**Part 1 - Your Startup**

1. What is Startup's main product / service?

3:03 Come on, come on. Our startup is Startup 13. We work with the prepayment of receivables. We discovered that there was an option to prepay credit card receivables and the amounts charged for this operation were absurd.

Studying the market, we discovered another inefficiency that was not reasonable. Large companies with excess cash are reimbursed at 100% of CDI simultaneously as their suppliers discount these large companies' securities in factories or similar with average rates of 2% per month. We were talking about a spread of 1.8% or more than 10x the annual CDI. In one month, the financial intermediary earned 10x what the big company earned in cash applied. And so Startup 13 was born, with the proposal to disintermediate this relationship, allowing the giant company to pay directly to its supplier at a discount, eliminating the financial sector from the relationship and maintaining the margin in the productive sector. For the large company, the cash above the CDI's profitability, and the small one a cheaper option of working capital.

2. When did your startup establish itself?

3:12 It was established in 2016.

3. Where is your startup located?

3:20 am She is based in Salvador. Currently, I stay in São Paulo, so we have some people outside of Salvador as well.

4. What is your role?

03:31 Currently, I work as Startup 13's CTO.

5. What kind of ecosystem are you currently working on?

03:35 Anticipating is a startup in the Fintech sector. Our goal is to anticipate receivables in the relationship between the drawee and the assignor.

6. How many employees do you currently have?

03:51 today we have a total of 20 employees, collaborators.

7. What is the composition of your team?

3:54 The team is made up of a small marketing team. In reality, it is a person; we have a sales team, two, three people; we have a service support team, this also counting on more external hires, an outsourced company; then, the support team is composed of two people; and two more people working in direct sales, what we call the transferor sale. Then, there is a sale to the drawee and sale proceeding. In the development team, we have a separation into three teams. We currently have an integration team with two people, a development person and a deployment person responsible for controlling the deployment life cycle. In terms of integration, we work with SAP, ORACLE, and Totus integration. The second team is a data team that works with BI analysis, which works with Data lake, which works with all the algorithms' intelligence part. We have the third development team, which we call web support, which are the applications that have a direct impact on the end customer. This team works with these products' delivery. We have a separation: two people focused 100% front-end, front-end developers. We have a UX and UI person, Marcelo. And we have two back-end developers. We were doing back-end work, DevOps, anyway. All the teams have some daily demands, and they pull the activities that should be done from our backlog. Each team controls its score. They organize themselves there. They are also responsible for the entire application lifecycle after definition, development, and support.

8. Do you proactively participate in product life cycles, such as production, testing, and launch? If so, to what extent?

7:13 So the last one is if I participate actively. Yes, I do. Most products are going through a different moment now that we are redoing some things. So we have since a product that already existed and that we redid, due to technical problems that existed, separation of the teams until the conception of a new product. We are saying that it is starting from scratch, from the analysis part, right there with some customers, with customers we have, the banks, until the final implementation, delivery, testing, and launch. Today I actively participate in all stages, I still experience, and I believe that I will stay in all these stages in the coming months. But we have already been making a baton pass, in the sense that some products already have a life cycle, and then the team itself now takes up the activities of the backlog and evolves.

**Part 2 - Software Engineering Practices**

1. What software development practices, tools are you using? Briefly describe how?

9:54 Come on. We were talking a bit about development practices. We do not use any formal development practices. We, as I told you, the teams are mixed. The figure of Product Owner within the team, a team that works with Kanban methodology, another team that works with Scrum methodology, each team today, talking about development, works with a different method, which was the one that was best suited there. So, today what we have as a model today is SCRUM. So, there is a backlog definition process that is not yet clear within the company. We are still improving it due to the improvements we have made to the old platform. But the selected activities go there for a daily meeting that takes place every Thursday. We score, do the whole process to achieve these activities, and then go to our backlog to go to our sprint. We use a 2 (two) week sprint. And it is a process that we have been maturing.

11:53 Then, talking a bit of tool, we use n tools, the agent has a greater focus on Asana to control activities, we used Jira before. We ended up switching to Asana. There is a platform for us to test. There are some easier things there, so we use Asana, the whole team uses it. Including the sales and marketing people. We use Hubspot. We use the Pipe Drive for sale. We use Zendesk. We use Github a lot. Most of the documentation, all of the architecture documentation, drawing we use Draw.io, Miro for Event Storming, or the architecture of an application when the product starts or doing something new. So there are N tools, we can talk. And some tools range from testing static code analysis, such as Sonar Cloud. I won't be able to list them all, but the purpose of all these tools there facilitates communication and improves the quality of deliveries. Then you enter a universe of Cloud that we use: the AWS tools, tools for both CI and CD, and automation, such as DataDog to do all part of observability, log monitoring, and support.

2. What are the most important quality attributes (UX, performance, security, reusability) for your current products?

14:33 quality attributes that we consider most important to me, I think that in the first place we consider UX. I would be putting it this way for applications that we have as the final public, our final client: the assignor, we need to have a very refined UX job. And it is a job that we have been doing. And we launched a new platform just to solve some *gaps*that existed. And this platform has also been bringing new security requirements, including meeting some Data Protection Law needs. Thirdly, I think we focus on performance, but this is something that we are refining post-launch. We have been taking care of a performance analysis throughout the development. But our development practice process includes, for example, *code review*. In the *code review*process, we do a performance and security analysis.

3. What testing practices do you adopt to validate and verify your product / service quality?

16:06 Test practice that we adopt. We currently use it, not 100%. This has been maturing in the company, plus every API that comes out, at least what we call edge security, 100% is tested there. And the *endpoints*that are made available. In some applications, applications such as authentication and applications such as anticipating critical applications, we do an end-to-end test process with a robot. A DataDog robot runs every five minutes between different endpoints, between Brazil-United States points, and does a Selenium test to check if the platform is performing. And we monitor the performance indicators on the platform. Both by the AWS platform and by the DataDog platform.

4. How much did you invest in testing activities?

17:41 , item four, we understand that an API or a service does not come up without a minimum test requirement. This is already barred from static analysis, and in the CI and CD process, integration continues. Today, if the developer opens a new PR there, it is right to *merge*with one of the *branches*. We do a separate *branch*: *master*, *dev ,* and *send box*. These are the environments that we have. When he opens PR for any of these *branches*there, the Sonar Cloud robot already validates the criteria that we call *quality gates*. If that does not pass the quality *gates*, it is automatically rejected. And the developer has to go back and fix it. Today we have *quality gates*that evaluate the amount of code that is being repeated, duplication of code; we have the number of tests that are being repeated, and we also have the test *coverage* and *code coverage*. So today, nobody can upload any code that doesn't have a minimum required for testing. These quality *gate*indicators that we use are evaluated every month. Because we come from a platform that didn't have these practices, and by the time we started to migrate, we've already migrated based on security, security analysis, and code quality issues that are evaluated via static analysis of the Sonar Cloud that then it goes through a *code review*process. How much did we invest? Only today, the front-end part, we do an end-to-end test, and we have another test engine there that also validates at the time of the commit, on the machine, when the developer does the commit.

22:17 When we are going to get an activity there to do, I have to make an API with a get method that will return the cities' list. Then, who will take it determines the number of points already imagining that there is the test? There is no such possibility of not considering testing.

24:03 And then we have a practice process, a code *review*that he aims to get what the quality gates do. I cannot require the developer to do an individual verification is a process to define, and we are improving it continuously. In case of problems, we can only go up to the next PR when that problem has a test covering it. It is a more edge test that aims to get the main features. So I will have to check that it is hitting the bank if I did a get and it got to the database. Test it layer by layer, we get to do it, but we don't do it anymore. We are going in a direction where the end-to-end test is more interesting than the layer-layer test.

Even though I know that ... but this is at the developer's discretion, so, for example, I was a developer, I constantly upload something, I always upload a feature starting with tests in a TDD process. But this is not required. It is not required. We don't oblige. "Ah, you have to test all the layers." He is doing a process to make people aware that the sooner it fails, the better our process, *continuous deployment*. There is a process for us. Today we work with three environments. First, there is a process for you to upload something. I want to upload a new API. How do I do it? I put it into production. Until it reaches production, there is a process. And the process consists of: your code has to be self-contained enough for us to press the button, and it will run in the environment.

So the developer has full access to the development environment. So does he have to do it? If we think about an application here, a *lick*, he has to develop a *lick*, which is a function *as a service*, and he has to develop an IPI gateway that will connect to that lick. Then along with this IPI gateway, and to allow it to be possible for me to execute these scripts in all environments, it needs to climb the Cloud formation. Today, there is no way to pass something into production without a Cloud formation, an X script that defines what you are uploading. Suppose he needs a database, all the database configuration he needs, and all the *policies*to create security. This will pass to us. We analyze it in the first moment in the sandbox. So, before moving on to production, we run sandbox. So, if it was run there in a sandbox, it will also run in prod (production). The goal is when it gets to prod. It will push a button and work. This is in some cases when we do not have the CD running.

But he needs to follow a checklist to be able to upload such an API into production. And there has to be a reason too, why he wants to make this available in production. It has to do with a backlog product. It is a new product that has already been discussed, it already has architecture, so it follows this process of promoting it step by step. When we are talking about the complete application, what this application directly impacts the client, we have an internal process of evaluating one person, which he calls QA.

30:16 So when it goes to the sandbox, the first QA done is done, another dev does it. So, you developed. You said Michael: "it's OK, I've run dev, I've promoted to the sandbox, I wanted to start a QA. So, we have to check if someone made a mistake here because I'm not going to pass this on to the QA team without first validating it. We don't have such a mature team. So, these checks end up having to exist. So, go over there and do an initial check, check if you're authenticating, how many calls are being made here to the server on this front end. The applications are *singles page applications*.

Having something to be corrected, it goes back. And it doesn't pass to the QA staff. We do a test process saying, I tested the screen, the behavior was such, when you clicked such button, the behavior was such, mark everything cute and give it back to the team that developed it, until you adjust and mark it: it's ready, it's prepared, it's ready. After we have this first validation, which usually lasts one afternoon, we take the entire system and validate it. So, we pass it on to the team that will sell or defend that product. So, this group will do OK.

We are talking about people doing support services. We are talking about people who are doing support to sell the passive or the drawee. Once approved, we promote its production. We are talking about a whole product cycle. A new feature depends a lot on the feature. Because we make constant updates, so it depends a lot on the functionality. For example, it was a bug that we caught here. We already created a test, so beautiful. It ran in dev. It ran in the sandbox. Theoretically, it can run in prod. Theoretically, it is not to give an error. So, we go there and check, did it go into prod? Did this formation go through in prod? Then take the master, and you can execute there too. I got into it here, yes, that's right, so it's good to take it too. So, they are micro-updates, but there are a few more steps to give a final OK when it is a product.

5. How do you document your product at different stages of development and testing?

21:02 How do we document? Most of the things we document or in Miror, more definition of architecture, and in Github. So the whole repository of a product from us, we work with microservices, but there is a microservice there that sends notifications. So, in this micro-service for sending messages, there is express documentation of how that service works. So, if you are going to use that service, what do you need to know to integrate with that service? The documentation today for each of the microservices is in the microservice itself. In the master branch, you will have the documentation. It's how we control it today. The documentation is a .txt, a Readme that we commit to. And there is an image of how the architecture we made in Draw.io is, put it there, but what you need, what are the APIs available there, and the purpose of that service.

**Part 3 - Technical and Dynamic Debt**

1. How aware are you of TD in your startup?

34:27 We monitor technical debts, and we promote technical debts. So, what happens, when there is a new definition? For example, I usually advise the problems that we should encounter in that approach that is being made. We cannot constantly evolve. So, I'm going to give you an example of an adaptation of an angula team. The part was angula. The part was *React*. We decided to embrace *React*.  *React*does not have a very clear definition of architecture. Unlike Angula, which comes a little more, it already defines it. So, for us to do the unit tests, of the modules, of our components, I saw that the team was having a particular difficulty, and then I pointed out that it was generating a technical debt for us. So, at the first moment when we were building, the goal was to reduce the technical debt. We pull something from the *back-end*and the *front-end in all sprints from the moment we started to deliver*. So, we do not advance any sprint without advancing in the reduction of technical debt. So sonar helps people because we cannot accumulate a security breach, *vulnerabilities, or issues.*

36:47 You can't accumulate it because we aim, for example, at zero security. So, you can't have any. And you can only have 20% duplication of the code in the total of the whole project. So, you can't go beyond that. We have to reduce those that already existed. This creates a particular obstacle because sometimes the guy takes a code to make a small change. He may not be able to upload that code. So, patience. This is part of the game. Every sprint, we choose a technical debt, and who decides this technical debt? It depends on two factors: the first is how this sprint is doing, which is the number of points that we can solve versus the number of issues the sprint already has. And the size of the technical debt. And then, if it's too big and it doesn't fit in the sprint, or we postpone it. Or we slice it into smaller technical debts. And it pays off little by little. This is the approach we have been taking, and the team has been feeling the evolution.

When working on a microcomponent platform, this also creates another problem: the deployment. Because we have an application that is already working, but it has technical debt. For example, in language management, we have this problem, we have two products running with this debt resolved, and others are not. We have been doing a service catalog, saying which service we have, which version the package is in if it is up to date, if it has observability if it has a language. We will map it. And with each sprint, we get one, with the goal of not always leaving it behind. But it is, I confess, that the crowd is constantly generating some other new technical debt. So, the goal is not to leave, even if that will compromise the sprint. So, we prefer to slice and get something to solve than to pay nothing to solve.

2. What is your perception of TD?

40:17 So, we are fully aware that there is technical debt and that we have to pay it to move forward.

3. How do you deal with DT? Do you ignore TD? Do you accept and manage TD? You avoid TD

40:27 We manage. Our goal is always ... there are legacy services. So, legacy services always have a huge amount of technical debt. And we can't redo it overnight. Today we consider that we have a new platform and an old platform. We support the old platform, and the new platform is what I told you, that we do not let go. So, old, unfortunately, we can't support everything. So, what we are doing is that we are slicing and gradually, we are picking up and bringing pieces of this old application to this new application, which has these services, which validate, which have a code review. Even so, technical debts within the new platform still exist. And we will control and bring. So, I usually talk about the boys that every time I fire many people, we have to include the test, have technical debt, or not even more because he is bothering the team so much that he prevents people from moving forward.

4. How did you deal with TD in the early stages compared to now that you are in the growth phase?

45:31 In fact, the first thing we created was a process and greater team awareness. This is the first point. The second point is, in fact, before, we didn't look. I still had a very junior team. That was the process to develop and rise. There were no environments, for example. The developer did it on his machine, tested it there, the other looked and went up. It was a problem, go up again, go up again. So, before, there was no control. We had no control. We still need to improve a lot, but today we already stop a lot. Even earlier, we took things that compromised the product and the cost of hosting our products. This raises a lot of expenses. I end up having to scale the servers vertically and horizontally. And I end up having applications that need a lot of resources to run. We already understood that. Today we have greater control. It is not ideal yet. We are still in the process of structuring.

5. Did you change the startup's focus during the transition from the initial to the growth phases? Did any of the situations occur?

1. **Zoom-in**. Has a single characteristic of a product become the complete product itself? - If so, explain the role of DT?
2. **Zoom-out.**Has the entire product become a single feature of a much larger product, mainly because the original product is insufficient to meet the customer's needs? - If so, explain the role of DT?
3. **Customer segment**. When trying to solve the right problem, did your startup discover a different customer segment than initially planned? - If so, explain the role of DT?
4. **Customer need. Has your startup realized that the problem you are trying to solve is not very important**for customers and has discovered other related issues that are more important? - If so, explain the role of DT?
5. **Platform pivot.**Has your app become your support platform or vice versa? - If so, explain the role of DT?

f. **Business Architecture**. Did your startup change the business architecture, for example, aiming at low volume, high margin, instead of focusing on the mass market? - If so, explain the role of DT?

g. **Capture value.**Has your startup changed the way/method of capturing value (monetizing)? - If so, explain the role of DT?

H. **Growth engine.**Has your startup made significant changes to its growth strategy to seek rapid and more profitable growth? - If so, explain the role of DT?

i. **Channel pivot.**Has your startup identified a more effective way to reach your customers than the previous one? - If so, explain the role of DT?

j. **Technology Pivot**. Your startup delivered the same solution using completely different technology - If so, explain DT's role?

48:27 In fact, the V1 model didn't make sense to the market. We didn't even sell it. But when we did the first version and tried to put it on the market, my partner already saw that it didn't make sense the way it was. It was at that moment that I even entered. At that moment, I entered society to make this new product, which is the same as it is today. Only today is a different phase of development. It's the same product. Sorry, the goal is the same as anticipating receivables. Today, however, we expect receivables with our cash. And we have been working for a few months now to prepay some money with third parties, which are partner banks.

We pivoted the idea, maybe in the first six months. A team made the first pock, and then Camillo saw that it didn't make much sense, pivoted the idea that makes sense. The first idea just didn't work. We started to develop in 2015 more or less, in 2016, more or less, 2017, we began to operate. With some improvements still in focus now. Before, we used our capital, and now we use the drawee's capital and the partner banks' capital. There is a big difference there in this process.

6. How would you explain DT's role in your Startup pivot scenarios in one or two sentences?

51:18 In the scenario of pivoting the idea over there, I think it's not that long ago. Because we threw away what we had done, I believe that the role of technical debt made more sense when you left the initial level, in pock, in 2016, when you are together with the client doing a proof of concept. And the second moment is when you receive an investment. That was the case of Startup 13, which received a contribution and was later sold. When we received the contribution there at that first moment, the focus was on technology. So, the technical debt in this scenario is significant. But it affected the negative direction. We did a lot of gambling in the past, and now as we are here, we can't get very far, deal-breaker.

At that time, startups in the initial phase, we don't have a lot of resources. It was Camilo and me. I already had a solid foundation of architecture and software engineering, and we did the first version there. In my case, as a partner, I left the company. That was from 2016 to 2019. So, when I left the company, we no longer had an architect figure in there. We had no option then. And what we could do was hire junior developers. So, inevitably, without having the architect's figure in there or someone to accompany the development. There is a crucial moment there, of uncertainty, that you don't know if it will work. And you can pivot there. So, in this moment of proving the idea there, I think that fatally, technical debts will exist, they will accumulate.

We are at the stage we are in now, in which we need cost reduction, agility in deliveries, and security in deliveries mainly. So, the applications that we are developing today are applications that we understand can be scalable. And the application we had before, we were having problems. But I know that we spent time trying to validate the idea. For a year, we have been paying technical debt, redoing, redoing. The cost is very high. But then we have to put it on the scale that we had no money to hire at that time. We didn't know if the idea was going to succeed. Does it feel like this for the market? At the moment that we started to have the contribution, damn it.

56:23 We had a lot of junior developers. There was no point in wanting to scale within such an architecture. There's no way I wouldn't go forward. But it was either giving them total freedom to write the way they think it was supposed to be done, and we validate the idea. Or we would be stagnant. The learning curve when you take a student from college is practically ... it is enormous... because there is a huge gap between the market and the academy. You arrive with architecture, with an architecture proposal with three environments that bring technologies that are being used by big companies like Google, Amazon, Microsoft, and want to put in your startup in T0, the probability of failure is very high. Today the best school is to have the problem and look for solutions.