EPISODE 1514

[INTRODUCTION]

[00:00:00] ANNOUNCER: This episode is hosted by Jocelyn Byrne Houle. Jocelyn is a product manager, founder and investor. Today, she is an Operating Partner at Capital One Ventures, where she focuses on data, ML, FinTech and enterprise applications.

In the past, she has held product and technology leadership roles from multiple start-ups and for big companies like Fannie Mae and Microsoft. Follow Jocelyn on LinkedIn or on Twitter under the handle @jocelynbyrne.

[00:00:31] JBH: Traditionally, relational databases have been the foundation of financial core ledger systems. Building and operating mission critical financial ledgers at scale is complex. That complexity is compounded when you're working in a multi-tenant system.

Distributed databases do have a reputation of being more challenging to use and manage, but could they be a superior choice for running financial ledgers? Twisp is rethinking how to combine financial ledgers with a distributed database system.

[00:00:58] JBH: Jarred Ward, Founder and CTO of Twisp joins the show today to help us understand the technical decisions and trade-offs he's made and the team's made to build a scalable financial ledger for massive multitenant workloads.
One note of disclosure: Views and questions expressed in this podcast and related material are my own, or those of my guest, and do not reflect the views of Capital One Ventures or its respective affiliates."

[INTERVIEW]

[00:01:12] JBH: Hi, Jarred, welcome to Software Engineering Daily.

[00:01:14] JW: Thank you for having me. It's nice to be here.

[00:01:16] JBH: Listen, I'm really excited to talk about Twisp today. Typically, I like to start with terminology just to get on shore footing about the rest of the conversation. Let me start with outside of the world of tech. What are financial ledgers? And why do people use them?

[00:01:37] JW: Yeah. When we think about the concept of accounting, the root of the word is essentially how do we count numbers? And from an accounting perspective, we're interested in counting money. And how you start to have money or balance as accrue inside of the system is you start to go through a process of adding entries or transactions. I got a hundred dollars from someone else. And now I have a hundred dollars in my hand. I counted a hundred. Well, I get a hundred more. Another transaction occurs, and now I have two hundred dollars.

A ledger is a very simple way of us keeping track of all those transactions one after another that are occurring inside of the system. And it allows us to do some interesting things with those transactions. We can count a balance. How much is in there? How much went in? How much went out? And when we think about ledgers, we like to think of them from the concept of append only. Meaning, we're always just adding new transactions. If I got a hundred dollars and gave 100 back to you, that's a new transaction. Then I gave that hundred dollars back to you. It's quite a simple idea of just taking a set of entries of money that is moving and keeping track of it and counting it for us. I mean, at its core, it's really just that.

[00:02:49] JBH: I didn't think I'd be talking about double entry accounting again after business school, but here we are. And when I start thinking about this and got engaged a little bit with your company and understanding it, I thought, "Wow! I'm really digging in on double entry accounting and ledgers. Let's see where this leads." And it's led to some really interesting technical conversations in the context of like cloud and database architecture. But I'd love to hear from you, how did you come into this problem statement and get interested in this whole topic?

[00:03:21] JW: Yeah. I studied computer science in school. And I moved to Portland, here in Oregon, in 2006. I worked for a power utility. It was a Berkshire Hathaway company. Very conservative. They were called PacifiCorp. I actually worked on some really fun stuff there

around you know running the power grid. But there was something that wasn't very exciting about working at a power utility.

I worked in a cubicle. I had to be at my desk at a very specific time every single day. I mean –

[00:03:51] JBH: Oh, I understand developers love that, when you have to come in early.

[00:03:56] JW: Totally. We could only wear jeans on Friday. Like, that sort of place.

[00:04:00] JBH: I remember that time in the world.

[00:04:02] JW: Yeah. Around 2010, 2011, I started hearing about this company called Simple. And Simple was what we now call a neobank. The entire premise of what they were doing was banking that doesn't suck. What would it mean to build a bank for the people?

And I was super into it. I remember seeing them on Hacker News and seeing the branding and like seeing all the stuff that they were getting out there. And one day, I just saw, "Hey, Simple is moving to Portland." And I immediately applied. I was like, "Okay, if they're going to be here in the same town as me, like, I want to be a part of this company."

The first project that I worked on there was a two-factor authentication system. That doesn't matter a whole lot. We built that on top off a Twilio. But after that, the second project I was tasked with was an envelope budgeting system. What does it mean to take money and set it aside out of account balance for savings or a future vacation or something like that?

[00:05:00] JBH: That's the envelope that's like vacation savings or –

[00:05:03] JW: Exactly. Yeah. So, you take some money out of your accountant. An engineer who had been at Simple before me had actually attempted to solve or write a system that did this. And they got it up and running. And they actually ended up having to turn the feature off inside of a system. And the reason they had to turn it off is because numbers kept going out of sync, "Hey, when I add up the envelope budgets, it doesn't actually equal the amount of money that's inside of the account." And they had built what we coined now an ad hoc accounting

system. Meaning, I'm going to use the features of a database. And I'm going to write some code that's going to add numbers together and do these things.

And if you look at it, there's no ledger at all really formalized inside of the system. I came into this project. And this is kind of a theme throughout my career. These projects that had failed before. And then I'm like, "Well, I'm going to try it and see. I think we can actually make this project happen." So, we rub some ledger on it.

[00:06:03] JBH: I was going to say, that's good branding for your career. Because it already didn't work. You can't do any worse, right? You got to be able to lift it up from there.

[00:06:11] JW: That ledger – again, that very simple concept we talked about. Every time we move money in and out of account, we just append something in append-only fashion onto a ledger. And we keep track of balances by doing a **[inaudible 00:06:22]** on that ledger. Gives you a very clear history of what happened. How you exactly got into the state that you have?

And double entry, we don't have to get too much into that. But that's one step beyond a very simple ledger like I explained it. Those are those t-charts that we see in accounting where there's a debit side and a credit side. We can think of those as two running sums on both sides there.

[00:06:45] JBH: Interesting.

[00:06:45] JW: We can talk more about double entry if you want, but –

[00:06:48] JBH: You know I do. You know I love – You know I do want to talk about that. What I hear you saying is that it seems straightforward when you get started, but it gets tricky pretty quick. If you wanted to get away from like bespoke model. Like, we're going to build a ledger ourselves. Like, what makes a good ledger if you were going to get away from building it yourself in the FinTech world?

[00:07:15] JW: Yeah. I think a good ledger, there's a few things that I look from just like an operator perspective. Obviously, I want something that's highly available. If someone swipes a

card at the grocery store and they're trying to buy groceries and they are counting on that ledger system to be up to make sure that the balance is there to actually allow them to take their groceries home, we want to know that that's always up. So, that is one very important property of that underlying ledger. I want to know that it's a highly available. It's very stable.

And correctness is another thing that I am very interested in. I want to know that if I take all the entries in a ledger and add them up at the end, they better add up to the balance number that that system is reporting to me.

Now, those sounds like pretty easy problems. Well, at least the correctness kind of does, summing a set of numbers and getting – Now, when you start to expand this thing and bring it up to scale, what happens when I have millions of accounts? What happens when I have tens of thousands of transactions an hour or a minute being appended onto this ledger? Now, the scope of data inside of our system has moved beyond the scale of just being able to easily validate it. And that's where those correctness guarantees come in for us. How do I know that this thing is going to serve the correct balances?

And then performance is another important thing that we want to think about. If I'm opening up the cash app or something on my phone and it's about – it should pop up quickly. It shouldn't take 30 seconds for it to compute what my balance is.

[00:08:48] JBH: Even as you're talking, I'm realizing that what's really important, you start out with the title of, "Hey, we're going to talk about ledgers." And you think, "Okay, seems pretty straightforward. It's well-defined." But as you're talking, I'm realizing it's really this intersectionality of money, which we all care about. But also the feeling of trust that you have with your customers. And then the philosophical problem of time, which is happening in many different ways. In the multi-tenant world, time is a different problem, right? And so, it is this intersection of some really hard technical problems, some block and tackle math problems, dollar problems. But then I think the emotional side is also such an important one for – You worked at not just simple, but some other banking efforts.

[00:09:35] JW: Yeah, they were acquired by BBVA, which is a massive multinational bank. And I worked on the systems there as well.

[00:09:42] JBH: It's that trust, right? For anything new that you're asking people to do with their money, once you break it, it's hard to get it back. And so, the ledger has to be kind of perfect, right? That's interesting.

[00:09:53] JW: Yeah, you can't get it wrong. It was interesting. We had a big engineering push. If you want to think of it kind of as the vision statement for engineering, it's simple. We had two tagline things that we focused on for probably a good year or two. And those was stability. Meaning, this thing should be up and stable for our customers when they're swiping cards. And correctness. And I am not even joking that stability and correctness became the mantra of the entire engineering team. And I will tell you that those became the mantra because we weren't focusing on it well enough before. We had some problems.

[00:10:30] JBH: You learned the hard way. Yeah, that's how it goes in tech.

[00:10:32] JW: We did, yeah.

[00:10:34] JBH: It's interesting you use the word correctness. Can I ask? I would have said accuracy. Is there a reason you choose the word correctness out of curiosity? You can just say no and we can move on.

[00:10:46] JW: I mean, I think for me, it's an engineering term. Or I think of it more as an engineering term. When I think about correctness, it's something that I can attempt to prove. So, correctness, meaning I can go and look at the algorithms that are inside of my system, and prove that maybe under air conditions, or adverse conditions, or chaos conditions, that the result of this computation running is always going to output the correct result at the end.

And there's all kinds of interesting stuff with correctness. We work with a really cool company here in town in Portland, called Galois. And Galois does what they call formal verification. Or at least that's what we engaged with them for. And formal verification is this process of attempting to go in and prove that an algorithm that underlies a system, whether it's cryptography, which they do a lot of, proving that it's actually coming out and doing the correct thing on the other

end. So, I would say that accuracy is a result of having a system that can calculate those things correctly.

[00:11:49] JBH: Helpful. That's helpful. Thank you. Here's what I'm hearing, is that developers benefit, right? Because they don't have to learn accounting, right? You have the right ledger. You don't have to learn accounting. You just need something that helps you immediately connect to a proper or correct accounting, right? The FinTechs, if they were had this kind of capability, FinTechs can be more nimble. They can add more services more quickly. These are all the benefits of doing ledger correctly. How does Twisp tackle creating a correct, usable, new type of ledger?

[00:12:24] JW: Yeah. I think the perspective that we've talked about thus far is a very simple view of a ledger. And to me, that is how the world operates. There are very simple primitives. And we start to compose those primitives together and build more and more complex systems. So, we've been talking about the foundation of this system, of a ledger system. That we need to very simply be able to take amounts that are coming in through journal entries or transactions and be able to sum balances across those.

Now, the complexity of this system is going to explode once it hits the real world. And let me explain just kind of what I mean by that. One of those complexity explosions is, “Okay, now –” Like I mentioned before, we have a million accounts in here, or 5 million accounts, and here are 10 million, and all these transactions flowing in. So, we need that strong correctness guarantee down at the base to know that all those numbers coming through are accurate.

Now, let's layer more ledgers on top of each other. Now, things start to get even more crazy, where let's say I have a partner bank, or I'm integrated with Stripe, or using Lithic for card transactions, or Marqeta. Or I have a direct Visa integration. And all of a sudden, I have money just bouncing around out of all these different systems. A single card swipe can potentially hit multiple different banks, multiple different ledgers, multiple different systems. And now, I'm in this position where I'm starting to have to keep track of money moving through multiple ledgers, multiple different transaction types. I have to keep track of balances at all these different places.

Okay. So, I've painted a picture of a little bit of that complexity that can start happening. So, obviously, you want to be able to trust your ledger to be able to do that. Now, we are in a position right now wherein we think about these FinTech organizations who are building these systems.

We have a little phrase we use internally. We say that they're reinventing the ledger. A ledger at its core, it's our belief as a company that it is not something every company should have to reinvent, right? Now, inside of AWS or GCP, I can get a database and start to use it. And I don't have to build a database to use a database.

We think of the ledger in the exact same way. I should be able to say, “A ledger is a well-known concept that doesn't need to be reinvented every single time,” and gives that tool to developers who are building these systems to know, “I have a rock solid, scalable, correct system that I can build my ledgers on now.” And I don't have to know all the intricate ins and outs of exactly how its reporting those correct numbers, or how it's doing that accounting. That is there for me. And now I can start to delight my users by building a product on top of that, that is more than just like I think accounting is magical. But it's more – Like, more than the magic of accounting for our customers.

[00:15:21] JBH: And then if I use something like Twisp, can you share some examples of how would an organization get started connecting Twisp to what they already have?

[00:15:32] JW: Yeah. It's an interesting question. Because if you come into an organization, and I tell them, I have the best ledger system since sliced bread. It has all of these things that you want and need that you have an engineering team who's trying to maintain right now. And they say, “Great, I want to use it.” Or, “I have a new feature that I want to do.”

And what we start to run into is, if I suggest to you a rip and replace, great. Take your existing ad hoc ledger system, or whatever you have running right now, rip it out, and we're going to swap the engine on that thing.

[00:16:07] JBH: And we just talked about how important that is for trust and everything else that you do.

[00:16:12] JW: Yeah. Yeah.

[00:16:13] JBH: Yeah, it's nerve-racking.

[00:16:15] JW: So, when we talk about, “Okay, how can I start using Twisp?” We have a bunch of data connectors built into it that can change data capture streams off of existing database or ledger systems that you have, existing webhooks that you're maybe receiving, and start to build a shadow ledger.

We're actually replicating off something building the secondary view of it. And then we can start to validate, “Okay, do we have all of our funds flows and processes going clickbait?” Have you got running next to it? And then we start to sweat some of our systems over to Twisp. That's in the case if I have an existing system that I'm trying to switch over. If I'm Greenfield, hey, man, go read our documentation. We'll get you access. Get some API keys. And you can start building against it right away.

[00:17:15] JBH: That's interesting. I like that idea of the shadow ledger. Making your life a little easier. So, what you're saying and Twisp is that you've got a core accounting engine. So I, the developer, don't have to spend a lot of time, the deep understanding of GAAP is accounting, right? That's one thing. And then you also have this kind of – or is it an orchestration layer?

[00:17:34] JW: I think orchestration layer is – underneath the covers, we use a workflow-like language. We actually use Amazon States Language, which is the language they use inside of step functions to run workflows down inside of the ledger system itself. So yes, that orchestration engine, that's where we start to build a library of how do I make a ledger useful?

The concept of just being able to count, that's one thing. But when I painted that picture, that complexity of a full running financial system, moving funds from here to there, reconciling them, doing all these processes, if exceptions occur. That's where the orchestration stuff comes in, where I can respond to certain things happening, or run a little bit higher level logic on top of the ledger system itself. And that is our orchestration system. And we build up a library of those components.

An example would be ACH processing. Maybe I have a banking partner and I'm receiving ACH files to move money through the federal government's automated clearing house, that I'm actually have a built-in orchestration in Twisp that can generate and consume those files and automatically do that heavy lifting of translating them down into ledger entries for me.

And we think really hard about what does it mean to compose all these pieces together to create that large, complex system. So, that orchestration is really where we invest on trying to tie one-to-one to very direct use cases that our customers are trying to solve, and give them an out of the box experience that can say, “Here's this really powerful accounting core. Here are tools to orchestrate that accounting core. And here's a very easy way for you to plug and play and start to compose all these pieces together to represent your business and what you are doing with money.”

[00:19:21] JBH: Yeah, it's interesting. You've got like three components here. You've got this operational component in a multi-tenant cloud world. It's got to be really stable. It's got to be really reliable and fast. Then you've also got this issue of accounting. It's got to be correct ledger. It's got to let me continue to grow my financial services very quickly.

And then the third thing I really liked about what you're saying is this workflow component. I'm just sort of picturing lots of times you’re in just sitting at the whiteboard, right? Laying out all the inflows and outflows. Where someone could possibly call for this information coming out of your ledger? And you have to write it all down. Create events. Write individual jobs, right? That is how we used to do it. And in this case, I think what you're saying is there's just a library of functions that I can pull together to create a workflow much more efficiently?

[00:20:16] JW: Yeah. I mean, we have a number of experts who are building. I think of that always, it’s always sunny in Philadelphia image, where you’re smoking a cigarette. And there's that just huge like board with all the strings connecting. Those are like what some of these funds flow discussions are like.

So yes, we have a team of experts who are building funds flows for these well-known use cases that are repeated across every organization. We also allow our customers to build their own. So,

if you want to have your own orchestration, you have your own special sauce that you need the ledger to do, we also give our customers the ability to build them themselves and orchestrate it in that well-known fashion where we can give those transactional guarantees –

[00:20:56] JBH: That's awful. I like that. That's a really good product element. So, all of this is enabled these – and we just talked about this sort of three pods, right? Of like reliability, scale, and performance, accuracy, correctness, and then also workflow and orchestration. All of this really is powered by the kind of database native ledger that you've built on Dynamo. And I thought we could just a little bit talk about what have you built? What part is Twisp? And what part is Dynamo? And why?

[00:21:27] JW: Yeah. I don't want to say that we strictly just use Dynamo for the boring stuff. But we chose Dynamo for a few reasons. The first one is it's essentially rock solid from a reliability perspective, or close to it. Amazon will give you a five-nine SLA for a Dynamo global table. It has an extremely clear scaling story.

One thing I love about AWS is they make it very clear. Here's where you will hit limits from a scaling perspective. And by knowing those limits upfront, we can now design our system in such a way that we can take advantage of knowing those limits to scale beyond them. The idea of partitions inside of Dynamo, that we can start to think about, "Okay, if we need this operation to be able to process 100,000 transactions a second, that means we need to spread it over this many partitions. How can we start to think about the algorithms for that?" So, there's the reliability. Those really well-known limits of it. It's rock solid managed by AWS. Like I said, that five-nine SLA. And it also gives us a very – from our perspective, a very clear cost story.

Dynamo has a pay per use model. So, we know if we insert one small piece of data into the system, they are going to charge us for one small piece of data that went into the system. So, it also gives us – Like, I talked about those well-known scaling limits. We have well-known cost limits as well. So, we can calculate all these really interesting things using it as our underlying system. So, that's why we chose it.

[00:23:01] JBH: That's what it does. We love it for these reasons. Now, what doesn't it do?

[00:23:06] JW: Yeah. Dynamo's transaction interface that it has. I think they just recently bumped it up to being able to do 100 different records inside of the database. You can update them transactionally. For our system, that's not enough. We sometimes want to do all or nothing batch files into the system where maybe one cert 100,000, 500,000, a million records, and we want them all to become visible at once. You just can't do it with that.

There's a lot of eventual consistency inside of Dynamo, eventual consistency. Meaning, I write a record, and that record has an index. There's a possibility that if I attempt to read that record back out after that index was written, that it might say that it's not there, even though I just the moment before. So, there's some of those things that are inside of it. And we solved it by layering in MVCC layer on top of Dynamo.

And MVCC is essentially a consistency model that some databases have where you can actually look at the data inside of a system in a snapshot, snapshot isolated fashion. Meaning, when I start a transaction, I can see the database as of this point of time. And I can show the customer as at that point in time. And it gives us interactive transactions. The ability to say I want to begin, do all this work, and then commit all of it in the end.

Now, when I say MVCC, they're on top of Dynamo. Most people are like, "Okay. Like, why aren't you just using a different database?" Well, those things that I mentioned in the beginning are a big enough reason for us to use Dynamo that we invested in doing a small transaction layer.

Now, I don't want people to think that this is this big, huge complex thing. It's about a thousand lines of code, which compared to the rest of the system, it's extremely small. And we hired Galois here in town who I talked about without formal verification process to actually formally verify the underlying transaction model that we have inside of this.

So, we have a great deal of confidence in this small little chunk of code that is giving us the ability to use Dynamo in this transactional fashion. And also, with that said, there's a bunch of places inside of the system where we just use raw Dynamo as well. So, I hope I didn't wander too much there. But, I mean –

[00:25:20] JBH: No. You said it's a shim, this transaction layer.

[00:25:23] JW: Yeah, it's a small little shim. So, when we start a transaction, we persist the transaction object into Dynamo itself. When that transaction is done, we commit that transaction object. And everything that we write into the database inside of that transaction has a pointer back to that transaction itself.

So, when we commit, that thing is almost like the flag that says, “Okay, everything we just wrote is now completely visible. We also didn't invent this MVCC layer. There's a paper out there called Hekaton. There's a product called Hekaton. It’s a SQL server in-memory addition that Microsoft uses. There's an academic paper that describes the transaction system that they use inside of there. We have an implementation of that. And like I said, it's a thousand lines of code. Not very much. Very simple. Easy to read. It gives us those abilities to essentially use a hyper scalable system underneath the covers like Dynamo to do some of the strong transactional things with the ledger.

[00:26:19] JBH: Interesting. You made some other decisions, technical decisions. I think you decided to use Go, or Go Lang, right?

[00:26:26] JW: Yep. Yeah.

[00:26:28] JBH: Tell me a little bit about that thought process.

[00:26:31] JW: Yeah. So, Go lang is an interesting language, mostly from the perspective that is extremely – it's extremely simple. When I say simple – I don't want to start any like flame wars with this thing. But when I think about Rust, or I think about C++, or a language like Scala, there's a lot of really nice features in those languages that make them a little bit harder to learn, but make them quite a powerful language for being able to use.

Now, Go lang, on the other hand, is a very simple language. But there's a few things that it does really well. It has really strong concurrency support inside of it. Concurrency, this concept of channels inside of it. And you can atomically – these are built into the core language itself, not libraries on top of it.

And the reason that's important to us, this concurrency, is Dynamo is an interesting system, because it has those well-known scaling limits. That means when your Dynamo cluster gets very large, you can launch massive amounts of parallel work at the system. Meaning, I can ask Dynamo to do a whole, whole bunch of stuff at the exact same time. And that's how we get a lot of the performance inside of Twisp, is we launch a lot of things in parallel, concurrently at Dynamo, even for a single transaction that we're doing inside of this system.

Again, because all that formal verification that we've done, we can now also come to this understanding that, “Okay, because we have multiple partitions, I can launch multiple things in parallel at the underlying system.”

Now, Go is really good at that. And it makes it really easy to do it. So, that is one reason that we looked at Go, is just because that really strong concurrency support.

[00:28:18] JBH: Yeah, that makes sense.

[00:28:19] JW: Has great tooling around it as well. It compiles quickly.

[00:28:24] JBH: And you need to keep hiring developers. So, yeah.

[00:28:26] JW: Yeah, exactly.

[00:28:27] JBH: You talked a little bit about the substitute for using Twisp is to build it yourself, which you recommend against, right? We've made any point clear. We've made that point clear. Who else or what other technologies would be competitive or in the same space? I think we've touched briefly in the past on QLDB.

[00:28:46] JW: Yeah.

[00:28:48] JBH: That's one. But I don't want to just say it's that. Are other places or types of technologies people might look at? And why would they do that or not do that?

[00:28:58] JW: Yeah. I mean, I would say that there is also an emerging category that's coming out the – I don't like ledger as a service as a word. But like, I would say that there is an emerging ledger as a service category that we're starting to see. There're some companies out there, **[inaudible 00:29:14]**, Proper, Fragment, **[inaudible 00:29:16]** has a ledger product in it as well. So, we're definitely starting to see validation in the market that this is a category that folks are looking for.

QLDB is a really interesting one. QLDB is great, because it is truly a ledger underneath the – I mean, it's powered by ledger. That concept that I was saying, always having things on to the end of it and having a guarantee that, like, I know everything in there adds up to whatever the end is. QLDB is that as a service, in my opinion. It provides that really strong ledger guarantee underneath the covers.

Now, it's not an accounting system. There's nothing specific into it that is specific to accounting. It is really taking that concept of a ledger that I would say just like we're going to add on all these things on it. And I know that all those things are never going to change out underneath me.

So, if I use QLDB as an accounting system, I still have to build all those accounting primitives on top of QLDB. I still have to think about how to model my tables and things to store double entry accounting. And QLDB, also, it does not have those same scaling characteristics that we were talking about in the beginning.

QLDB is awesome. We use it in a number of places inside of the system to guarantee that none of the data is changing inside of Twisp, where we can actually run proofs and validation that the data is still correct inside of the system. And we think it's really good at doing that. No offense to the QLDB folks. We think of it as a Merkle tree as a service. DynamoDB, we think about Dynamo as a B-tree as a service. We think about these cloud components as part of the operating system that we're using to build Twisp.

[00:30:57] JBH: Very interesting. That's super helpful. So just switching gears a little bit. How's Twisp – Where are you with Twisp at the moment?

[00:31:03] JW: Yeah. So, I was at BBVA. Like I said, Simple was acquired. And I moved up into BBVA and worked on open platform, which is banking as a service platform. Rest in peace, when PNC purchased BBVA USA, they shut down Simple. They also shut down open platform as well.

So, I was working there. And toward the end of my tenure at BBVA open platform, this is really where these ideas for Twisp started shaking out. Because I had seen so many reinventions of this ledger. I mean, we were bringing people on to open platform. They're asking, “Well, how do I build a ledger, or get this data down into my system and keep track of it?” We're hearing this over and over and over again. And I knew from my career, how many times I built ledgers from scratch as well.

So, I guess what I'm trying to say here is that spark of Twisp, I would say, happened around 2018, 2019. And I actually started working on the implementation kind of nights and weekends around that time. And an interesting thing happened mid last year, where I actually got a proof of concept to a point where I was like, “Oh, man. This is actually viable. I'm building ledger systems on top of Dynamo with all these correctness guarantees that I've been talking about.”

[00:32:23] JBH: Just working in your like unheated Garrett with your candle.

[00:32:27] JW: Yeah, exactly. Yeah. So, I got the code to the point. And at that point, I had been doing consulting for other companies helping them build FinTech systems. And early last year, we got a team together, me and the other founders of the team. We raised a little bit of money. And we hired a couple more engineers. And we started taking this thing out into the world on the roadshow and getting it in front of people, finding out where it works, where it doesn't. How can we resonate with the problems that people are experiencing? What does it mean to get someone to actually pay for this thing and use it? And what does it mean to have them be happy and delighted about using it as well? That started last year.

Over the past year, it's been that kind of that startup life, where we build something, we get it in front of customers, as many customers as we can. Find out what's resonating. Go back to the table. Maybe make a little bit of new product. Go do it again. It's this constant iteration of refining, language, refining positioning, refining product features. Just heat seeking down on that

product market fit that everyone is looking for inside of a startup. So, we're headlong into that process.

[00:33:37] JBH: That's great. And who are the customers you're focusing on now?

[00:33:40] JW: Yeah. So, right now, we are mostly focusing on FinTech organizations. So, some examples – I can't give names.

[00:33:48] JBH: No, no. I was just curious. Just as a segment.

[00:33:50] JW: Yeah, business neobanks, card processors, other interesting – people that are doing some embedded finance stuff. And these are paying customers with contracts, actually running systems on top of Twisp. So, we're having more and more real-world experience that people building on top of it, and it feels great. And it feels really good to empower these customers to build these complex systems and give them that really solid, reliable core.

[00:34:16] JBH: I mean, that's such an amazing, very short encapsulation of your journey. And thank you for sharing that with us. I'm always curious with like really technical leaders. You're also describing the evolution of nights and weekends, doing a lot of very intense coding. All the way to now much more influencing strategic sales and business development. Tell me a little bit about that evolution and how you feel about it today as an entrepreneur?

[00:34:45] JW: Yeah. I mean, first and foremost, I think I have to thank my cofounders, because these are folks that I lean on a lot to help fill in some of these gaps that I have as a leader. I am a very a technical person. I have been a software – a high-level individual contributor software engineer my entire career. So, there's a lot that I'm learning on the fly as we go through this process.

It's one thing to build the most amazing feature complete product that does something. It's a whole other ballgame to start to talk about it and start to get a drumbeat building for people to hear about it. It's a whole other thing to be able to go into a meeting and tell a story where there's language where it resonates with someone and they say, “I really want to work with you,

because you are talking to me in a language that I underst –" that is a whole other thing from typing into a computer.

So, like I said, I lean on my cofounders a lot to help with that. We have a couple other really strong go to market folks on the team who helped me think through these things. Obviously, I'm there to contribute ideas and expertise that I have. And I'm in a constant journey of learning. And I think just as a person, like that's something that I seek out, like, I always want to be expanding my skill set, learning new things. And, frankly, being a CEO has been an extremely challenging, extremely rewarding. And I have learned a ton through this process. I would give myself a failing grade as a CEO. But I'm hoping, in a little bit, I can get myself a passing grade and –

[00:36:20] JBH: You look like a hard grader though. I will just say. This is not our first conversation. I feel like he might be a hard grader. One last – Just last question. So, you've evolved from being an individual contributor. There's ICs out there who want to start their companies. What is the one piece of advice you would give people as they continue their journey as an IC?

[00:36:41] JW: If you're continuing as an IC and trying to work up the ladder, I mean, the first thing I would say on that IC side is you want to be at an organization that recognizes that individual contributor is a separate track from management. I've worked in organizations where, to move up the IC ladder, you actually have to move into management, which becomes a whole other skill.

So, first and foremost, try and find yourself in a company that is going to recognize you as an individual contributor and has a clear ladder of making enough that. Now, climbing that ladder is a whole other thing. Obviously, you need to be an expert at your craft. Continuously learn. Stay on top of some of the trends. You don't have to learn all the new trends, but at least be conversant in what's happening in the industry out there as a whole. Invest in learning new things. Invest in learning new languages. Invest in understanding how computers work.

I'm a polyglot from the language perspective that I code in. I think you could solve problems in just about any language. Some people start to get fanatical about it. I recommend don't be

fanatical about the language that you're operating in, because there's a lot more to it than just the language.

Now, second thing is take on the hard projects. I talked about in the beginning how for – or second project at Simple, the goal system. We called it chatter. Simple had a service-oriented architecture. And they named all of their services. Funny thing. Now, I realized that there's some controversy around giving things silly names like that. I'm not condoning it. That system was called chatter.

Anyway, I would say take on those hard projects. Be fearless in a way. Because, really, how I found moving up this career ladder, for me, it's been solving hard problems that other people maybe didn't step up to the plate to do. And if you can step up to the plate to do something, you may be recognized for it.

[00:39:03] JBH: Well, I really enjoyed speaking with you today. Thank you so much for coming on the show. And looking forward to hearing more about the evolution of Twisp.

[00:39:12] JW: Well, thank you very much. I hope we can talk again.

[END]