Interview mit Daniel Bogdoll, CEO bei SAYM

Leo: Erstmal, um das nochmal festzuhalten zu dir als Person. Du hast ja ein eigenes Unternehmen, SAYM. Magst du einmal kurz ein bisschen beschreiben, was euer Unternehmen macht oder was euer Produkt so anbietet?

Daniel: Klar. SAYM ist eine Plattform für Fahrgemeinschaften. Das ist eine App im jetzigen Stand, Android und iOS, und wir verkaufen das Firmen als Software-as-a-Service Paket, sodass die das den Mitarbeitern gratis anbieten können. Wenn die Firmen zu klein sind, um das alleine sinnvoll anzuwenden, dann auch gerne firmenübergreifend. Der Kernunterschied zu bisherigen Fahrgemeinschaften ist, dass wir nicht einzelne Leute verbinden, zu Zweier- oder Dreier-Fahrgemeinschaften, sondern dass wir einen großen Pool an Leuten aufbauen, die in ähnliche Richtungen fahren und dass man aus all diesen Leuten immer wieder wählen kann, mit wem man zusammen fahren will, auch zu verschiedenen Uhrzeiten, sodass die Zuverlässigkeit von Fahrgemeinschaften deutlich steigt. Und wenn mal wirklich etwas schiefgeht, dann haben wir Partner, die andere Mobilitätsoptionen anbieten, zum Beispiel Car- oder Bikesharing, auf die man dann zurückgreifen kann.

Leo: Ach cool.

Daniel: Also wir legen einen großen Fokus auf die Zuverlässigkeit des Systems.

Leo: Es ist ja auch wichtig, wenn man damit täglich zur Arbeit fährt.

Daniel: Genau.

Leo: Cool. Dann, magst du vielleicht einmal kurz den Tech Stack von euch aufschlüsseln, also mit welchen Frameworks arbeitet ihr?

Daniel: Klar. Also im Backend sind es nur AWS-Services, also DynamoDB und alles was es dazu gibt, also Amplify, Lambda Functions, Cognito und Upsync. Das benutzen wir alles und dann im Frontend React Native mit Expo und Redux als State Management, und auf der Component-Seite fast nur native Komponenten oder React Native Elements, wenig selber geschrieben.

Leo: Okay, habt ihr auch ein Web-Interface?

Daniel: Wir haben ein Web-Interface für die reine Registrierung, also wir machen es so, die Firmenkunden, ich kann dir das auch gerne zeigen, wenn du willst...

Leo: Ja, gerne.

Daniel: Die Firmenkunden kriegen hier eine Website, wo sie so ein bisschen Informationen über uns noch finden, also welche Firmen machen jetzt mit in unserer Community, was ist eigentlich genau der Ansatz, wie funktioniert es, wie stellen wir sicher, dass es zuverlässig ist und so generelle Vorteile. Dann kann man sich hier anmelden und hat nur die Registrierungsmaske noch online verfügbar.

Leo: Ah okay, und sobald es dann um den ganzen Rest geht, das passiert dann alles in der App?

Daniel: Genau. Also der Gedanke dahinter ist, man hat ja nichts von der App, wenn man noch keine Matches hat, das heißt, wir wollen die Reibung der Registrierung so niedrig wie möglich halten. Also eine Expo-App ist ja direkt irgendwie 30 MB groß, die will man nicht sofort runterladen, wenn die einem gar nichts bringt. Und so eine Website kann man in einem Wiki, in einer E-Mail... Das ist super accessible, auch von der Arbeit aus und sobald man registriert wird, wird bei uns eine Lambda-Funktion getriggert, die checkt gegen alle vorhandenen Registrierungen ob es schon Leute gibt, die eine ähnliche Route haben und sobald wir einen oder mehrere Matches gefunden haben, benachrichtigen wir beide Personen und ab dann braucht man auch die App.

Leo: Okay.

Daniel: Genau. Das ist entstanden über React Native Web, glaube ich. Also wir haben den gleichen Registrierungsprozess nochmal in der App, mit 90% oder so gesharter Codebase.

Leo: Und funktioniert das gut, so mit dem Sharen der Codebase?

Daniel: Ich glaube am Anfang war das so ein bisschen ein Struggle, die Komponenten zu finden oder zu ersetzen, die nicht auf beidem laufen, weil nicht jede React Native Component ist kompatibel mit React Native for Web. Das heißt, wir haben schon leichte Unterschiede, aber ansonsten ging es relativ schnell. Also ich habe es nicht implementiert, der Philipp hat's gemacht. Ich glaube, nach einem Tag oder so lief das meiste.

Leo: Nice. Du hast jetzt gesagt, dass ihr hauptsächlich die React Native Komponenten benutzt, also dann wahrscheinlich die nativen iOS- und Android-Komponenten, ne?

Daniel: Genau.

Leo: Haltet ihr euch dann da auch an das Design System von iOS und Android jeweils oder habt ihr ein eigenes aufgestellt?

Daniel: Wir haben kein Design System aufgestellt, wir haben uns generell ein bisschen überlegt, wie es aussehen soll, und es ist aber denke ich eher Android-beeinflusst, weil die meisten von unseren Entwicklern haben Android-Geräte. Aber wir sind noch nicht so weit, dass wir einen Fokus auf ein Design System gelegt hätten.

Leo: Okay.

Daniel: Also wir haben ein Farbschema, und jeder Button sieht gleich aus. Wir haben quasi solche Designs global definiert aber Buttongrößen und so, das ist Screen-abhängig.

Leo: Okay, cool. Und siehst du ansonsten noch bei diesen Komponenten, die ihr gerade nutzt, bestimmte Vor- oder Nachteile? Also gibt's da was, das dir besonders positiv rüberkommt oder wo du denkst, da könnte man vielleicht noch was verbessern?

Daniel: Also, ich habe ja React selber erst vor einer Weile gelernt und mir gefällt das modulare System schon sehr gut. ich bin auch ein bisschen im Frontend mit aktiv, weil alles was im backend ist verstehe ich nicht, da will ich die Hauptarbeitszeit von meinen anderen Entwicklern haben und ich mach da ein bisschen, was ich kann. Die Wiederverwendbarkeit ist schon sehr nice. Also, wir haben noch nicht quasi einmal den ganzen Code gereviewed oder ähnliches, um eine Generalisierung jetzt perfekt zu machen, aber ohne copy-pasten zu müssen kann man schon relativ viel Komponenten einfach nehmen, die irgendwo anders definiert sind. Wir haben auch so... Alle Components, die wir nicht nehmen konnten, das sind halt nicht so viele, dabei haben wir schon ein eine interne Component Library, dass du einfach weißt wo die sind, du kannst darauf zugreifen und dann hat dein Button halt immer den gleichen Shadow oder solche Sachen.

Leo: Ah, cool. Und für diese Component Library, nutzt ihr da Storybook oder so oder habt ihr euch das selber aufgebaut?

Daniel: Das ist einfach ein Ordner meine ich.

Leo: Einfach ein Ordner, okay.

Daniel: Wo die ganzen React Komponenten drin liegen, genau. Das gefällt mir gut, dass die Wiederverwendbarkeit da ist, und ich sehe auch das Potential, das mit relativ wenig Overhead zu einem System zu kriegen, wo nur noch diese standardisierten Elemente verwendet werden. Das heißt, ich glaube, dass man es da schnell hinbekommt, irgendwie anstatt ein 30-seitiges Guideline-PDF aufzustellen, wie es alles aussehen muss, nur Komponenten benutzen muss. Und dann sollten alle Screens irgendwie in einer Art ähnlich aussehen.

Leo: Ja, ich denke, das geht auch ganz gut mit React, mit den Komponenten. So, jetzt schaue ich nochmal ganz kurz, ob ich etwas vergessen hab. Ja, ich habe mir so einen groben Leitfaden aufgeschrieben, einfach... Obwohl ich das Interview auch relativ offen halten möchte, eigentlich.

Daniel: Alles gut. Ich habe gelernt, das ist ein semistrukturiertes Interview.

Leo: Richtig, genau. Habe ich heute auch gelernt. Genau, dann würde ich dir einmal ganz kurz die Component Sprints vorstellen. Du hast ja ein bisschen davon schon mitbekommen, dadurch, dass du ja auch bei einem der Usertests teilgenommen hast, danke dafür nochmal.

Daniel: Gerne.

Leo: Wir haben ja den Background, dass wir bei Crisp Design Sprints machen und haben uns da überlegt: Was könnte denn nach dem Design Sprint der nächste logische Schritt sein? Du hast ja als Output von einem Design Sprint ein Prototyp, der validiert ist, von Usern. Den könnte man vielleicht nochmal iterieren, aber von diesem Prototyp möchte man ja dann eigentlich als nächsten Schritt wahrscheinlich ein MVP bauen und gucken, ob der wirklich auf dem Markt ankommt. Component Sprint sind dann ein Schritt, den ich ganz gerne dazwischen machen würde, in dem anhand des Prototyps geguckt wird, aus welchen Komponenten besteht dieser Prototyp, welche Komponenten davon könnte man vielleicht schon mal schreiben? Sodass die Entwickler, die das MVP entwickeln, sich nicht gar nicht mehr wirklich um das Design kümmern müssen, sondern die einfach schauen können, dass sie die Business Logik implementieren und mit fertigen Komponenten schon arbeiten können. Sagt dir Atomic Design was?

Daniel: Ist das eine Bibliothek oder ein Konzept?

Leo: Das ist ein Konzept, von Brad Frost entwickelt. Ich schau mal, ob es da eine schöne Grafik zu gibt... Ja genau, was er macht beim Atomic Design ist er sagt,

wenn du Komponenten hast, kannst du die fast ein bisschen organisieren wie ein physikalisches System. Du hast Atome, Atome sind die kleinsten Komponenten, die es gibt, also das wäre zum Beispiel ein Eingabefeld oder ein Button. Dann hast du Moleküle, Moleküle sind Kombinationen aus mehreren Atomen, also das könnte dann zum Beispiel eine Suchmaske sein, das hast du dann ein Label, ein Eingabefeld und einen Button. Das ist dann so ein einheitliches Ding. Dann hast du Organismen, die dann nochmal mehrere Moleküle zusammenfassen und dann die Templates, wo dann alles sozusagen zu einem großen Layout zusammengefügt wird. Daran wollte ich mich so ein bisschen orientieren bei Component Sprints, einfach um so eine Art Hierarchie hinzubekommen. Der Component Sprint soll in vier Tagen passieren, also auch relativ schnell, der Design Sprint davor dauert ja auch nur vier Tage, einfach damit man so schnell wie möglich dann auch wirklich ein Ergebnis bekommt.

Daniel: Ich dachte, der dauert fünf. Oder hat sich das geändert?

Leo: Ja, der standard Design Sprint dauert fünf Tage. Es gibt aber viele Firmen, die inzwischen gesagt haben, wir packen den ersten und zweiten Tag in einen Tag und sparen uns dadurch ein bisschen Zeit und haben dann dadurch den Freitag Puffer, um noch ein bisschen nachzuarbeiten. Dazu, also für den Component Sprint hatten wir uns überlegt, vielleicht vor dem Design Sprint schon mal zu schauen, was könnte denn das CI sein, dass, wenn ein Unternehmen schon ein CI hat, dass man damit schon mal grob die Atome baut, einfach damit man während des Prototyping auch schon ein bisschen Grundlagen hat und da schon mal ein grobes System irgendwie hat. Und während des Component Sprints geht's am ersten Tag darum, zu schauen, wie sieht der Prototype aus, was gibt's für Screens, welche Komponenten gibt es da, wie kann man das vielleicht unterteilen? Welche Komponenten sind von welchen Komponenten abhängig? Also ein Formular ist halt abhängig von Eingabefeldern und Buttons und so. Und welche haben Priorität, also welche Screens sind zum Beispiel die wichtigsten? Tag zwei und drei sind dann wirklich Implementierung, also da sitzt man dann mit, ich sag jetzt mal, drei oder vier Entwicklern wirklich konzentriert am Tisch und jeder nimmt sich eine Komponente raus und implementiert diese. Und am vierten Tag wird dann geschaut, dass alles, was noch nicht dokumentiert wurde, noch dokumentiert wird und da wird dann das Handoff gemacht an die Entwickler, die das MVP dann entwickeln. Wir haben das selber dann mal ausprobiert an Wevent, das war die App, die du dann auch getestet hast, sozusagen. Das war jetzt

kein, ich sag jetzt mal, echtes Produkt, also nichts was wir wirklich launchen wollen, aber wir haben uns das einfach mal als Probe genommen, um zu schauen, wie das ganze denn funktioniert. Der Design Sprint, das waren vier Personen, vier Tage lang und der Component Sprint drei Personen und auch vier Tage lang.

Daniel: Wo hast du die Entwickler hergenommen, um die Components zu bauen, oder hast du das selber gemacht?

Leo: Das waren zwei Kollegen von mir aus Düsseldorf, mit denen ich auch häufiger Projekte gemacht hab. Die haben sich zum Glück dazu bereiterklärt, das auch kostenlos zu machen, genau. Den Prototyp hast du ja glaube ich schon gesehen, den hast du ja getestet. Ich zeig ihn dir trotzdem nochmal kurz als Referenz, wenn es denn lädt. Das ist immer relativ fett. Das sind halt die Screens wo wir gesagt haben, die müssen wir irgendwie implementieren, als Komponenten, hier zum Beispiel diese Formulare, die Buttons, das Menü, so Geschichten. Und nach den vier Tagen Component Sprint ist das bei rausgefallen. Wir haben mit Storybook gearbeitet, sagt dir das was?

Daniel: Ne.

Leo: Storybook ist ganz cool, das ist im Grunde genommen ein Framework, das funktioniert mit React, mit Vue, mit React Native, wo du einfach sagen kannst, ich habe verschiedene Komponenten und möchte zu diesen Komponenten...

Daniel: (Telefon klingelt) Sorry, ich muss kurz dran.

(Telefonat)

Leo: Ja. Genau, mit Storybook kannst du einfach sagen, ich habe verschiedene Komponenten und kannst diese ganz gut und relativ einfach dokumentieren, also das ist das was die Entwickler letztendlich dann auch bekommen, als Paket. Da hast du dann direkt dann die Library, da kannst du sehen zum Beispiel, das sind die Buttons, die wir im Prototyp hatten, die gibt's auch als Primary, dann hast du auch die Möglichkeit zu sehen, was gibt's denn hier eigentlich für Properties für diese Elemente, zum Beispiel kannst du die einfärben, oder du kannst auch sehen wie das aussieht mit weniger Inhalt oder mehr Inhalt.

Daniel: Und wo liegen die Sachen? Ist das jetzt cloud-gehostet oder liegen die bei dir lokal irgendwo?

Leo: Das Storybook habe ich jetzt einfach auf Netlify deployed, ist aber auch ein Repo. Die Struktur, die du hier siehst, ist auch einfach im Repo genauso abgebildet.

Daniel: Okay, das heißt du kannst einfach eine Repo-Abhängigkeit in deinem Projekt erstellen und dann...

Leo: Richtig, genau. Du siehst ja auch... Gut, wir haben jetzt hier mit Linaria gearbeitet, das sieht so ein bisschen aus wie Styled Components, also CSS einfach im JavaScript. Aber so sieht dann die Definition von so einer Story aus. Also du sagst zum Beispiel, im Button gibt's verschiedene States, zum Beispiel Primary, und dann kannst du hier einfach in React so einen Primary Button reintun, und das wird dann dargestellt.

Daniel: Okay. Und in welche Richtung arbeitest du? Also, designst du in Storybook oder ist Storybook quasi nur ein UI für die Orga?

Leo: Genau, das ist hauptsächlich ein UI für die Orga. Es ist aber auch ganz praktisch, also wir haben das auch als Plattform genutzt, um darin die Komponenten zu entwickeln, weil du da direkt siehst, wie es aussieht und so. Also du hast auch Live Reload und die ganzen Geschichten.

Daniel: Okay.

Leo: Das spaltet sich dann weiter auf, wir sind da ein bisschen vom Atomic Design abgewichen, indem wir gesagt haben, dass wir bei Molekülen aufhören, weil alles andere macht eher für Webseiten Sinn, aber nicht unbedingt für Apps oder für Web-Apps. Und was dann am Ende auch rausfällt sind Templates. Ich mach das hier mal gerade auf dem iPhone zum Beispiel... Das ist wirklich direkt Code, den sich die Entwickler rauskopieren könnten, so wie er ist, und da dann einfach nur noch die Businesslogik mit einbauen müssen. Der Tech Stack gerade war jetzt React, Linaria für Styles und Storybook für die Dokumentation. Es ginge aber denke ich auch mit jedem anderen Tech Stack. Das Hauptsächliche Konzept ist wirklich die Entwicklung dieser Komponenten. Hast du generell jetzt an dem Punkt Fragen, so zur Vorgehensweise oder zur...

Daniel: Also was wir gemerkt haben, was ein hoher doppelter Aufwand ist, wenn wir Komponenten oder generelle Screens vordesignen, die nochmal in den Code zu bringen. Also wir arbeiten ja mit Figma, das kennst du bestimmt.

Leo: Ja, genau, das nutzen wir auch.

Daniel: Habt ihr einen Weg gefunden, wie man diese doppelte Arbeit sich sparen kann? Also zum Beispiel du hast jetzt hier gesagt, ihr habt fertige Components, so Button Components, ne? Kannst du die jetzt exportieren, sodass ich die in meinem Grafikprogramm benutzen kann?

Leo: Da wäre in Figma sehr cool, was wir inzwischen auch benutzen ist Framer. Was bei denen ganz cool ist, das läuft ähnlich wie Figma, nur kannst du zusätzlich zu deinen normalen Komponenten auch einfach React Komponenten da reinschmeißen.

Daniel: Okay.

Leo: Deshalb hatten wir auch überlegt, vor dem Design Sprint schon so Basiskomponenten zu erstellen, damit man die dann auch wirklich beim Prototyping da reinschmeißen kann. Also Framer kann ich dir sehr empfehlen, das kannst du dir gerne mal anschauen. Das ist gerade im Moment leider nur eine Mac App, die bringen aber wahrscheinlich in ein paar Monaten eine Web App raus, also...

Daniel: Okay, dann müssen wir darauf noch warten.

Leo: Genau, ja. Daniel war ja jetzt in Amsterdam bei der Framer Convention...

Daniel: Ah, da gibt's schon eine? Okay, nice.

Leo: ...und da haben die das Beta-Produkt vorgestellt, das sah schon sehr promising aus.

Daniel: Weil das ist genau so eine Stelle womit hier doppelter Aufwand anfällt. Und kriegt man das auch andersrum hin? Wenn ich jetzt in Framer quasi ähnlich wie bei Figma eine Component designe, kann ich die direkt in eine React Component umwandeln? Oder geht's nur in die eine Richtung?

Leo: Das sind eigentlich direkt React Komponenten. Einziges Problem ist, da ist super viel Overhead dabei, also die Positionen sind alle mit "position: absolute;" gemacht und so. Du designst halt für eine Bildschirmgröße und das dann so hinzukriegen, dass das zum Beispiel responsive ist und so ist schwierig.

Daniel: Okay, das können die noch nicht, oder?

Leo: Genau.

Daniel: Okay, aber das hört sich schon mal promising an.

Leo: Ja.

Daniel: Weil ich finde, das ist so die größte Lücke, die ich gerade sehe, in dem ganzen Tech Stack, dass du vom Design zum Code diesen Jump hast, wo du doppelt arbeitest. Der ganze Rest ist ja irgendwie nice integriert... Aber okay.

Leo: Also ich glaube, da tut sich auf jeden Fall noch was. So, genau, dann generell, siehst du in dem Component Sprint so als Konzept Potential für die Optimierung von der Entwicklung von MVPs, oder bzw. an welchen Stellen könntest du da Potential sehen?

Daniel: Also ich glaube, du hast ein bisschen Potential darin, dass du ein einheitliches Design hinbekommst von Anfang an, auch ohne Arbeits-Overhead. Ich glaube, das ist eine gute Sache. Ich glaube, was du im MVP wenig hast ist Wiederholung. Also dein MVP soll ja, wenn du jetzt dem Lean-Startup-Gedanken folgst, möglichst viel Wert schaffen durch möglichst wenig Dinge. Und ich glaube, da wird sich wenig Wiederholung zeigen, und ich glaube, dann ist der Wert dieser Components... Also der Wert der Wiederverwendung ist glaube ich bei einem MVP geringer, als wenn du sagst, du hast den MVP getestet und jetzt wissen wir: Wir brauchen noch sechs weitere Funktionen und die werden sich irgendwie ähneln oder du musst Prozesse aufsplitten in zwei, die ein bisschen anders sind. Dann würde ich... Also, wenn dein Produkt wächst, würde ich sagen. Dann würde ich sagen, dann ist der Wert höher, also für eine MVP-Entwicklung.

Leo: Okay. Siehst du denn... Ich weiß nicht, wie das die meisten Entwickler machen... Ist für dich ein MVP ein Throwaway-Prototype? Sowas, was man baut und wenn sich das validiert hat, das wegschmeißt und dann von vorne anfängt, oder...

Daniel: Also für mich wäre das am liebsten so, aber das Problem ist, Informatiker denken nicht so. Du musst ja auch irgendwie dein Team glücklich stellen. Also der MVP, den wir jetzt entwickeln, der ist kein Throwaway-Produkt. Wir sehen Potential darin, im Backend viel zu ändern, weil wir festgestellt haben... Das ist jetzt ein bisschen unrelated zu deiner Frage, aber wir haben festgestellt: AWS in der ganzen Kombinatorik dieser Einzelprodukte, die sie haben, da sind sie teilweise noch nicht fertig. Also wir haben zum Beispiel Probleme, mit diesen Lambda-Funktionen auf unsere DynamoDB zuzugreifen.

Leo: Oh, okay. Das ist ja nicht optimal.

Daniel: Weil die Authentifizierung irgendwie anders läuft, als über die User, die über Cognito... Also solche Kleinigkeiten sind noch Probleme, und dann ist halt die Frage:

Nutzt man das überhaupt, oder hostet man den Server dann doch wieder selber, mit einer MySQL DB oder wie auch immer. Das heißt, da kann Refactoring passieren. Und im Frontend kann auch Refactoring passieren, aber eher darin, dass man so eine Library von unten aufbaut, die ein bisschen konsequenter noch genutzt wird. Aber ansonsten würde ich sagen, ist das Produkt schon... Es wird nicht weggeschmissen oder so.

Leo: Okay, ja.

Daniel: Genau, also so ist es bei uns auf jeden Fall geworden. Wenn man dem Lean-Startup-Gedanken 100-prozentig folgt, dann machen wir es falsch, da bin ich mir auch sicher. Du merkst halt auch, wenn du Sachen richtig machst, dauern sie länger.

Leo: Ja.

Daniel: Und wir entwickeln jetzt schon, weiß ich nicht... Wir haben ja erst eine PWA gemacht, die haben wir weggeschmissen. Jetzt machen React Native. Wann haben wir gesprochen, vor zwei Monaten oder so?

Leo: Ja, das ist schon eine Weile her, ja.

Daniel: Das ist schon eine Weile her und wir sind immer noch nicht fertig. Wir sind fast fertig, aber immer noch nicht. Also wir haben drei Monate oder so gebraucht. So lange sollte man glaube ich nicht brauchen. Aber ja.

Leo: Von dem was du jetzt gesagt hast würde es sich ja vielleicht sogar anbieten, dass man den Component Sprint nicht vor dem MVP macht, sondern nach dem MVP...

Daniel: Absolut.

Leo: Wenn sich der MVP validiert hat und man sagt, wir wollen das dann wirklich weiterentwickeln und wachsen.

Daniel: Das glaube ich auf jeden Fall auch, weil die... Also mit unserem Produkt ist es so... Ich kann dir das auch kurz zeigen, wenn dir das zum Verständnis hilft...

Leo: Ah ja, gerne.

Daniel: ...wie weit wir jetzt gerade sind. So, mal gucken, ob ich mich anmelden kann, das sollte eigentlich klappen. Also, was wir jetzt machen, das ist einfach ein klassisches Dashboard mit deinen nächsten Fahrten, ein bisschen Kommunikation

mit dem Fahrer. Du hast eine Fahrtenübersicht über die nächsten zwei Wochen, wo du planen kannst, mit wem du wie wann wo fährst. Du hast eine Übersicht über deine Matches generell. Eigentlich kann man hier draufklicken, aber dann crasht die App gerade, also mach ich es mal nicht. Und da siehst du noch ein bisschen mehr Details, da kannst du auf der globalen Ebene entscheiden ob du miteinander fahren willst, und das war's auch schon. Mehr haben wir nicht. Diese Glocke hier mit Notifications kommt noch raus. Wir haben hier ein Feedback-Board von Nolt, aber das haben wir natürlich nicht gebaut, das ist nur ein Webview. Und hier sind noch Impressum und Support, mehr ist das gar nicht. Das heißt, die Komplexität ist jetzt gering gehalten, damit wir fertig werden und testen können, aber wir haben schon quasi Features in der Pipeline, die wir uns vorstellen können, die die Komplexität enorm erhöhen. Gutes Beispiel ist, jetzt in unserem Fall, Anzahl der Mitfahrer. Also was wir jetzt machen, wir haben nur zwei Personen immer in einem Auto. Weil alle Prozesse... Zwei ist ja immer ein Sonderfall, egal wo du bist, und viel einfacher als n Personen. So, und wenn wir jetzt über n Personen reden, dann... Nur diese Änderung würde bei uns so viele neuen Screens und Prozesse hervorrufen, dass es dann davor schon Sinn macht, über Wiederverwendbarkeit von Komponenten sich mehr Gedanken zu machen. Weil jetzt siehst du, wir haben fast null Wiederverwendung.

Leo: Ja, das stimmt. Da würde das ja echt nicht viel Sinn machen.

Daniel: Genau. Aber sobald wir merken, okay, wir bauen das und das Feature und das bringt die und die Komplexität, dann macht das Sinn. Anderes gutes Beispiel, jetzt für unseren Fall: Wir haben jetzt versucht, eine Mischung zu bieten für regelmäßige und flexible Arbeitszeiten. Die gibt's bei der Registrierung, da gibst du deine Arbeitszeiten an, und dann sind diese Zeiten immer deine Default-Zeiten für die nächsten Wochen. So, und dann ist das quasi die Sache, wenn du immer regelmäßig arbeitest, musst du die nie ändern, aber wenn du unregelmäßig arbeitest, musst du die schon ändern, um da eine Dynamik reinzukriegen. Und jetzt ist die Frage: Macht es Sinn, diese Mischung zu haben, oder ist es cooler, bei der Anmeldung zu sagen, ich fahr unregelmäßig oder regelmäßig, und dann zwei Prozesse daraus zu machen. Auch ein gutes Beispiel, weil dann würde die Überlappung auch ziemlich hoch sein, Komponenten-wise.

Leo: Okay, ja, das ist schon mal...

Daniel: Also du siehst, nach dem MVP kommt viel Komplexität, die man im MVP vielleicht vermeidet.

Leo: Das ist schon mal auf jeden Fall sehr guter Input. Eine Frage wäre auch noch so ein bisschen die Frage der Zielgruppe. Weil, was ich so ein bisschen gesehen hab, ist halt, dass Startups ja relativ viel auch einfach auf Stock-Komponenten zurückgreifen, also Material UI, oder, da gibt's ja... Ne, Bootstrap, was auch immer.

Daniel: Also wir haben auch Bootstrap drin.

Leo: Wie siehst du das so, für welche Zielgruppe könntest du dir so Component Sprints vorstellen? Sind das für dich eher Startups oder eher vielleicht auch schon mittelständische Unternehmen, die einfach mehr Dokumentation in ihr UI bringen wollen, oder...

Daniel: Du hast ja beides, oder? Also nach meinem Verständnis hast du zwei Vorteile: Einmal, du wirst schneller, und auf der anderen Seite bist du auch einheitlicher. So, den großen Unternehmen ist glaube ich die Geschwindigkeit... Das ist vielleicht nicht das Kernproblem von denen, weil die eh langsam sin, damit haben die sich abgefunden. Aber ich glaub, das einheitliche, das Branding ist denen ein bisschen wichtiger. Beim Startup würde ich klar sagen, wenn du denen Components zur Verfügung stellen kannst, mit denen die Entwicklungsgeschwindigkeit um 15% steigt, oder was, dann ist das ein super wertvoller Faktor.

Leo: Okay, ja. Also beide Zielgruppen sind valide.

Daniel: Ja.

Leo: Cool. Dann, angenommen, du würdest jetzt so einen Component Sprint mit uns zusammen machen, hättest du gerne, dass Entwickler von deiner Seite mit am Component Sprint beteiligt sind, oder würdest du es eher so sehen, dass du das komplett an das dritte Unternehmen abgibst, und die machen das und liefern dir am Ende ein Endprodukt?

Daniel: Absolut brauchst du Entwickler drin. Also zumindest in meiner Erfahrung jetzt. Komplettes Abgeben würde ich sowieso nicht machen, weil du bräuchtest einen, der das Knowhow hat, was noch kommen kann, Szenarien, die man vielleicht abdecken möchte, mit so einer Component-Erstellung. Und du brauchst auch die Entwickler, die vielleicht das aktuelle Produkt schon kennen. Bei uns ist das jetzt vielleicht ein bisschen ein Sonderfall. Alle Entwickler, die wir haben, sind auch

irgendwo Design-affin. Also wir haben gerade keinen Entwickler im Team, der einen Fuck gibt. Das heißt, denen ist es auch wichtig, wie das Produkt aussieht und deswegen würde ich die auf jeden Fall an Bord haben wollen.

Leo: Cool. Jetzt muss ich mal überlegen, ob ich noch was hab. Ich kann vielleicht auch nochmal kurz über die Retro gehen, die wir selber gemacht haben. Vielleicht so ein bisschen noch die Frage vom Scope. Also es kann ja sein, dass so ein Produkt, wo man vielleicht so einen Component Sprint für machen möchte, schon relativ komplex ist. Jetzt haben wir halt überlegt, du hast die Möglichkeit, glaube ich, den Component Sprint was die Personen angeht, also die Anzahl der Entwickler, relativ gut zu skalieren. Jeder kann ja unabhängig von den anderen an Komponenten arbeiten. Das heißt, ich habe so ein bisschen die Hoffnung, dass, wenn du ein größeres Projekt hast, dass du einfach sagen kannst: Okay, dann nehmen wir statt drei Entwicklern halt sechs, die in diesen vier Tagen den Component Sprint machen. Siehst du da vielleicht potentielle Schwierigkeiten, was diese unabhängige Entwicklung voneinander angeht, bzw. die Skalierbarkeit? Du hast ja jetzt auch so ein bisschen Erfahrung dadurch, ihr wachst ja gerade auch als Unternehmen, und ihr skaliert ja gerade auch.

Daniel: Ja.

Leo: Wie ist da so deine Einschätzung vielleicht?

Daniel: Ich bin mir nicht sicher, ob die Woche Vorbereitung dann reicht. Weil, wenn du sagst, du gibst sechs Entwicklern einen ganzen Tag, das sind dann 50 Stunden, dann musst du schon sehr genau wissen, was du eigentlich brauchst. Und dafür musst du den kompletten Prozess eigentlich schon geskecht haben, von dem du diese Components entwickeln willst, und dann vergisst du meistens trotzdem irgendwas. Ich glaube, wenn man da einen guten Weg findet, in den drei Tagen davor, oder wann ist der Entwicklungstag? Am dritten wahrscheinlich, ne?

Leo: Der Entwicklungstag, das sind die Tage zwei und drei, bzw. vielleicht vier auch noch.

Daniel: Okay, dann müsste man am ersten Tag schon sehr klar definieren, was ist jetzt eigentlich der Scope, den ich haben will. Ich mein, so Components sind ja von der Implementierung relativ unaufwändig häufig.

Leo: Im Optimalfall, ja.

Daniel: Das heißt, du musst schon sehr weit denken können, um so Informatikertage auch auszufüllen, glaube ich. Oder wenn du sechs hast, natürlich noch weiter. Ich sehe ein Risiko, dass du dir in der Prozessausgestaltung nicht genug Mühe gibst und dann entweder Sachen fehlen oder vielleicht Sachen zu viel entwickelt werden, die man doch nicht braucht. Aber wenn man da einen guten Prozess findet, dann... Warum nicht, ne?

Leo: Okay.

Daniel: Also wie gesagt, wenn ich jetzt die beiden Beispiele die ich eben genannt hab... Wenn wir jetzt hingehen würden... Wir haben jetzt das MVP, wir haben das Userfeedback bekommen, wir brauchen auf jeden Fall Viererfahrgemeinschaften. Von unserem Arbeitstempo weiß ich aber schon, in einem Tag kriegen wir den Prozess nicht definiert. Weil, erstmal ist es kreative Arbeit, das ist immer sehr anstrengend, dann willst du vielleicht noch ein User Feedback zwischendrin, ob das Konzept gut ist, bevor du es baust. Das heißt, ich glaube, man bräuchte da als Firma selber, bevor man in so einen Component Sprint einsteigt, eine gute Vorstellung davon, was man haben will. Ich denke schon, dass das... Wahrscheinlich wird's eine Woche oder so bei uns dauern, bis wir mit dem Konzept soweit zufrieden wären, dass wir jetzt sagen, wir bauen das jetzt richtig.

Leo: Das heißt, so eine Pre-Flight Woche vor dem Component Sprint wäre vielleicht angebracht, wo man so Sachen schon mal alle definiert.

Daniel: Ich glaube, wenn der größer wird, dann wäre es hilfreich, ja. Also wäre zumindest mein Bauchgefühl jetzt gerade.

Leo: Okay, cool. Ich glaube, von meiner Seite aus war's das an Fragen. Das waren die Dinge, die ich auf jeden Fall wissen wollte. Hast du sonst noch irgendwas, was du dir aus der Seele sprechen möchtest?

Daniel: Wir haben tatsächlich schon mal über so einen Component Sprint mit euch nachgedacht. Natürlich nicht jetzt, wir müssen erstmal beim MVP nachschauen. Und was ich mir gedacht hab: Es wäre eigentlich voll nice, wenn einer von unseren Entwicklern sogar bei der Implementierung dabei wäre. Das würde ja den Preis drücken, also je nachdem wie man es betrachtet würde es den Preis drücken, und du hättest nochmal einen Knowhow-Transfer nach außen. Also es drückt den Preis und es macht es wertvoller. Das wäre so ein Gedanke, den ich hatte.

Leo: Weil der Entwickler auch direkt sieht, wie es implementiert ist.

Daniel: Ja, der war ja dann ein, zwei Tage komplett dabei, hat auch so die Philosophie, lernt vielleicht auch ein, zwei neue Tools, wie auch immer. Wenn ihr das schon häufiger gemacht habt, dann lernt man ja auch nochmal, wie man das schneller macht, zum Beispiel. Das sind Gedanken, die wir hatten, aber da kann man ja in ein paar Monaten drüber sprechen.

Leo: Ja, sehr gerne. Bis dahin ist das auch hoffentlich dann komplett ausgearbeitet.

Daniel: Ist das ein Konzept, was es jetzt in diesem Atomic-Ding schon gibt oder sind Component Sprints quasi jetzt eine... Was ihr euch selber ausgedacht habt?

Leo: Component Sprints haben wir uns selber ausgedacht, also den Begriff gibt's bisher noch nicht, oder zumindest habe ich ihn bisher so noch nicht gesehen. Atomic Design gibt's natürlich schon, und das ist auch eigentlich schon veraltet, also ganz viele sagen inzwischen schon: Nutzt nicht mehr Atomic Design, weil das ist zu sehr hierarchisch aufgebaut, das ist zu sehr noch der Blick auf, wie bauen wir die Komponenten auf, und was so ein bisschen fehlt ist, welchen Zweck erfüllen die Komponenten eigentlich. Da gibt's jetzt noch andere Systeme. Es gibt zum Beispiel ein Buch von... Ich kann mir den Namen nie merken... Mal kurz schauen... Imprint... Ja, Alla Kholmatova. Die sagt, es ist neben den Komponenten selber super wichtig, dass diese Komponenten auch wirklich benutzt werden, weil sonst hast du so eine Library, die da einfach rumliegt und vielleicht schon erweitert wird, aber dann kommt es irgendwann vor, dass eine Komponente nicht genau das erfüllt, was gemacht werden muss, dann wird dafür eine neue Komponente entwickelt. Dann hast du zwei Komponenten, die fast das gleiche tun und dann wird das, vor allem wenn du das weiterdenkst, zeitlich, wird das zu komplex. Mein Ansatz war jetzt, vielleicht da so ein bisschen so einen Hybrid zu fahren. Weil ich mag diese Hierarchie eigentlich ganz gerne, vom Atomic Design. Das macht es sehr einfach, den Component Sprint zu strukturieren.

Daniel: Hast du es schon mal angewandt, neben deinem eigenen Mini-Sprint? So im Entwicklungsalltag, also baust du dir solche Bibliotheken selber?

Leo: Wir sind gerade dabei, für Crisp so eine Bibliothek zu bauen. Das ist aber auch das einzige. Also ich habe das sonst bisher noch nicht wirklich angewendet.

Daniel: Okay, das heißt, du kennst persönlich die Limits auch noch so gar nicht.

Leo: Richtig, genau. Das wollte ich jetzt alles so ein bisschen exploren und schauen, was da so möglich ist.

Daniel: Und, arbeitest du verschiedene Varianten von so einem Component Sprint jetzt aus? Das ist eine Bachelorarbeit, ne?

Leo: Mhm.

Daniel: Da hast du ja gar nicht so viel Zeit.

Leo: Richtig, genau. Was ich jetzt mache ist: Das war die einzige Variante, die wir ausgetestet haben, jetzt mit Wevent sozusagen. Und die Interviews, also das was wir jetzt gerade führen, ist jetzt sozusagen der nächste Teil, um einfach zu schauen, in welche Richtung könnte man jetzt vielleicht, wenn man mehr Zeit hätte, noch schauen, wie man das noch optimieren kann. Das sind dann Dinge, die man vielleicht in einer weiteren Bachelorarbeit noch machen könnte.

Daniel: Oder in einer Masterarbeit, wer weiß.

Leo: Ja, genau. Vielen Dank, dass du dir die Zeit genommen hast.

Daniel: Ja, cool.