EPISODE 1377

[INTRODUCTION]

[00:00:00] KP: The banking industry uses technology that some modern software engineers may regard as out of date or old fashioned. Entrepreneurs wanting to create products in the banking space historically faced a steep curve to build software that could integrate with established banking systems. Christopher Dean seeks to change that. He founded Treasury Prime, a company that offers a suite of API's to embed a full range of banking services, from cards, to account openings, to payments. In this episode, we talk about how that solution works behind the scenes, and what API's are available for developers.

[INTERVIEW]

[00:00:38] AH: Chris, welcome to Software Engineering Daily.

[00:00:42] CD: Well, thanks for having me. I hope this is a great interview.

[00:00:46] KP: Tell me a little bit about yourself.

[00:00:48] CD: So, I'm Christopher Dean. I am the cofounder and I act as the CEO for my company, Treasury Prime, which is a startup in San Francisco. Four-years-old. And my backstory here is that, back in the day, I was a machine learning researcher happily in academia, and for family reasons, we moved up to San Francisco, and I get sucked in by all the startups. And I've been doing that for 20 odd years now. And I've seen all the things from the like wild success. And we sold the company for a lot of money, and that was great. To the chairs being thrown across the room with people in conference rooms. I've seen everything. And I'm really happy to be at Treasury Prime where things are going great, but we're also all reasonable people.

[00:01:42] KP: What does Treasury prime do?

[00:01:45] CD: So I used to run the FinTech group at Silicon Valley Bank, which well it is. But the main bank that most startups use. So that's where I would go. Or I do. That's where I bank right now. And when I was running the FinTech group, because they bought our previous company, and I realized – Jim and I realized, my cofounder, that there is this interesting problem that so many corporations, technology companies and FinTechs want to access the US banking system, and the banks want to help them, but they don't know how. What Treasury Prime does is we're a bridge between these two worlds. We have both FinTechs as clients and banks clients, and we provide an API to connect the two. We're a pretty nerdy product. Internally, we think of ourselves as an API first product. We don't build anything because we got an API around it. And you can do everything that a bank can do.

And in fact, we run deposit operations for many of our banks. You want to open a bank account? We can do that. What kind? Any kind. Commercial, retail, trust, whatever you want, we can open it. Do you want to send money around? Wires? ACH? We can do that? Do you want to issue cards? Do you want to do bill paid? Oh, my gosh! Do you want to deal with paper checks? I'll tell you please don't. But yeah, we can do that too. So we can do all those things. And I think the central insight we have is that the bank is still an important player here, because they hold most of the economic reigns, and that it's useful to have both the bank and a technology partner for the FinTech and for the corporate clients. We have more API clients. And we're that bridge. And that's the insight we have. We have a banking network of double-digit banks to say that. And that works great for us. Pick and choose who the best FinTechs are for the best banks. And that ends up being a long-term fruitful relationship for everybody.

[00:03:38] KP: Could we zoom in on one of those use cases like opening an account? I could see where if the bank itself was your client, maybe you would help them give functionality to open an account for a customer? What sort of end user outside of a bank would want to open a bank account for their software's user?

[00:03:55] CD: Yeah, it's interesting. This is kind of the aha moment for Treasury Prime back before there was even a Treasury Prime, is that the software that a bank wants to use and the software that a FinTech wants to use is actually the same software. It's just slightly different use cases. So the easiest example for opening a bank account is I want to provide banking services to my end users. And there's neobanks which do this right now. In the US, it's actually hard to

become an officially chartered bank. But, still, there are a lot of them. There are like 5000-ish plus. Another 5000-ish credit unions. And it's hard to get one, but there's more people that are interested in doing banking than that. Like everyone knows Chime, or Aspiration, or someone like that who just open bank accounts for NGOs, and that's what most people think. But there's a lot of other use cases here. Like we have one client, my favorite example, it's a Zeebo. And what they do is they're a landlord SaaS company. So they have a SaaS product. It's a regular SaaS product. It's a web app, click on it, and it manages, helps the landlord manage their building. So it takes rent, and manage, fixing the broken window, runs background checks on people.

But as part of that, what they want to do is open a bank account for that landlord's building. They make an LLC for that building. They open a bank account. All the money goes in there. All the security deposits go in there. All the payouts for mortgages, or when you need your security deposit back comes out of there. And that's like a sample use case. And because Zeebo has an access to that bank account directly, they could do things with their app. They're just really, really useful for their end users.

It's not so much the end users want a bank account. They just want a way to run their business. And it's really easy if you have a bank account attached your business, do all sorts of fun, magic things, like lines of credit, or alerts on certain kinds of transactions, or providing a debit card so I can go to Home Depot to fix stuff, to fix the broken window. And we have an API to do all that stuff.

[00:05:59] KP: So you outlined a pretty standard set of things that lots of companies who'd want to do like opening an account, and whether or not they should ordering checks and that sort of thing. Is there an existing like standard or protocol that all banks were following and you're just using that protocol? Or how do you replicate that across so many people?

[00:06:15] CD: It is a huge pain in the ass is the answer. There's no standard. If you got side view, things are way better. But US, best economy in the world. Banking system, it's the worst of any of the First World countries. Like you want a good banking system? Go to Singapore. US is really not very good at this stuff. The banking system works. Like you can open a bank account, you get a mortgage, stuff like that. But the underlying tech is awful.

Basically, there's a handful of software companies that control this, and they don't like each other, and they compete with each other. The system is archaic. One of our banks uses a core banking system, which is the ledger. It's like the database for a bank. That was built in the 80s. And because it was built in the 80s, it works. Like you think a mainframe at work in the 80s. And yes, it still runs on the mainframe. And that's different than another one of our banks. So that's LendingClub bank, runs this really old system. And then we have another bank, we have lots of banks. Like Piermont is another example. And they run a reasonably modern system. And they're just completely different. And we have to have separate shims and layers to talk to them.

I always joke. Everyone thinks we're an API company. That's true. But our API is like this one inch deep on the whole product. There's really complicated workflow engine behind the scene with all the shims to talk to two dozen different banking systems you need to run at any one bank. There's a lot of stuff.

[00:07:45] KP: So it doesn't surprise me to hear that a bank not born in Silicon Valley would be a bit behind technologically, at least compared to a Silicon Valley company. How far behind on average is the banking industry?

[00:07:57] CD: Well, that's a hard question to answer. But generally, if my cofounder, Jim Brusstar, was here, he would say they're 20 years behind, which I think is about right. It's only really right now that many bankers are actually just getting comfortable with the Internet. And that sounds ridiculous, because it's super surprising. Most of the bankers, they think of old-fashioned banking, where you go into a teller's of. You go and talk to a teller in a branch somewhere. There are offices. You deal with people face to face. They value that relationship. That's great. But a lot of us do banking on our phone. I mean, I do. I have a great banker. And I haven't seen her and for years, I think, in-person. And that's part of the problem. And this is why so many FinTech activity exists today, because the banks have been basically pretty complacent. They're generally not good at product development. It's writ large. But who's good at product development? It's startups. They're really good at it. So finally, startups have found a way they can partner with banks effectively. Treasury Prime is part of that solution. That's really what we do for the banks. We take a bank and we turn it into a FinTech bank, one that can

partner with these people just, because it's so hard. It's so hard to partner with a startup for a bank.

[00:09:18] KP: So the banking industry, or maybe the computer industry on behalf of the bank has definitely solved some hard problems in the past. I mean, the fact that, I guess, transactions from databases were perhaps motivated by banking, and you don't want someone to overdraft an account, at least not outside of some controlled way. Could it be that these technologies are just really hard problems, and in the same way certain machine learning still uses Fortran under the hood, it makes sense to run an 80s technology?

[00:09:44] CD: Most of what people think is safe in banking is just it's a myth. The transactional technology, it usually doesn't work very well. Like, for instance, just settlements. If you think of banking as this massive distributed system, it has all the same problems a distributed system has. And it's exacerbated by the fact that the latency between the transactions is just very, very large. It's largely like in terms of days often.

So if I go and put money at one bank, it's there. And then I send it to another bank, it used ACH rails. Take three days maybe. How long until I recognize that money is landed? What notification will I get that the money's there? And the answer is it will take two or three days, depending, and you won't get any notification unless you go to the other bank and ask them. There's not a complete system here.

And part of the reason it's like this is because, like I said, there are just so many banking and fund carrying institutions in the US. It's hard to get them all to agree on how to do things. So paper checks started, that. There were clearing houses where pieces of paper went to and they send to different banks. The technology we use is very similar to that acceptance, that of sending paper round, we send files, and we FTP them, or SFTP them to the Fed, and then the Fed moves them around. And then it still takes days. It's very, very slow.

The machine learning stuff in Fortran, why do you use that? I mean, I did that. It's because Fortran is really fast. That's why. And it can be for some of the complex matrix operations. Unless you can **[inaudible 00:11:18]** use a GPU, right? They're super crazy fast. The reasons

for banks is different. It's more that's the way it's been done. And it's hard to get lots of institutions to change. That's part of the problem.

[00:11:30] KP: Do you have to face any challenges around compliance in the way you build your software?

[00:11:35] CD: Well, we do for sure. If you remember how we started, like we were at – A startup guy, a technologist. We have this company. Silicon Valley Bank bought it. They were a client of ours, and they bought our company. And I was good exit. And the thing that was interesting is we worked inside a bank and like finally could see how the sausage was being made. And it was terrible what was done there. It so manual. And SVB is a great bank. They're really well run. And so you can imagine what most banks are like.

But early on, we had some things happen where we were the backend for Stripe Atlas before they moved to BBVA. And we were opening lots of bank accounts for Stripe. And the regulators came in and said, “Where are all those bank accounts?” And what we did was say, “Okay, we have an audit trail, and we were using, like we do now, we're using an immutable database.” And so we just sent them a report of everything that ever happened to that account, from application, to KYC, to every transaction be monitored. And create this big PDF for a handful of accounts that were interested in. And that was the start for us of how we dealt with compliance. Like just having the raw data is the first step that you need. And so when we built Treasury Prime, we baked that in from the beginning.

And for us, compliance is just a core part of the product. That if you think about it, if we open accounts for the banks themselves and run transactions for the banks themselves, which we do for more than half our banks, the reason we can do that is because we're as careful as they are on the compliance side. And so when a FinTech comes to us, we say, “Look, this is work. And you have to pay attention to the rules. Because, really, at the end of the day, FinTech is responsible. But we can provide you all the tools and help to do it, and show you how to combat fraud, show you how to set the right policies. We can run some of that for you.” And we do run it for some of our clients, the FinTech protect clients. But the banks are in it themselves. But I can tell you that, at scale, most FinTechs realize they can get better margins if they do it themselves, because they know their customers better than we'll ever know them. If that makes sense.

[00:13:53] KP: Well, banking, just because it's – Financially, there's actual money being moved around, it seems like it would be a honey pot for fraud and criminal activity and people trying to hack protocols and that sort of thing. Can you speak at all to the levels of security you have to worry about?

[00:14:10] CD: Oh,100%. I mean, it's real money. And it turns out people get really mad when you lose their money. So like don't do that. Yeah, I mean, we have a joke internally. Mike Clark who's our VP of Engineering who built a lot of stuff here, he bought a lot at SVB. He has a joke that like bank-grade security isn't. So we have to be the protection here. Like our security, monitoring, and I'm not going to get into a lot of the details because I kind of just don't want people to know, but like we have people dedicated to this whole problem. Like we you know have a security officer who looks at all the stuff who runs every – We run pen tests. We do code reviews. We go through the algorithms. We have suspicious activity reports that are created internally as opposed to just the bank ones. We go through all that.

But basically, if you think about a bank, the systems at a bank aren't super secure. So what we have to do is put an armor plating around all of them through our API. We won't let anyone connect directly to the banking systems. We make them go through our API. So we do the kind of protocols you think of. When you want a REST interface to an API, we do that. And then we have direct checks after that. No transaction we do goes through without us checking it. And we can do that because a lot of us are used to dealing with very large data sets. Like we built large distributed systems in the machine learning side. And we're just used to that. So it's tricky to get this right.

But mostly, it's not about preventing the protocol from being hacked directly. It's about making the the data in the system be correct. Most of the fraud happens between two separate banks, where one of the banks is not ours, says they have X-number of dollars. And our bank tries to move that over, but that was incorrect. They don't have a hundred dollars. So someone's left holding that money. And our goals is to try to prevent that by appropriate holds, opening, and looking at transactions carefully to see if they exhibit a behavior of fraud, that sort of thing.

[00:16:18] KP: So there's some clear stories for use cases here in my mind with banking institutions, big retailers, things like that. How far down the long tail do you go? Is there a startup that would have a reason to adopt your solutions?

[00:16:31] CD: Of course on the bank side, they're all banks. And a dozen of those. And on the FinTech side, there's a couple orders of magnitude more than that. And most of them are startups. Like most of our clients, not all of them, but I would say 70%, 80% of them start by saying, "Oh I have an idea and I have some funding. But I don't have any clients yet." It's because they need a banking partner to have clients. So they'll come to us, like that landlord one, the Zeebo I mentioned, they didn't have anything before we started. They needed a banking partner. We have mostly clients like that who want to do some FinTech problem in the US that involves the movement of money and storing money.

We have another client, Mudflat. Mudflat is a great client. And they do this really old school problem of they do payments for long-haul truckers. Like you get up your long-haul trucker. You pull the truck stop. You pay for gas. It turns out that's a lot of money, because it's a truck, and gas is expensive. And if you can pay them in the most efficient way, that can be a real savings for the trucking company. And so they have clients to do that. And how would you structure that? Kind of like you think. You open accounts for every trucker, for every trucking company, and you make payments in and out of that. You present them with a nice user interface. And that's what Mudflap does. It's a great business.

So lots of things like that. All the way up to established companies who deal with – They have a lot of plumbers or something like that who need banking services. It's like that. So it's all over the map. So we go all the way down from we'll do a call, just money movement apps, flow of funds out all the way up to they're pretending to be a real bank, neobank in this case.

[00:18:13] KP: Well, Mudflap is an interesting use case. I don't know too much about them, but it wouldn't surprise me if the people that work there were experts in supply chain operations kind of stuff, but weren't necessarily web developers, API people, let alone know much about banking. Are there any, for companies like that, getting into this space and exploring these ideas? Are there any gotchas or learning curves that people have to go through in figuring out how to run their business in this setting?

[00:18:42] CD: Yeah. I mean, this is almost every – On the FinTech side, almost everybody, is that they really don't know how banks work. And I can tell you the number one complaint we get about our API is things we can't change. They're like, "How come wires work this way? How come ACH work this way? How can paper checks work that way?" It's like, "Buddy, that's just how they work. I'm sorry. It sucks. I'm with you."

But the gotchas usually are around that fraud is real and you should really work with someone like us to prevent it. It's far easier than you think just to lose even in a modest setting where you have a few thousand accounts moving money for the mall to lose a quarter of a million dollars in a day. It's really far easier than you think. So make sure you're working with someone who can help you with that.

And I'm a technical person. So I think of all these things as technical problems. But this is the case where it's actually not a technical problem so much, because periodically what you'll need to do for some of the bigger fraud systems is you have to say you have to call the bank directly. Someone has to. And say, "Hey, this happened. Can you call the destination bank, or the correspondent bank, or whoever it is, and let's pull our money back?" And that's a conversation between people. And this is something about banking that is often very surprising to folks, is that a lot of the more complicated things are very people-oriented, because that's how banks are. And you can try to hide that as much as you want. But I don't know a single FinTech who's been successful at hiding that completely. Every fintech I know at scale, including all of our clients who are bigger than nine months old, talk to the bank directly at least once a month, not once a week, because of that.

[00:20:29] KP: Interesting. I would not have guessed that.

[00:20:30] CD: Yeah, me either. Yeah. I thought we could make an API. I mean this is like years ago when we had the previous company **[inaudible 00:20:37]** that we can make an API like you're saying. Let's just make a good protocol on top of banking. And that's what we'll do. And that'll be great. And it turns out this is complicated, and partly it's because the backing systems are complicated. And so it's hard to prevent some of the problems that occur.

But also if you think about it, we rarely in banking that perfect information. And because we don't have perfect information, especially for the bigger transactions, the more complicated things, we have to rely on signals. Like if you were sending me a million dollars by sending you a million dollars, I might call you first and say, "Hey buddy, send me a million dollars. Just the right Kyle." Like, "Oh, maybe it isn't the right Kyle." And you think, "Ha! Ha! That's never happened." That's happened to our clients before where they forgot to do that and they had to pull the money back. And they only got their money back because, in a mad panic, they called the bank, they called us and we called the bank with them and we helped them through that. But you need the bank around.

[00:21:36] KP: Well, I guess it's almost comforting that I have those options. I wasn't going to bring this up, but I'm suddenly reminded of Bitcoin, which doesn't have options like that. Do you have any general thoughts on Bitcoin and blockchain technology?

[00:21:49] CD: I love the blockchain, because it's just beautiful. Like that paper is beautiful. It's just like the Bitcoin paper. It's just beautiful. The thing about blockchain in general that's hard is that, as a group, they're slowly having to learn the lessons banks have learned over the past few thousand years. In the US, we generally don't have bank runs anymore, right? We don't have these boom bust cycles, and that's because we put enough controls on the system. So as they start to spin out of control, we correct it. Most of the blockchain stuff doesn't really have that, and that's the real danger here. We have clients, crypto clients, who when they want to move their money from Eth to fiat currency, US dollars in our case, we facilitate that transaction for them. But it's the wild west out there. Most banks are I think correctly really worried about the money laundering problem. You don't want some terrorists in wherever using crypto to change in US dollars just to wash the money. And they're worried about that. And we can help them with that. But generally, you have to be extra, extra careful if you're going to deal on the crypto side.

[00:22:59] KP: Well, I'm sure you're familiar with Facebook's famous motto Move Fast and Break Things. That seems to be the way people like to build software anymore. But that's sort of terrifying when it comes to banking, that a developer could put some test transaction through and cause a big issue. It seems like testing would be especially important. I'm curious if there's any best practices you recommend to your developer users.

[00:23:22] CD: Yeah, it's a good question. Like you want to be aggressive, but you can't lose people's money. So that's bad. Losing people's money is bad. It's real money. Like if your Facebook back in the day and the site goes down, you can't see baby pictures for a couple hours or a few days. It's kind of okay. But if someone you know can't access their bank account for a while, it's often not okay. So that's illegal. So you really have to be extra, extra careful.

For us, on our side, we suggest every customer we have, every non-bank customer we have is a developer. They write code often as part of like when they're talking to us, they'll talk to the engineering staff just to figure out how things go. We tell them about our code base, which is we have a pretty simple code base. We limit the number of technologies we use just to reduce the attack surface area. But we have more unit tests and we have more integration tests than we do actual code. And like they should do that too.

Be sure that everything is going to work properly. You can't actually assume that, especially the caller, the end user is actually acting in your best interest. You have to be defensive here all the time. So that's what we do. Just have that defensive mindset.

[00:24:34] KP: So if you have a nice API, I can call that from any language. Are there any additional resources that help me if I'm specifically a Python or a Java developer or something like that?

[00:24:44] CD: We have a large community and they use everything like you're thinking, right? They use all the popular languages. They use Python and Node, whether it'd be Java. We have a user who's actually using Perl, which is like nice and old school. It's pretty funny.

[00:24:59] KP: Interesting.

[00:24:59] CD: Yeah. It's just, "Nobody use that anymore. That's awesome. Go! That's awesome." And a couple using C++. Nobody using C, interestingly. They were just too, too old. And we got that one guy. He's using Haskell, that team. Like that's also awesome, but that's hard. Generally, our API is so simple that we have some things. Like we have the open API swagger stuff, that if you want it, you can have it. Hardly, anyone uses it.

I can tell you, like the API is like a fourth iteration. It's really easy to use. You want to send money? Here's how you do it. People are used to integrating with like these REST APIs. I kind of want them to use our stuff directly, but they almost all just call the REST API. So our documentation over the years slim down to here's a curl command. That's how to do it.

We used to have, like everybody has, here's the Ruby one, here's the Python one, here's the Node one. And everyone just – We've just looked and said no one was using those. What are we doing? Maybe it'll happen again.

[00:26:03] KP: Makes sense.

[00:26:03] CD: Yeah, it is funny.

[00:26:04] KP: What's the most popular endpoint?

[00:26:07] CD: Oh, it's a transaction endpoint. The account endpoint. Like what's my balance? And then like what's happened recently? Like we have – It's common for some of our apps to call what's my balance 100 times a day as they're doing things. You can get an alert on those, but some people just don't want to do that. There's webhooks, which will tell you, "Hey, someone bounce changed, or this transaction occurred." That's the most common.

And there's an internal one that the end developer doesn't have access to, which we have a whole separate cluster for, which is managing card transactions because that's actually the biggest volume when someone swipes a card at Starbucks and wants to buy their cappuccino or whatever. There's just a lot of those. And it works how you think. A request comes in and then says, "They bought their coffee for five dollars. Is there five dollars in the account?" Say, "Yes." So that's okay. And then we have to reserve the five dollars so that you know they don't accidentally double spend it. That has the biggest traffic. And we actually have a whole separate cluster machines just to handle that traffic.

For a lot of users, this is a more complicated thing they want to do. Not the developer users. They want to you know be in the off loop or get notified of every transaction and stuff like that. And like say yes or no to certain transactions. And we can do all that. But the latency

requirements here are pretty small. You don't have that long before, because someone's standing at Starbucks swiping their card waiting for an answer. You don't have that long before you have to say yes. So you have to make sure that that stays up all the time, and that's our biggest traffic by a significant margin. I think it's 10 times what the account endpoint is.

[00:27:42] KP: Do I need to worry about that as I scale up? Let's say I launch my lightning in a bottle app and I'm going to go from a thousand to a hundred thousand users overnight. Aside from good practices about, I guess, not hammering the API and retries and stuff like that, can I expect that system to stand up?

[00:28:00] CD: Oh, yeah. Sure. Not our first rodeo. It's a big distributed system on the backend. It's like we scale up and down throughout the day as you need. Architecture is pretty simple. We have our DR, which is more complicated, disaster recovery stuff, which is more complicated. But the basics, that's really pretty easy. We run on AWS. There's Carlos Armas who runs our operation, VP of Operations and Network stuff. He set it up so they're light bulbs. The traffic gets bigger, we turn on another box. It handles the load. The parts of the system which need transactional reliability, we have just a big couple. Actually a few big Postgres databases in the back. There are orders of magnitude bigger than we need just to make sure we don't have the problem we encounter.

Carlos likes to say, who runs this stuff, he says they're light bulbs. You want more light? You just add another light bulb. And that's worked well for us over the years. And like you think, sometimes we have clients who go through – They're in Forbes or whatever and all of a sudden their traffic goes through the roof. And like we had the Forbes was big. And we had a couple in Wall Street Journal, and that was pretty big as well. We have to be able to handle those spikes, because developers don't want to worry about that.

[00:29:16] KP: So from your description, I would trust that Treasury Prime can manage my scale. But I don't necessarily trust that the historical banking infrastructure that is behind you can handle my scale. How does that work out?

[00:29:28] CD: 100%. It's smart. I can tell you, generally, with maybe one or two exceptions, when a call is made to our API, we rarely say we'll call the bank synchronously. We almost

always say, I think 99% of the time, "Great. Hold on. We'll get back to you." So someone wants to send money, for example, we could just send it right then and say hold on while we send it. But we don't. We say, "Thank you. We've received your request. We're going to batch it up," because reasons. And we're going to put this in status like pending or something. And when that's processed, usually a few milliseconds later, we'll change the status to processed and/or sent depending on which one you're at. And then you can either like pull it again later or you can like send your webbook that says that happened.

And why do we do that? Just for the reason you said. Sometimes these banking systems are really creaky, right? And you say like, "Oh, this thing that like most the time takes a couple hundred milliseconds, oh, this time it took 20 minutes." It's like, "Yeah, we just have to like put it in the queue and keep retrying and like keep monitoring that system till it's what's up." And that's why like I was saying earlier, our system is really this gigantic workflow engine to accommodate all this. I'm with you. You should not trust these banking systems and stay up under heavy traffic, because mostly they can't. But we can modify them so we can send them batch requests. And that's what we do throughout the day.

[00:31:00] KP: Well, financial transactions are one of those things I think people have very little patience for failure on. They expect your system to be up most of the time. Are you able to ever take a down time? Or if not, how do you avoid it?

[00:31:12] CD: We're in a lot of banks, and the banks will have their own schedule. So sometimes the banks will go down in the middle of the night and schedule maintenance like every – One of our banks, it's like every other weekend. And Saturday night they'll say, "Hey, we're going to do patching to our internal systems and we're going to be down for a couple hours." We can't go down ever on the API. We have to respond some reasonable way. So what we do in those cases if someone asked for their balance, we know their balance hasn't changed because the banking system is down. So we will return the last number that we knew. So essentially we cache. But it's a very smart cache that we know is we're guaranteed to be correct.

So our systems stay up. The banking systems are all sometimes go down. If you try to send money during the down times, what happens is what I was describing before. You make the

request. It goes into a pending state. And it will just sit in the pending one for a while until the bank systems come up. But from the developer's point of view, if everything's still working, it's maybe just a little slower than I would like sometimes. And if we're running the ledger, which we do sometimes, then we can just do that instantly. And that's never down.

[00:32:21] KP: Well, Chris can we talk about the tech stack? How does all this run?

[00:32:25] CD: Sure. This is not my first rodeo. Not Mike or Jim's first rodeo either. We've done this a lot. We have the simplest tech stack of all time, I think. We have nginx in the front just to manage like raw HTTP requests, HTTPS requests. And on the backend behind that, there's Clojure and Postgres. And that's it. We're big believers in Clojure. Some of us are long-time list programmers. But the immutability of Clojure is really helpful in banking apps. So it's really easy to make a mistake with non-immutable systems and money. And we just prevent that whole problem. We love the fact that our stack is so simple. We end up having one monolithic app to handle almost all the API. The auth is actually a separate piece. But just one model with a cap. And that's worked really well for us. It's made it so that it's really easy to scale the system. It's really easy to build the system. It's really easy to bring new engineers up to speed, because it actually is really simple, because there's so few moving parts. I think that's part of the reason we have such great uptime, is that there's so few moving parts that there's never that surprise where, "Uh-oh! We have this other piece that broke."

[00:33:36] KP: Well, how does the monolith function in a distributed fashion?

[00:33:40] CD: Sure. So they're all stateless. The monoliths, we run them across however many boxes on AWS. And we spin up a new one, spin up a new monolith. There's a load balancer which connects to one. If there's something that needs to be queued up, we have a queuing system built into Postgres, in our case, where we write the record and say someone else handle that. Another worker will pull that out. But it could be anybody who pulls that up. And our common infrastructure, we talk to on the frontend, is nginx sending these HTTP requests. But on the backend, it's Postgres sending messages back and forth. And we have a few Postgres instances to handle that.

Our key here is that we want as little few different kinds of technologies possible, because each one of our engineers are like genius level engineers. Everyone's just amazing. And we want them to be able to work all the way across the stack and not have to learn, like, "Oh this piece is in some other language or some other technology they don't know." So we really try to fix the system super simple.

[00:34:41] KP: Was there much debate about being a monolith versus a microservice architecture?

[00:34:46] CD: Not really. All of us have built all the different kinds of systems. And the general agreement by the early engineers was that the reason that you should do microservices is more of an organizational one. So the two groups can work independently of each other. And because of the way our code is structured, because it's all stateless, that was less of a need. It still comes up every once in a while. But we didn't really see the need for it. And the position we've had is like, "Well, when we need to split things up, let's put things up." And given the way we structured, it's pretty easy to split up. I wrote the first line of code, and that was about four years ago. And it hasn't significantly changed since then. It'll probably change eventually, but I'm not holding my breath.

[00:35:30] KP: Makes sense. Have you seen any interesting use cases or changes in behavior as a result of the pandemic?

[00:35:37] CD: Yeah. I can tell you, there's two things that have happened. The one which most people don't really realize is, of all the banks in the US, most of them are community banks. They're a couple billion dollars in deposits, which is a small-town bank. And we have noticed over the years that they're really tuned into personal relationships. You deal with the tellers. And that's how they do all the work. The pandemic showed all those banks that you're not always going to be able to come in the office. You're not always going to be able to come downtown to do your banking. And just that kind of attitude that's happened by the leadership of these small banks has really changed what banks are willing to do. And you think that wouldn't really matter, but it turns out that has really greased the wheels over the last year, year and a half, for the FinTechs to do more work with banks. And this is part of the reason you see the other side of

the problem, but there's just an explosion of FinTechs happening right now. And part of it is because you can get a deal done now because the banks are more willing.

I can tell you, we have a bank that used to do maybe four deals a year. And now they're probably going to do, I don't know, 20 or 30 this year. And that's just one of the banks. So we have other banks which do like quarters of the magnitude more than that. But it's really changed people's perception of what banking is and who the actual client is and how they should interact with each other.

[00:37:01] KP: So I don't know what the motivation is for a person or a group to start a new bank, or even if that's a common thing anymore. But do you see technology like yours as being an enabler for that? Was there a barrier previously? A technology one for preventing people from opening banks that now is gone?

[00:37:17] CD: Well, it's funny. There's two kind of banks really that we talk about. One is a regular bank, a commercially chartered bank where you're regulated by the US government with different regulatory agencies. But FDIC, or maybe your state chartered something like that. Those are hard and rare. And most of those are not started by technologists. You can see in the news periodically some person, a technologist, or some established company like Square or someone will buy a bank. And generally, when they buy a bank, they're using pretty antiquated technology. And part of the reason for that is that that is what the regulators like.

Our software, we have clients right now who did this. They started in a bank recently. They're using this not great software. It's okay. It's not great. And then they use us to wrap all those systems so that we can modernize their stack.

The other thing you can do though, which is more common, is to start a neobank. That is someone who acts like a bank, like Chime, or Ospiration, or **[inaudible 00:38:16]**, or somebody, who it's not a real bank. And that is it's not a chartered institution. It's a FinTech. And those people, they entirely use all new software. And most of them use us or someone like us to power their stack, because they want to use only the best modern software. They want transactions to be idempotent, for example, which a lot of the banking systems are not.

[00:38:42] KP: And when it comes to credit cards and things like that, I know there's a layer of chargebacks and credits being issued. People have to kind of monitor fraud and things like that. Is that a part of your services or do I do that with an auxiliary vendor on top of your platform?

[00:38:58] CD: Right. People do use us to power credit things. Credit cards users are handled by specialists. There's a number of vendors we partner with to do this. Treasury Prime is an interesting position in the world, because we have to be Switzerland. We have to be equally friends with the banks, the FinTechs with the partners. Like we have a big partnership. Like one of the ways we issue cards, I'd say probably the highest percent issuing cards through Marqeta. They are a great product. We do it with other, with FIS and Fiserv as well, I think a little bit. But mostly we partner with people. And we have to be agnostic about who we partner with.

So for us, we have a couple people doing credit cards, like Marqeta just started doing it, who we used. And it's mostly because the complexity of managing that credit line is more than most FinTechs want to do. And so it's only a few FinTechs who are willing to do it right now.

[00:39:54] KP: Well, similar question around the credit reporting agencies. What if I want to do some sort of quick background lookup or even report a delinquent account?

[00:40:03] CD: Yeah. I mean, there's different ways to do this. But whether it's credit building or dealing with KYC things, that we do all that stuff. Like if you have a bad, if you know have looked either through us or some other venue and said, "Oh no, I found this this client base that needs a credit repair." Maybe they're recent immigrants. Maybe something like that. We have clients like that. They can use us to like help build their credit. And how do they do that? Like we're Lego blocks, right? We're an API where you can build stuff like that where you can like say, "I'm going to create a bank account. I'm going to pay into it. I'm going to buy a CD with that, and then I'll pay it out later. That will build my credit score." And then we can report that with the bank to the credit reporting agencies, like Experian or whoever.

And then on the other side of that problem, when you're opening a bank account, we actually contact those people directly to say, "Tell me about this guy. Chris is trying to open a bank account. Is he legit? What's his credit score like? If it's really risky, a lot of banks a lot of people say it's just not worth it. So we do both sides of that problem.

And it is one of the odd things about the US. Maybe one of the reasons that make our banking system like not great, is that a lot of the KYC systems go through these private vendors who some of them work okay, but a lot of them don't. And we still have to like figure out who you are and how credit worthy you are based upon what a private agency will say.

[00:41:31] KP: When I think across banking products I use, I obviously have pretty typical stuff at a big name bank. I have one credit card with a particular retail outlet and some miscellaneous things here and there. Do you think that'll stay relatively consistent? Or will the financial products the average consumer is engaging with be changing over time?

[00:41:51] CD: It's a great question. The fun thing about Treasury Prime is since like we're an API company and there's lots of people who use our API in different ways. We can sit and watch things that are working just through a number of API calls and that don't. And I thought that this would coalesce. If you'd asked me five years ago I'd said, "What's going to happen? We're going to walk around with one or two do everything cards and that'll be it." Have one bank account to connect into everything.

But what's happened is the opposite. There's an explosion of this stuff. Like it seems like people have lots of cards now for different use cases. They have multiple accounts. Like Treasury Prime knows things that we can't share with other people about like how many – People have accounts with different banks. Some of them are clients. Some of them not. And it's just really interesting to watch. Like what's happening I think is since you're removing the friction from so much of the financial system, people are saying, "Sure, I can do that."

Like you're saying, I'll have this retail, this card at this retailer, but why not two or three? It's not really any more complicated for me to deal with. I have an app to deal with these. I put them on my phone. I can easily understand what's going on. I can use various services to get an overall view of my financial activity. So why not? And it works. It's very surprising to me. We have like – I have a friend who's in college. Like he's, let's say, 21 years old. He has two credit cards and like four debit cards. And it's like not that unusual.

[00:43:18] KP: Well, are there any exciting things on your roadmap you're able to share?

[00:43:23] CD: Oh, I mean, there's always, always stuff we're doing. Like there's interesting credit things we're doing right now. Mostly, Treasury Prime deals with deposit side. We have some credit products that people are starting to use that we're in alpha with. Those are super interesting. Generally, we empower this banking infrastructure. And what's happening right now is that the banks themselves are starting to become a network. So we have clients now who are doing services across banks, because one bank will have a better price than the other or a better service than the other. And so they'll do half of their – The FinTech deck will do half of their work at one bank and half of it at another bank. And we see more and more people wanting to empower these tools.

We're already best in breed at like getting started quickly and getting things out. We're known for being the fastest person to get anything in production. And that's great. I don't think that's something we probably need to worry about as much anymore, because what's happening is people are getting to market quickly, and then they want to do extra complicated things. And so for us, it's the credit side and doing things using our cross-bank network.

[00:44:34] KP: Well, for a developer that wants to take a look at the API docs, learn a little bit more about the features we talked about and the ones we didn't and see what Treasury Prime has to offer, where's the best place for them to go?

[00:44:45] CD: Dashboard.treasuryprime.com. You can go to treasuryprime.com marketing site and click on the developer links. Yeah, all our docs are open. They're all right there. The alpha docs **[inaudible 00:44:55].** But the stuff in GA is all there. You can sign up for a Sandbox account. You can move fake money around. We'll create a couple bank accounts. So I think it's fake ten thousand dollars in it. And you can move things around. And that's easy to see. I do a demo where we just use Curl to create some accounts move some money around, create some cards, get the fake card number of the pan out. And for a lot of people, that's the aha moment, "Oh, wait. It's that easy?" "Yeah, it's that easy." Don't worry about the complexity. We got you. Like the API make sure that you can only do things which are allowed, and possible, and legal, right? We prevent you from doing all the illegal things.

[00:45:39] KP: So there's a great developer story. What about for a BI analytics professional who wants to look at a big batch of transactions?

[00:45:46] CD: Well, we have a product there. It's a harder story. Like we have a data warehouse product, and we started it for the banks. But it turns out some of the FinTechs like it too. So we will give you a clean annotated feed of every transaction, every action that's been taken on everything you've ever touched. And we do that by exporting it to a data warehouse. In our case, it's a Postgres database. And you can access that and do whatever you want. We can show you how to put tooling on top of that.

But the mind-blowing thing is that it's kind of crazy how much activity there is in a single account. And being able to model that and view it is really, really eye-opening. You can get these incredible views of your customers you weren't aware of before. Like that's the advantage of using an API for all this stuff, is that we just write down everything you do. And we have an audit trail for everything. Like I said, I'm using an immutable database. So it's like you can look at a snapshot whenever you want. And that's really insightful. We have one of our clients who like finally said, "Yeah, I guess we're going to use the data warehousing product now."

So we turned it on, and like we didn't hear from him for a couple weeks, which is unusual. And then the CEO called me directly and said, "I had no idea that 10% of my clients were this busy." It's like, "Well, yeah." And yeah, I mean, that's how it works. Most of these users are power law. They have like super, super busy people and then like less busy people. And like, yeah, we should focus on the busiest people. It's like, yeah, I agree.

[00:47:19] KP: Well, good insight. Yeah. Well, Chris, thank you so much for taking the time to come on Software Engineering Daily.

[00:47:24] CD: Thank you very much. I appreciate your taking the time. And I will hope that other software engineers had a interesting listen too. Thanks so much.

[00:47:37] KP: Absolutely.

[END]

