split big tasks into smaller tasks. Uh, so yeah, that's, um, could also be different because depending on how long you have worked with it,

- Speaker 0 00:17:06 Did you say they, is it a collaborative or does it have a collaborative element to it, or is it a little more focused on one
- Speaker 1 00:17:15 Now after the restructure? Yes. Now we collaborate, uh, uh, every second, Monday when we have the sprint planning meeting and now we actually have a team leader. So now it's easier to like filter everything through the team leader, to the developers in each team. So now we discuss a lot and we prioritize which we wants, uh, but how to split the big tasks is pretty new for the bike run example. Then we were maybe one or two people, uh, discussing how to split it into smaller parts and more logically. Because for example, if you build a bike rental system for three teams, then you want to test, for example, parts of the front end client before the complete client is finished. Okay. So you want to say, okay, does it work with one product? Does it work with two products? Plus it work with mock data. Does it work with the API?
- Speaker 1 00:18:37 So it's, it's a process of finding these steps in the process and how can you test the components that we want to, that we want to build in? How can you test the components when they are interacting with other components? Uh, and then in a final product, when you're adding guest information, when you're adding a car, so all of these parts have to fit together, but most of it, most of the help comes from the developer and the product team leader can have some, some inputs, but this usually the developers who have the final word.
- Speaker 0 00:19:24 Okay. It almost sounds a little bit like you are like, you have a intention to split on a, with a feature in focus. So as you said, but could that also be a feature such as create the API and then the API is a feature or is that part of something? Yeah. Sure. Okay. Okay.
- Speaker 1 00:19:49 For example, for the backend team, uh, that's the team that I'm part of, um, I'm starting to get cross functional, so I'm in the front end team and backend team for the web, but most of my time goes into the backend team. So for example, if I want to take the information from the booking system and make it available in another format with the web's own back in API, then I will decide, alright, which features do I need to build? Or which functions do I need to build just to make it work? Uh, one example can be, you know, um, like, okay, how does the client want to talk to the web API? What information do they need? And then I have to ask that our two [COMPANY NAME] team, what information can you supply? What information can you give to us about the bikes and what are the guest requirements for each guest, riding a bike. So in this case, we only have lengths and name as a requirement for each guest, riding or rental renting a bike.
- Speaker 1 00:21:16 And then I have to take that into consideration and also add some stories so that I can test the data that I received from [COMPANY NAME]. And then I add some tasks how to transform it so I can test the transformation process. And finally, I add some tasks like, okay, now the web back in API wants to present this information to the front end client. How can I implement that? So that's also some sort of story splitting. So, but the main story could be like, okay, build a backend API for bicycle rentals. And then I have to find each sub-task within that and then make the splits myself. Yeah.

- Speaker 0 00:22:19 Uh, so would you be able to describe how you think that you, maybe your colleagues is better, how they perceive the way of, uh, the way that you use source?
- Speaker 1 00:22:41 Well, what should I say? How they perceive how we work with it or,
- Speaker 0 00:22:45 Yeah. So if they have, and this could go both ways, if they, if you think that they find things difficult, if you think that it makes it easier, those kinds of things.
- Speaker 1 00:22:59 Yeah. We know this, that it makes it very easy to, to test each function individually and release each story to test more often. Okay. So I don't have to be finish the whole product before I can release something to test. I can be a little, maybe 10% of the product and release that one for test. If I have a separate story or task for that. And when working with my colleagues, I think that everyone thinks that it's much easier. Now, if we can split it into smaller parts, then that is much better than to have like a two big test. Um, because then you know that now I'm finished. And then you also, for example, if you give the task with the status to ready to test to a support colleague, then they can also read the task and the description and know what to test and what that specific feature does. In this case, it's really hard for the support team to test the backend stuff. They can only test the front-end parts. So it is ourselves, the developers who are responsible to test the interactions between the systems. Um, and usually it lands on my own desk to test my own stuff. Uh, and then when we have more complete systems between the teams, then we can try to make it fit together. And then we bring in the team leader to help with the testing. Okay. So that helps really much to make the, the stories as, as small as possible because we can test them or individually and we can release them it through it, through to replete. You can deploy them
- Speaker 0 00:25:08 Difficult, very difficult. So what, what aspects, uh, do you take into account when you need to split something from big into smaller pieces,
- Speaker 1 00:25:24 Uh, time focus and, uh, to minimize the amount of code I have to write to minimize bugs because usually we already have a big system in place. And when we want to add some features, I really don't want to add code, um, all over the place. I, I need to find one or two areas that I can focus on. And then I tried to find smaller tasks that focuses on those points.
- Speaker 0 00:26:01 Okay.
- Speaker 1 00:26:03 So for example, uh, if I split the backend Epic into smaller parts, one can be that add support to search bikes.
- Speaker 1 00:26:15 And then I know I just have to create a new class. For example, that takes care of the searching. I don't need the backend API to be ready in [PRODUCT NAME]. I don't need it from them to be finished. I only need the function and that I can test the values I put in and the values that I get back. So that's one part. The second part can be that I want to test, uh, just the booking process of a bike. And that's also very focused area. It's, uh, only one file I have to be in. And I know exactly where in that file, I can do the code. And that also mean to my sister bugs and also increases, uh, what should I say? No, it makes it easier to review your commit, seeing it afterwards, um, in the tool yirah that we are using now, uh, we name each sub task the same as, uh, the number on each task.
- Speaker 1 00:27:24 For example, [PRODUCT NAME], uh, online one, two, three, that's the test number. And if we give, or if you write the same number in a gift

message, then we can, uh, we can get a list of all the commits that belong to the specific task in JIRA. And then we can click the list and see, all right, this task required, five commits, and then I can go to beat pocket and see each commit and see, and also review the code that I, or one of my colleagues have written. So that makes the code review process a lot easier if you have smaller tasks, because then you don't have so much coding.

- Speaker 0 00:28:09 Uh, so do you know of any, uh, either guidelines in terms of, uh, guidelines provided from your company or different needs that have been implemented or adopted to kind of deal with the process of splitting user stories?
- Speaker 1 00:28:31 Um, no, nothing that I have seen so far, uh, the only structure we have is that if we have a task and we estimate the time to be, for example, more than four hours, then it's not a small bike anymore, then it's a feature maybe, or a small feature or a B bike. And if it takes more than three days, then that's the normal feature. Okay. So, um, that's one of the most important, uh, aspects of how much time it takes, but we don't have a process or a documented process of how to spend them. Uh, I think it's just comes from experience and how this, um, how you're working with a product and how you usually make changes to the code, um, part by part.
- Speaker 0 00:29:32 Okay. Uh, so then do you know if any, do you know of any specified definition of, um, when something has successfully been split?
- Speaker 1 00:29:46 Yeah, I would say the bike rental projects that's, what's successful. Yeah. Uh, I can see if I can just, uh, for example, this one is for, uh, for example, this is for the front end part. Uh, the front end part for bike rental was split in one, two, three, four, five, six, seven, eight, nine, 10, 11, smaller, uh, smaller parts or tasks. Uh, the end part was only one big task with some minor subtests that I created myself. And then for the, [COMPANY NAME] team, uh, they didn't have like a project for the bike rental. They had, uh, just multiple big tasks that they, they planned it, then they estimate the time and then they just took one and tried to finish it as quickly as possible. So the thing that we went back in and back front-end team has worked more structured with the bike rental projects.
- Speaker 0 00:31:03 So if you were to take, what, what would you take away from, uh, the success of that project in to next project? So what aspects are transferable into an different or new project?
- Speaker 1 00:31:20 I think, uh, it's important to ask the developer who actually implements the tasks that are required and to just ask them, how did you feel about these tasks? Was it too complicated, too, too big? Uh, or what's the time estimation way off? Uh, I think that's very important to, um, talk to each other and ask, what did you actually feel about the project and the tasks that you received? Uh, you can always look after the project to say, okay, we estimate that every task to maybe eight hours a piece and then the project to maybe 50 hours or 60 hours to build, but it doesn't say so much about how the process actually, once the developer thought about working with it, or if they had a long time to bargaining the features, or if it could help, like splitting one task into multiple stories, could that improve the, the debugging process or maybe that they could test it more quickly or be done with it more quickly? I don't know. Uh, we just have to be better to, to review the project afterwards to see how it actually works.

- Speaker 0 00:32:53 So my, my next question is falling quite relate to what you just said, uh, which is, uh, if you ever evaluate the process of the users, story splitting retrospectively,
- Speaker 1 00:33:08 We haven't done so yet, but, uh, because the bike rental system was finished, uh, on Friday last Friday. Uh, we are actually in the process right now. Uh, we have, we still have the developer until today to finish off the last couple of bugs that were found. So when his work is done after today, then we will receive the documentation for it. And then we will also review, uh, what he thought about the project. Um, this person is a consultant that we hired to do the front-end work. Okay. So this time we will actually review.
- Speaker 0 00:33:56 Okay. And, but this is not something that you have done thus far within the company. Is that the new thing?
- Speaker 1 00:34:02 Nope. Okay. It's the new thing just to now, we want to really find the best way to work with it because we could have like 600 or 700 features and bugs mixed in, in just one big pile of tests. So it was very hard to get an overview of what you had or the problems you had or the problems you didn't have. Uh, so that's one of the new things with this structure that we have started with, uh, to understand this better. Yeah.
- Speaker 0 00:34:42 To kind of wrap up slowly. What do you think that the main benefits are of performing a user stories splitting?
- Speaker 1 00:34:54 Yeah, I think it's a, it improves the way for us to see, to know, to get more focused, to better estimate time that you need to spend. Um, and you can also easier see, um, how many parts of the system one task will affect. So if, for example, if one task affects many parts of the system, we want to minimize it by making smaller tests so we can be more focused in the code.
- Speaker 0 00:35:34 And, uh, then to contrast, uh, the positive points, uh, can you see, or have you had any experience with drawbacks of user story splitting and what are those in that case?
- Speaker 1 00:35:52 No, I don't think it's negative in any way. I wouldn't say for me, it's only positive to have a big task split into smaller parts, so it's easier for me to work and it's easier to estimate. And it's easier to just say that, okay, this morning I will focus on this task. And after lunch, I will focus on these tasks. Uh, I don't see any bad or negative effects on that. All right.
- Speaker 0 00:36:24 Uh, my colleague doesn't have any questions directly. Uh, do you have anything that you feel that we have missed to ask you about, or would you like to, um, provide any last information that you think might be valuable in this context?
- Speaker 1 00:36:45 Yeah, I'll see. I think I said a little bit about it before, and it's, I think that experience is key, uh, to have a good splitting of a project because experienced developers or team leaders, they will see the smaller parts, uh, in a big system and a B project. And maybe if you have a product owner, uh, the product owner also knows exactly how or should know how the product works. And then it's easier to give the developers hints of, okay, you should focus on this part first, so you don't make it over complicated. So yeah, I think the ex the developer experience is really a good key to, uh, to, to listen on and to just use with the next planning of a story splitting.
- Speaker 0 00:37:57 All right. Thank you. Thank you for your time. I will stop the recording now.