EPISODE 1156

[INTRODUCTION]

[00:00:00] JM: Pair programming allows developers to partner on solving problems and learn from each other more effectively. Pair programming has become harder to do as remote work has become more prevalent. GitDuck is a tool to enable more effective pair programming.

Dragos Fotescu and Thiago Monteiro are the founders of GitDuck and they join the show to explain what they've built and their motivation behind GitDuck.

[INTERVIEW]

[00:00:30] JM: Guys, welcome to Software Engineering Daily.

[00:00:33] DF: Thank you for having us.

[00:00:35] TM: Yup, thanks.

[00:00:36] JM: You're the creators of GitDuck, which is a pair programming tool. Why has the tolling around pair programing historically been lackluster?

[00:00:45] DF: I don't know. I tend to think more the other way around like why is people getting with more [inaudible 00:00:51] on this right now, and obviously with all the COVID situation, this is one answer, but also think about all the trends on remote work that was happening in the past years. That led a lot of people to start demanding better tools for remote collaboration. So I don't think that's like lack of good product in the past, but it's more like now it's really the time to have a good product for that.

[00:01:17] JM: What kinds of tooling do you need to build a pair programming tool?

[00:01:22] TM: Yeah. So in our case, we are more than a pair programming tool. Pair programming is kind of like one piece of that. So one part, very important for us is the video allow to share so that you can talk with each other. You can see if you need. You can share a screen if that's important. And then you have all the real-time connections to be able to share your code very, very instantly and you are able to literally send each other's file so that you have the file in your computer. So there is this real-time path that we put all of our attention to have low-latency and have people collaborating very instantly.

[00:02:04] JM: So you can collaborate over an IDE in GitDuck, and it kind of looks like a Google Docs style experience. How would you compare a Google Docs-like application, shared writing application to GitDuck?

[00:02:23] DF: Yeah. Actually, that's a great analogy. We use that a lot to explain what we are doing, and the way that we think about that is, yeah, it's like Google Docs, but with very deep integrations to the tools and the workflow from the developers write. So you start with that very basic insight of being able to edit each other see, or let's see what Jeff is writing right now and see and follow another person. But then we are adding more and more things that are related with how people are working to be able to just observe somebody coding to be able to following all the time, or it's somebody following you. Yeah, that's where we get different from just like a basic real-time typing docs.

[00:03:05] JM: What have been the hardest engineering problems in building GitDuck?

[00:03:09] TM: One of those are like developing integrations for each IDE or for each tool is like a different world. It's like developing an app for iOS and android and 10 operating systems more. And it's like each study has its own frameworks, its own like languages. So we build these core, what we call like core, which like abstracts and packages all these logic. So we can just build the specifics, different part from each IDE and, and those IDEs just connect via an API to this core. So we don't need to rewrite continuously this kind of logic for each IDE or integration.

[00:03:51] JM: Tell me about an integration between an IDE, a particular IDE, let's take VS Code, for example, and the other tools you built in GitDuck. What are the points of integration?

[00:04:02] TM: Yeah. So we need to integrate like code editor, like what you're seeing, what you're typing and the connections. We need to authenticate each user to a GitDuck account to each IDE. Then we need like – A lot of people are asking for, shouldn't the output of the build output or of the tests? So these kinds of integrations. I'm not sure if I answered your question.

[00:04:27] JM: Here's another approach. So let's say I opened up my IDE and I opened up GitDuck, what happens?

[00:04:34] TM: Sure. First thing you need to open, you need to enter a GitDuck on the web. And the moment you do that, your IDE wakes up and detects that, "Okay, you are in a meeting on the web." You can start sharing your code for the other participants on that meeting, right? So the moment you click start sharing my code or other participant of the meeting clicks start share my code, the other ones, the other participants can view that code and enter in this collaborative mode. And basically the host, the one that's sharing the code, it's sharing his whole project, his environment, and the other participants can access his local files and can edit and view accordingly.

[00:05:16] DF: Yeah. And one cool thing is that what actually is happening in the background is that you – Each person is typing on their own IDE and they can be used in different IDEs, by the way. But then we are applying the changes and background to the host machine, right? It feels like everybody is typing real-time, but actually everybody is essentially being applied in background, but changes to one file, the one that is hosting the pair programming session.

[00:05:44] JM: So when I type into my IDE, it's writing to a shared machine in the cloud?

[00:05:51] DF: No. It's all b2b.

[00:05:54] TM: Yeah, exactly.

[00:05:55] JM: It's all peer-to-peer.

[00:05:56] TM: Yeah. You are actually writing on the host machine.

[00:06:01] JM: Gotcha. Okay. So I write on the host machine, and then how regularly is it sending over the wire to the other machine?

[00:06:09] TM: Yeah, it's pretty instant. I mean, if you paste a bunch of texts that's going to be like batched. But it's very low-latency and real-time.

[00:06:22] DF: Yeah. I mean, the stress test that we do is basically we all go into a pair programming session and typing as fast as you can and just seeing who wins that race, right? So those are the tests that we are doing to allow people to like having this feeling of, yeah, we are typing real-time, although everything is being applied in background to the host's machine.

[00:06:44] JM: Gotcha. So the aspect of the video, to you outsource the video to something like Twilio or what kinds of off-the-shelf APIs have you used?

[00:06:57] DF: Yeah. So one basic insight that we have is that we put all our brains on the stuff that matters for us. And in our case, it's all about the deep integration. So that proves that tools. So we are indeed outsourcing the video path. We are using daily.code. So we kind of like trust that all the audio or the video or the screen sharing is out of the boxes, everyone is great, great quality. And gives us, actually, as we are putting a lot effort on the audio, on the code sharing sorry, gives us opportunity to optimize the video and the audio a lot, because we know the use case. So for example, one, the main feedback is that the video is the least important thing. When you jump in a call in a Google Meet or a Zoom, your video is always like full screen. You're seeing each other's face, and that's what's important. I mean, that's not important at all. What's important is the code, the content, what we are working on. And that's where put a lot of all the UX and all the details that – Everything from scratch, and how do we share tools that you're working for for development.

[00:08:09] JM: So daily code, that's like a video streaming system, right?

[00:08:09] DF: Yes, it's a white label –

[00:08:17] TM: Video as a service.

[00:08:19] DF: Exactly.

[00:08:21] JM: Cool. So I believe that you guys built GitDuck while you were learning JavaScript. Is that correct?

[00:08:29] TM: No, not really.

[00:08:30] DF: Not really. We knew a bit before. We have been using the same technology for a while in a couple of old projects as well. So we have been getting better with that.

[00:08:44] TM: Yeah. Actually, we built GitDuck, because we started working remote, like before remote was like mainstream. And we'd like noticed how hard it was. At the time we were working also on video related technology. And we started using this video related technology, which was a screen sharing app, screen recording app to share knowledge or to ask for feedback or advice. And from there, we started pivoting into what is GitDuck.

[00:09:14] JM: Tell me more about that pivot.

[00:09:17] DF: Yeah. I mean, like we started building that. Well, the true story is that we were building a fintech, and we run out of money. So the company failed. But we still have energy, right? We decided to keep working together on something new. What we decided to start working on was a user for the back tool, basically was this Chrome extension that was recording the user's screen, giving them tests to perform so the startups could understand what they think when they sign up, things like that.

As Thiago said, we started working remotely on that moment, and it was quite hard. We used to work on the office and we start learning from ourselves, like on communication wars. We struggled a little bit and now process to learn how to work on that different setup, that different context. We started using all to work, to ask technical questions between ourselves, just answer very simple things. And we discovered the GitDuck use case. We knew a lot about rubber duck debugging. We liked a lot this methodology, and we just go there. Okay, that could be something very interesting. We started talking obviously with more people, people from different kind of company sizes, because at first we thought that was like only remote issue, that only remote companies were facing. And we discovered that, no, a lot of people are facing those problems, people that had like distributed teams who are facing those issues as well. And we decided to go – Well, we honestly thought that GitDuck was a much interesting problem and much bigger thing to work on, and we decided to stop developing the other SaaS tool and start to tweak out technology that we built previously to work for developers.

[00:11:07] JM: The live stream capability of programming, is that something that you could see GitDuck being used for?

[00:11:16] DF: Yeah. And actually it's funny that you mentioned that, because that was the previous iteration of the product, we had the live streaming. And actually it's still live, but it's just hidden. We see that a lot of people are doing live streaming, specially these days on Twitch and YouTube. So we had that functionality. The reason that we decide to change the focus from live streaming to private meetings was just because we learned that we ourselves were using the tool in a different way and also we discovered that all that users were using it in a different way as well. Basically we learned that people, for example, are doing meetings with Google Hangout and streaming with OBS their screen to our RTMP servers so they could see the live meet in GitDuck. But while using others tools to do that.

So we suddenly learned those things and we found that actually that was very interesting, and we decided to build a new product from scratch focused on the real-time collaboration, and we literally built everything from scratch. We have like kind of two products. One is we are not put in front of people anymore, but it's working. I use it for recording videos for explanation video for our users. And the other one is the real-time collaboration with pair programming.

[00:12:40] JM: Gotcha. And when you say you build everything from scratch, is there anything that you built from scratch that you could have taken off-the-shelf?

[00:12:50] DF: Oh, definitely. I think that video that we mentioned, that I explained before, it's a good example of that. When we built the live streaming for developers, the previous iteration of GitDuck, we built a video streaming technology ourselves, and it was great, it was fun. We learned a lot. But the problem that we were facing is that we were spending a lot of time thinking on the basic things to make it work, the video, and not putting too much time on the things that make us unique.

So when we decide to start building the new real-time collaboration product, one of the first thing that we thought, yeah, the video must be working. We just don't need to worry about this. Let's just focus on the code piece, right? How people can share the code and collaborate with pair programming in a very efficient way.

[00:13:41] JM: Can two people work on GitDuck parallel to each other, or do all the participants have to be kind of serialized?

[00:13:52] DF: Yeah, even more people, three people, for people can be working in parallel. One is always the host. So he's the one that is sharing the files to everybody to participate and collaborate, but then people can be editing things in parallel, working on different files, different parts of the file, whatever people think is more productive for them. And then what's happening is that all the changes are applied to one person, the host. He's the one that is going to do the commit later that contains all the changes made by all participants on that session.

[00:14:30] JM: Are there any issues around that synchronization that are difficult?

[00:14:35] TM: Yeah. I mean, a lot of issues. It's actually like we don't really have like major services architecture, but in this environment, each pair is like a different service and each pair has its own data. And you need to maintain the consistency across everybody. So yeah, we definitely had like issues like synchronization issues, like one person will sync something, the other person will sync another thing. Yeah, definitely its tricky.

[00:15:05] DF: Yeah. Because, I mean, besides just synchronizing the files in real-time, the issue or challenge, or at least what the real challenge that we have is that each person is using a different tool. So you could be using [inaudible 00:15:18] using WebStorm, and I'm using Vim, and we should be able to collaborate very easily, and that's a big challenge, because all problems that they have, like different architecture. So we need to handle those things as well. So it's a big part of the value that we are building for.

[00:15:37] JM: So if I understand correctly, if I'm the host, let's say I'm using IntelliJ and I'm loading my Java files, GitDuck is going to be able to track the file system, like changes to the file system and then somebody else can be using VS Code in another computer and it's going to be essentially ported over or it's going to send it over the wire and then send all their edits over the wire and then be applied to the file system of the person using IntelliJ.

[00:16:09] TM: Exactly. The only thing to mention is that we are not watching the file system itself. We've considered this approach, but it's tricky, because you can only see the changes when you saved. So it didn't work for us. So what are tracking is the IDE. So we help hook up into the IDE IPIs. We detect when you made some change inside of the IDE and we send those changes to the other participants.

[00:16:37] JM: And the ways that the IDE is watched by a host, are those different from IDE to IDE? Does VS Code have a different system of representing code that has not been saved to the file system than JetBrains?

[00:16:54] TM: Yeah, definitely. I mean, every IDE is a world, and actually there are IDEs that makes the job easier and there are IDEs that makes it harder just because they have like more limiting API or they have like worst documentation or something. But yeah, definitely it's totally different. Their representational defaults or how you change the files and so on.

[00:17:19] JM: What are the places that you'd like to expand into from the pair programming tooling?

[00:17:26] DF: Yeah, that's the part that we get super excited as well, because pair programming is just like a first use case that we are covering. So the way that we think about is basically every dev tool, everything that you need to work, we are going to find a way to integrate and make it useful. So for example, the next thing that we are going to do is terminal sharing. So we can easily share terminal and work with others. And the other one is being able to share your server, the local server. So for example I had here the local version of GitHub that we are just tweaking some things and we are going to deploy a couple of hours. And with GitHub, we are going to be able to allow people to just browse that, like even if you don't have the local server on your machine, you're going to be able to browse that, check the changes. So if we are working parallel on that, you can see the changes being applied to the local server. So that's the two next thing that we are super excited about.

And then there are lots of small improvements that are super important to the way that people are collaborating or sharing their coding real-time. I think the one is people are requesting the most is that to be able to share their whole project and just share everything. And then the colleagues can be picking and working on the file that they need. So that's something that we are looking a lot as well to be improving on the [inaudible 00:18:51] part besides adding more integration to GitDuck.

[00:18:58] JM: Can you explain more? What do you mean by sharing a server?

[00:19:03] TM: Yeah. So it's basically like tunneling the local ports you have or your local services. So let's say I'm developing a service in Node.js, another in Java and I have my database. So I can expose those services that are internal in my local host on my local machine and I can just host those servers to services to the other meeting participants so they can access those services without needing to run them.

[00:19:27] JM: So why is that useful?

[00:19:31] TM: Yeah. Well, so when you're pair programming, you want to see, for example, the output of how your website looks or which new entries you have on your database or how your REST API, or maybe throw an integration test to your local REST API, something.

[00:19:51] JM: I see. Okay, that makes sense. So the big vision is basically system of having tight collaboration in programming for all different vectors of programming.

[00:20:06] DF: Exactly. More and more integrations that make just like remote collaboration to be as easy as if you're in the same room side-by-side.

[00:20:17] JM: So let's talk a little bit more about the engineering. Let's say I have my IDE up and running and I start GitDuck, what happens? What does GitDuck start to do to monitor that running IDE?

[00:20:32] TM: Yeah. First of all, GitDuck is like – Your GitDuck extension is going to listen to check if there's any active meeting on the web. So first you need to be in an actual meeting for the extension to wake up. Once you have that and you start sharing code, basically each IDE has APIs to listen to events like once a document changed, right? So we just listen to those listeners, to those events. And when those events happen, we just process those events. We send them to this core thing that we build via GRPC, and this core thing is actually basically the thing that checks, "Okay." That ensures the eventual consistency that sends the changes to the other participants and so on.

[00:21:25] JM: The pair programming tooling space, are there any competitors?

[00:21:34] DF: Yeah. I mean, I think the main reference, the main competitor right now is Double. They're doing a great job. Screen, they are also doing a great job. Both have from being a successful on-screen hero. Yeah, those are things that are the main direct competitors that we have on that space. I mean, that's exciting. I think a couple of years ago, there was nobody doing that, and now there are some people. So I think that's great for people.

[00:22:03] TM: Yeah.

[00:22:05] JM: One thing you focus on is low CPU and bandwidth consumption. Was there anything that was hard to do to get the consumption down?

[00:22:15] TM: Yeah. For example, there are alternatives that Thiago mentioned. They're all based on video, and video, just screen sharing consumes a lot of CPU, and there's like impact already the bandwidth and the latency and the CPU. Double does a great job on that, but still it's not like real-time. Like if someone is editing in your keyboard. Yeah, I think the fact that you don't need to screen share to share your code, that's the thing that impacts the most latency and the bandwidth and the CPU consumption.

[00:22:49] JM: And so how did you measure that and get it down?

[00:22:53] TM: Well, it's very noticeable. Because we used to share our via screen share, so you really notice like your – Like even on Zoom or Google Meet, if you enable your screen share, your CPU, you can hear your fans. It looks like your laptop is going to fly. So just by that, you can already see an improvement.

[00:23:16] JM: The design of GitDuck seems to be an important facet, because you need to make this a tool that's really easy to use and really enjoyable to use. How has design played a role in the engineering of GitDuck?

[00:23:31] DF: I think it's a main role that we have from the way that we think about product, the way that we design the things that we do. And the way that we tackle the problems that we're trying to solve not only on the visual design, but only also on the UX and how things are perceived and how we decide what this may build.

I think one very simple insight that we have, that we think a lot, is what we can remove, what we can really uninstall or remove from the user side. So the interface and the user flows on how the product work is really, really simple. We put a lot of effort and maybe more time on what we can remove from GitDuck and what we can put there.

And just having these limits, constraints on not having like a blank canvas that you need to fill, but just having to understand what is really important and what we need to prioritize and how we organize the information in a [inaudible 00:24:32] way. I think that's something that is we put a lot of thoughts in. But that said, I mean, the one thing that we really like to do is just change everything. Everything is being changed all the time. So the version that you see now in GitDuck is it's being already changed, and we're always doing that. We are always trying to improve. We're always experimenting on the best way to do things and trying to find the voice of the product. I think that's the very important thing when you're thinking about early stage, early times so that when you're still trying to figure out who you are and how you do things, or when you start to understand how your product behaves on what the voice that you have. It becomes easier to make decisions or this doesn't make any sense on our product. Or, yes, this is absolutely something that we need to add. Even if it's not very well designed. So there is a difference between polished, crafted things, that the UX and UI are perfect. But the other things that might be a little bit like with raw edges but makes complete sense from the product point of view and what the product means. And we don't fear to just add them and to see and hit the rate of broad basically. That's something that we do a lot. We pride just to build things and put as fast as it can things on people's hand and try to iterate with them and just don't be afraid to change things.

[00:26:09] JM: Tell me more about the architecture of GitDuck.

[00:26:12] TM: Yeah. So we try to keep it really simple. So we have like our main language is JavaScript or TypeScript. So we try to keep this language if possible. We have our servers in Node.js. Our frontend is Next.js plus React. Then we have – Each extension is a different language. For example, the VS Code extension is TypeScript. The JetBrains extension is Java and Kotlin. Then the sublime is Python and so on.

And then we have this core that we created, which is in TypeScript, and we actually compile that to binaries and we distribute that with the extensions that doesn't support TypeScript or JavaScript. Yeah, and this binaries communicate to the main extensions, processes via GRPC. I think I mentioned everything, like high-level. I mean, I don't know if you want anything more detailed or something.

[00:27:13] JM: Give me a little more color on the GRPC usage.

[00:27:16] TM: Yeah. So basically, like this core is made in TypeScript. So it's very easy to – It's the common logic, like what happens when you receive a new change, right? What happens when you enter in a meeting? Like the extensions wake up. Basically, it's this common logic that doesn't – It's totally decoupled of each IDE. So it's totally IDE agnostic. And it's very easy to use it. It's like an API, it's very easy to use it in TypeScript, because TypeScript is TypeScript. But when you have like Java, that you cannot integrate this TypeScript into Java. Basically, we have this server that communicates via GRPC. And basically that's how it communicates with this core API, right?

[00:28:03] JM: What are the security challenges that you've come across in designing GitDuck?

[00:28:08] TM: Yeah. I mean, we take security like super serious. Actually, I have a background in security. Yeah, we try to keep it like that's like super important, because at the end you're sharing your code. So things we've taken, like everything is ramped via P2P, like the code, everything is P2P. Video, we use the P2P mode of daily. Yeah, everything, we have to 2FA and everything. We have on-premise options for customers that need it. Yeah, and we have like bug bounty that we recently opened. So if any of the listeners is willing to attempt to hack it, as long as it doesn't harm anybody without permission is welcome to do so.

[00:29:00] JM: What are the places that you're focused on in GitDuck right now? What are you guys building?

[00:29:05] DF: Yeah. So right now we are building the extension for Aton, that's the new extension that we should be releasing very soon .We are doing a lot of improvements on the way that the extension, that all extension work. We have just released the integration for JetBrains. So now we support [inaudible 00:29:24] JetBrains, the amount that people are using this. We are going to already be supporting Aton and maybe more.

And the other thing that we are starting to work on is to be able to share the whole project to the whole folder. That's something that people are asking a lot and we are super excited to implement this very soon, maybe the end of this week or early next week, we are going to have it. So yeah, exciting things that we are working right now.

And the one cool thing is that world mapping actually is transparent. So people can follow and see what we plan to do and what we are working on. So people can just basically vote on things that they think that's more relevant. So we take a lot of consideration what people are asking for us and they see a lot to the users to understand what's more important to them.

[00:30:23] JM: GitDuck is entirely browser-based, right?

[00:30:26] DF: Yup, and the extensions.

[00:30:28] TM: Yeah.

[00:30:31] JM: Have you thought about building an electron app?

[00:30:34] TM: We definitely thought, and we think eventually probably we are going to switch to it, but I mean right now, it's much easier to build a web app and just iterate on that.

[00:30:46] DF: Yeah, I think one thing that it's all about speed, right? In these early days, we optimize a lot by how we can ship things as fast as we can and just by having the web-based product that allow us to iterate faster. But as soon as we have – Or have some time, we are definitely going to do that, because this just give us the opportunity to do other cool things as well as others things that we can improve the product. So yeah, we are going to do that eventually.

[00:31:16] JM: How do you guys test GitDuck?

[00:31:18] DF: By using it. Trying to break it.

[00:31:23] TM: Yeah. Right now we have a lot of automated testing in place mainly because of our stage and our iteration. Definitely we are going to work on that once we're more stable. But yeah, right now the main testing strategies is use it ourselves a lot. Talk to the users and listen when they complain, yeah. That's most it.

[00:31:44] DF: I mean, we are the main users of the product, right? So the benefit of using the product every day and using that as the basic feat of your collaboration, communications that every time that people get frustrated of it and every time that something fails, we have the power to fix it, right? So that's a privilege, more opinions. So we take very serious as a user and as the builders of the product to do something that we are proud of and that when we are using, we are delighted about it and we are happy to be using it. Yeah, [inaudible 00:32:20] of just listening to people. That's how we test it.

[00:32:27] JM: Did you say that there is a number of users that it starts to cap-out at or it starts to have too many users and there starts to be problems?

[00:32:36] TM: Yeah, it actually can support up to 20 people. We didn't test it with more people, but it definitely should work well with that people. I mean, usually it shouldn't be – Well, we've the request of live streamers. But right now we are not focusing on them that much. So yeah, it looks like good amount of people support it.

[00:32:59] DF: It also depends on the use case, because if you are working, like if you're collaborating with your team, there is a normal size of a team that people are usually having and that they working together. But if you start to – And that's what we are mainly covering right now. But if you start to think about people learning to code online boot camps, universities, things like that, the context changes a little bit and you start to have more people live. Yeah, they are using the product, although it's not really perfect for them yet as we are focused more on the small teams collaborating remotely.

[00:33:39] JM: What's your deployment model for GitDuck?

[00:33:43] TM: Yeah. So we'd use GitHub for code repository, and we have like GitHub actions that listen to – Well, it's actually manual step. We trigger the GitHub action manually. And that triggers each extension. It builds each extension for each platform. It also builds to give that core binaries for each operating system. And it also deploys the servers to Google Cloud platform.

[00:34:15] DF: Yeah. I mean, we have just automated this last week, right?

[00:34:19] TM: Yeah, it actually used to be manual. We just automated that because it started to be painful. And every time it was like longer and longer, and the chances of doing a mistake were higher. So we recently automated that, yeah.

[00:34:38] JM: So the deployment unit, is it a VM? Is it a container?

[00:34:43] TM: Yeah, we actually run containers in GKE.

[00:34:48] JM: GKE, okay. So you're mostly on Google Cloud?

[00:34:51] TM: Yeah.

[00:34:53] JM: Any reasoning behind Google Cloud over AWS?

[00:34:56] TM: We had more experience with Google Cloud, and we feel like the UI and whole experience better. That's the main reason.

[00:35:07] JM: I guess you guys don't have much intensive database requirements. So I probably don't need to ask about databases.

[00:35:13] TM: Yeah, not really. At least not for now. Like we've been exploring the use case of recording a meeting. Basically, like you can replay what you have typed, right? So that would need to be stored in a database. Right now, we don't have. Yeah, definitely, we don't need much database. Basically just for the user profiles.

[00:35:35] JM: And what's your pricing model? Are you charging people yet?

[00:35:39] DF: Yeah. So people can use GitDuck for free and there are some limits. If you have like up to three people, and there are some limits on how long the session can be. But yeah, for teams, for startups, for companies, we have like a paid model that basically it's $20 per user per month where we give more features, better control of the team, better management of everything. So yeah, that's how we are monetizing.

[00:36:14] JM: What was the fintech product you guys were working on before GitDuck?

[00:36:18] TM: It was a digital piggybank, how people save money automatically. So we are basically doing bank reading. So every time that people are buying something with their debit card, we were reading their bank transactions, calculating the spare change and transferring that money with extra charge through the debit card so they could save that digital spare change or other rules. Basically you could add rules, like say like I want to save 10% of everything that I'm buying, things like that. And that was focused on the banking regulation in Europe where we're building that. And yeah, it was called SaveBoost.

[00:36:59] JM: Was there anything about it that was particularly difficult? Or you just decided that GitDuck was a better option?

[00:37:05] DF: I mean, like technically there were all the challenges there and other things that were easier. I think technically security was a very important point in that time as we were moving real money. Since now, we have a lot of – We think a lot about that, how to make a like similar service and how we can communicate through us even though we are a new company. And that's as – I mean, that's part of our DNA when how we viewed the products, how we do things. So that was one side. The other side is that – I mean, the company failed because we run out of money. We failed to fundraise. And one very important learning that we had in that time is that it's very hard to build a B2C feedback product without enough funding. So that was something that we learned later, and we had the opportunity to rebuilt it from scratch after we closed. But we got more excited about the problems that we are facing also with how we were working. That was something that was always in our mind. So we decided to work on that.

Although, I mean, I personally still miss the product, like the product that we built was super useful, myself personally. So I think that's the thing that I miss the most, is the product itself to use it, not really building that company, but it's more like using it.

[00:38:32] JM: Interesting. So you guys were – You were users of your own product.

[00:38:36] TM: Yeah.

[00:38:36] DF: Yeah. That always helped. Like when you have the pain on your users, that's super helpful.

[00:38:44] TM: Definitely.

[00:38:45] JM: Cool. Well, anything else you want to add about GitDuck?

[00:38:49] DF: I mean, I think like one cool thing is not really super relevant. But I don't think anywhere else. It's a super nice opportunity as you invite us to mention like our fundraise. We have just closed our seed round with General Capitalist, with Notation Capital, Gradient, the Google AI fund, Y Combinator and others. So we are super proud of the partners. I think for us the way that we think about that is just it had helped us to build this service that can be trusted by developers and companies to have the support of these great people with us. It shows that we are really serious about what we are building and building a secure service and to help people better collaborate. And that's just the beginning. That's just the early days. I think even if you ask us the same questions in one year, two years from now, it's going to be completely different answers as we learn, as we integrate, as we grow, a lot of things are being able to how we do things right now, and we think a lot about this, how we can be very efficient with the size that we are and change as we grow and change the product as we grow, iterating, not having like the final product right now. Right now it's just like – But the perfect product for this exact moment. And I think it's our philosophy on building products and building the company.

[00:40:17] JM: Okay, guys. Well, thank you for coming on the show. It's been great talking.

[00:40:20] TM: Thank you.

[00:40:20] DF: That was a pleasure. Thank you very much, Jeff.

[END]