Interview 1

Speaker1: [00:00:02] Okay, dann lass uns starten. Hallo, Interviewpartner. Ist es für dich in Ordnung, wenn wir die. Gespräch hier aufzeichnen.

Speaker2: [00:00:12] Ist alles in Ordnung. Ja.

Speaker1: [00:00:14] Danke dir. Ich möchte kurz. Ich habe hier einen kleinen Interview Leitfaden vorbereitet. Das soll dazu dienen, das Gespräch ein bisschen zu strukturieren. Ich würde kurz erklären, welchen Studiengang ich mache und dann auch direkt starten mit dem Beispiel um was es geht. Dass ich deine Inputs abholen kann. Mein Studiengang ist an der Berner Fachhochschule und zwar ist es der Master in Digital Business Administration. Es ist das erste Jahr bzw. wir sind der erste Lehrgang. Falls du innerhalb deines Unternehmens Werbung für diesen Lehrgang machen möchtest, ss ist sehr empfehlenswert. Kann ich dir gerne gerne empfehlen oder dir auch im Nachgang nähere Informationen zur Verfügung stellen.

Speaker2: [00:00:59] Klingt spannend.

Speaker1: [00:01:01] Wieso ich heute mit dir ins Gespräch komme? Es geht um das Thema Low Code Softwareentwicklung. Und zwar möchte ich untersuchen, inwiefern die Softwareentwicklung durch den End-User überhaupt in einem Scrum Prozess abbildbar ist. Das ist so ein bisschen der Fokus der Arbeit. Und als konkretes Beispiel: Ich arbeite in der Versicherung. Dort ist es so, dass wir einzelne Business User haben, die in Zukunft selbst Applikationen entwickeln werden. Das heißt, sie sind heute nicht Teil eines Scrum Teams und sie werden sicherlich eher Teilzeit entwickeln. Also der klassische Citizen Developer Approach und nicht ein klassischer Softwareentwickler, der das Ganze macht. Gerne möchte ich damit beginnen, etwas mehr über dich zu erfahren. Kannst du mir sagen, was deine aktuelle Position ist und kurz beschreiben, was dein tägliches Tätigkeitsfeld ist?

Speaker2: [00:02:01] Ja, vielleicht zu meinem Hintergrund. Ich bin ausgebildeter Informatiker, das heißt, ich habe an der Fachhochschule Rapperswil, heute Ost, einen Informatikabschluss gemacht und bin 2018 dann zu Firma X angestellt worden als Fullstack Web Developer. Das heisst ich habe mich mit Dotnet Backend und im

Frontend Angular React klassischen modernen Softwareentwicklung beschäftigt. Übrigens dort auch schon in der Versicherungsbranche lustigerweise.

Speaker1: [00:02:34] Ah witzig, dann haben wir ja Überschneidungen.

Speaker2: [00:02:38] Genau. In 2019 kam dann die Möglichkeit auf, dass ich da bei einem internen Projekt starten konnte, das sich erstmals mit einer Low Code Plattform auseinandergesetzt hat. Das baute auf auf einer schon vorhergehenden Prototypen-Phase, die sie dort schon durchgeführt hatten und dort zum Schluss gekommen sind, dass sie dann ein wirklich Fully Fletched Project mal auf seiner Plattform durchziehen möchten. Und so bin ich da reingekommen, seither dort geblieben, weil es macht auch Spaß und es hat mich schon an verschiedenste Punkte hingeführt, die ich wahrscheinlich so auf dem klassischen Weg gar nicht so getroffen hätte. Namentlich waren das halt auch interessante Gespräche mit Leuten, die sich gerade auch mit diesen Plattformen beschäftigen. Intern auch viele Diskussionen Was soll das? Wie steht das zu normalen Softwareentwicklung? Wo geht das hin? Wollen wir das überhaupt? Bis hin auch zu Kunden, die einfach sagen Ja, wir wollen das Problem gelöst haben und schlussendlich ist uns egal, wie ihr das macht. Also da gab es alles.

Speaker1: [00:03:46] Aber das ist ja dann die dankbare Position, wenn der Kunde sich da eher freier zeigt, wie man es umsetzt. Es gibt ja auch oftmals so Governance, oder? Beziehungsweise die Frage ist, was sind mögliche Einschränkungen oder bei der Umsetzung auch.

Speaker2: [00:04:01] Genau. Insofern bin ich heute Teil der Cloud Practice bei uns. Das ist ein Mix aus Consulting und Delivery Unit, das sich um Cloud Technologien dreht. Da drin gibt es ein Sub-Team, das sich spezifisch auf Low-Code Plattformen spezialisiert hat. Da bin ich Teil davon und beschäftige mich da einerseits in der Umsetzung, aber auch in der Bearbeitung von Anfragen dann sehr stark damit, ob man das Low-Code machen kann oder eben lieber davon Abstand nehmen sollte.

Speaker1: [00:04:31] In welchen Themenfeldern habt ihr Low Code eingesetzt? Also kannst du dort unterschiedliche Bereiche definieren?

Speaker2: [00:04:43] Ich glaube bei uns der Start und auch das Hauptthema sind meistens Kundenschnittstellen, Portale, die sich schnell zusammenbauen lassen und im Kern eigentlich keine Rocket-Science implementieren müssen, sondern wirklich den Prozess abbilden sollen. Die Daten Modellierung, das UI möglichst so zu bauen, dass das dem Business Case dann auch entspricht und sich da nicht sozusagen from the scratch die ganze Architektur zusammenbauen muss, sondern man sich da auf einer Plattform zurecht findet und da auch mit der Zeit eine gewisse Routine entwickelt, wie man da schnell ein Projekt aufgleisen und durchziehen kann. Darüber hinaus gibt es natürlich auch Kunden, die aufbauend auf zum Beispiel einen großen Cloud Anbieter wie zum Beispiel Azure dann sagen gut, wir haben die ganzen Daten. Wir haben auch gesehen, dass es jetzt solche Schnittstellen und Konnektoren gibt. Wir möchten die besser nutzen diese Möglichkeit, dass wir diese Journey to the Cloud umgesetzt haben und bewegen uns dann Richtung Power BI, Power Plattform. Jetzt bei Microsoft oder auch bei verwandten Angeboten. Bei SAP ist es dann Fiori-Geschichte, die da draufkommt. Und überall wo ein großes Framework oder ein großer Datenpool schon steht, bieten jetzt verschiedene Plattformen auch so Aufsätze an. Dort gilt es dann vor allem abzuwägen, ob das Sinn macht oder nicht.

Speaker1: [00:06:09] Inwiefern unterscheidet sich die Low Code Softwareentwicklung von einer klassischen Softwareentwicklung? In deiner Sicht: Wo siehst du da die größten Unterschiede?

Speaker2: [00:06:20] Ich bin ja sogenannter Fullstack-Entwickler. Was ich beobachtet habe ist, dass man als Teil eines Projektteams, obwohl man Fullstack-Entwickler ist, sich immer auf spezifische Layers dann beschränkt. Beziehungsweise man teilt das dann auf. Ja, ich bin etwas mehr im Frontend oder ich schaffe an einer Backend Ecke mit. Ich glaube so teilt sich das dann schnell auf bei so einer Low Code Applikation, da gibt es diese Aufteilung sehr viel weniger, weil die Modellierung und auch die ganze Entwicklungsumgebung meistens abschließend bearbeitet wird und man das Ganze nur von oben anschaut, sage ich mal und nicht in seinen einzelnen Stufen. Es hat auch seine Tücken, aber schlussendlich auch seinen Vorteil, weil dann das ganze Team jeweils sehr nahe am eigentlichen Teil der Umsetzungs-Story dran arbeitet.

Speaker1: [00:07:15] Und welche Art bzw wer ist der Anwender bei euch von Low Code? Man spricht ja oftmals vom Citizen Development Gedanken, wo der User

programmiert. Wie lebt ihr das? Sind es bei euch eher wirklich Entwickler, die bereits Entwicklungserfahrung haben oder auch tatsächlich nicht IT-nahe-Personen, die dann mitentwickeln?

Speaker2: [00:07:43] Also ich ich selber arbeite bei Firma X. Wir sind ein Software Dienstleister. Das heißt wir haben keine eigenen Produkte, sondern entwickeln grundsätzlich Custom-Software. Und so sind unsere Profile meistens diejenigen, die Softwareentwicklungs-Know-how haben und auch diese Brücke schlagen können, wo zum Beispiel eine Enterprise Framework oder Architecture Roadmap sich Gedanken macht, ob da eine Low-Code-Plattform dazu passt oder unter welchen Bedingungen das möglich ist. Da holen wir dann meistens die technische Seite gut ab, indem wir halt eben auch diese Erfahrung mitbringen, die ganz entscheidend ist, um auch eine solche Plattform zu bewerten. Meiner Erfahrung nach kann man sehr gut auf der grünen Wiese starten mit einer Low-Code-Plattform, wenn man keine Dependancies berücksichtigen muss. Aber wenn das Ganze in eine Corporate Landschaft eingebaut werden soll, dann muss man doch Einiges unter der Haube verstehen, dass die Plattform dann mitbringt oder einfach als unveränderbar dann einfach hinstellt. Auch für die entsprechenden Entscheider die nötigen Grundlagen dann vorhanden sein müssen, um da mit zu reden oder auch Ausschlusskriterien dann zu formulieren.

Speaker1: [00:08:58] Und wenn du an das Szenario denkst, das zum Beispiel innerhalb eines Unternehmens nicht IT-nahe-Personen Low-Code-Applikationen entwickeln sollen, wo siehst du dort die Chancen und Risiken?

Speaker2: [00:09:13] Ja. Ich glaube also an eine Variante, die ich sehr zukunftsweisend und auch sehr sinnvoll gesehen habe. Das war, dass man so eine Art Capability Center aufbaut. Also Leute, die sich intensiv mit der Plattform beschäftigt haben, die wahrscheinlich dann eher IT seitig kommen und wirklich Ahnung haben, wie man die Plattform baut und sie bedient auch gewisse Konventionen oder Best Practice Guides für Inhouse dann erstellen und die schaffen dann so eine Art Referenzrahmen, sag ich dem gerne. Und in diesem Referenzrahmen kann man dann Leute onboarden. Also onboarden kann dann zum Beispiel sein, eine Fachabteilung, eben so ein Citizen Developer Poweruser. In anderen Kontexten sagt, ich hätte da Spaß dran, nicht nur ein Excel selber zu bauen, sondern das Ganze in eine eine Web Application umzubauen, die ich auch noch auf meinem Mobile brauchen kann für irgendeinen Ablauf. Und das

erste, was dann die dieses Capability Center macht, ist einen großen Freiraum aufzubauen oder zu sagen Gut, du kannst das machen, du kannst das testen. Wir geben dir vielleicht sogar Zugriff auf gewisse APIs. Aber das ist immer noch eine Art Playground, um zu validieren, dass das Ganze erstens Spaß macht, zweitens sich die Fragen, die man vielleicht zu Beginn noch hat, dass sich die klären und auch gewisse eben vielleicht ein onboarding workshop oder sonstwas zu Verfügung stellen, damit so ein ganzes Nicht-IT-Team mit der Plattform zurechtkommt und auch irgendwie so einen Spaßfaktor damit entwickelt. Ich glaube, in der Phase ist vor allem wichtig, dass sie sich zu eigen machen können, also möglichst wenig Interferenz mit einer IT Abteilung oder starke Strukturen vom Prozess her gegeben sind, sondern mehr so das Spielerische und auch die Befähigung im Zentrum steht. Irgendwo zwischen Jawoll, das ist eine tolle Idee und wir nutzen das jetzt produktiv für immer, da muss so ein Quality Gate stehen. Also eine Art Checkliste. Man sieht das in verschiedensten Formen, das kann man sogar zum Teil automatisieren, dass die App, die da gebaut wird, gewisse Anforderungen erfüllt, die dann von verschiedensten Stakeholdern definiert sind. Beispiele davon könnten sein, dass man auch gewisse Prozesse, also wer das Ding später auch noch weiter bedienen darf oder auch wer in Zukunft Ownership über so eine App dann hat, über wen der Support Kanal dann nachher läuft und so. Also solche Dinge müssen geklärt sein, bevor man sagt gut, das ist Mature oder skalierbar oder wie auch immer man das nennt. Und wenn so eine Hürde gut gemacht wird, dann ermöglicht man so einen optimalen Flow. Dann spielt die Innovation in diesem Playground und es fühlen sich Teams befähigt, da auch Dinge zu verändern und über die Hürde drüber zu kommen. Dann werden dann auch die Qualitätsansprüche oder auch längerfristig wichtige Themen für Data Governance abgesichert. Und ich glaube, so würde ich es beschreiben, wäre so meine optimale Vorstellung davon, dass sich auch eine größere Organisation darauf einlassen kann.

Speaker1: [00:12:34] Welche weiteren Herausforderung siehst du bei der End User Programmierung generell? Wenn du darüber nachdenkst, dass nicht IT nahe Leute dann tatsächlich Software entwickeln?

Speaker2: [00:12:48] Genau. Ich glaube, es hängt immer davon ab, wie gut sich jemand vorstellen kann, was mit einer bestehenden Applikation passiert, wenn da einfach Veränderungen gemacht werden. Das Bewusstsein ist nicht einfach zu lernen, wenn man die entsprechenden Probleme nicht selbst gespürt hat. In einer Exceltabelle

kann man sich das vorstellen, oder? Aha, ich habe die Formel verändert. Logisch, die früheren Versionen der Excel Tabelle haben noch die anderen Ergebnisse. Ich habe die Formel gedreht, also die neuen Versionen werden ein neues Ergebnis bringen. Jetzt eben bei einem Versicherungsrechner zum Beispiel ist das klar altes Excel, neues Excel. Aber bei einer Applikation ist das nicht so einfach. Oder man hat irgendwie alle Daten, alle Versionen laufen noch in der App drin. Unter Umständen ist es dann nicht ganz so klar, wie der Migrationspart aussehen soll. Oder dass man überhaupt einen braucht, ist vielleicht auch nicht so klar. Und da sehe ich eines der Probleme oder wo man sicher auch Bewusstsein schaffen muss, was mit so einem Data State von einer ganzen Applikation und eben nicht nur einer Datei passiert, wenn man da Logik umbaut. Was ziemlich gut funktioniert, denke ich, ist Anpassung auf UI Bedürfnisse. Also eben Button hier, Button dort, Validierungsmessages, zusätzliche Hilfestellungen, auch vielleicht solche Popup Guides, die irgendwie im richtigen Moment die nötigen Hinweise geben. Auch Übersetzungsfehler einfach schnell korrigieren zu können. Oder wenn Fachexperten da draufschauen und sagen Ja, da hat einer einfach nicht mitgedacht, das müsste auf Italienisch so und so heißen und das dann einfach selber machen kann, ohne gleich einen ganzen Rattenschwanz an Änderungen auszulösen. Da sehe ich Stärken. Sobald es eine längerfristig produktive Applikation ist. ist der Migrationspfad die Knacknuss.

Speaker1: [00:14:46] Wenn ich wenn ich jetzt dennoch nochmals an die professionelle Entwicklung durch euch denke, wie arbeitet ihr zusammen? Habt ihr dort zum Beispiel ein Scrum Framework, den ihr verwendet oder wie kollaboriert ihr? Wie könnt ihr auch die Aufwände schätzen für die Entwicklung?

Speaker2: [00:15:05] Also wenn wir ein Low Code Projekt machen, einfach die Plattform nutzen im Auftrag des Kunden da umzusetzen, dann haben wir da einen ziemlich klassischen Projekt Flow, Scrum. Da unterscheidet sich die Kommunikation eigentlich sehr, sehr wenig von der normalen Projekt Kommunikation. Ich glaube, was für mich ein Learning war aus den ersten Projekten, dass das ganze Fullstack läuft, dass man viel schneller eigentlich schon am UI mitarbeitet und auch mit der ersten Modellierung schon Entscheidungen UI trifft. Die sich dann lohnen. Früher auch schon mit einem Art Designkonzept abgefangen zu haben, wenn man ein produktives Produkt baut. Im Unterschied zu anderen Projekten, wo man noch mehr Zeit hat, bis man am Frontend angekommen ist, weil man meistens hinten anfängt. Ich glaube, wenn man

jetzt noch dazu kommt, dass ein bestehender Arbeitskreis oder Team schon existiert, der sich mit seiner Power BI Installation oder sonst was schon zurecht findet, dann ist denke ich wichtig, dass wenn man das Projekt dann irgendwie begleitet, dass man sehr genau hinschaut. Was sind eigentlich die Assets, die, die das Team da alles schon aufgebaut hat? Jetzt gerade bei Power BI entsteht da schnell mal ein Zoo von Dingen, oder? Und welche Dinge oder welche Dashboard, welche Queries usw nun effektiv genutzt werden sollen und welche einfach mal so probehalber irgendwo mal eingefügt wurden. Da muss man dann so eine Sprache im Projekt entwickeln, dass man das klar benennt und da auch einen Überblick behält.

Speaker1: [00:16:53] Inwiefern wirkte sich auf die Rollen aus? Ich kenne es klassisch aus einem Software Entwicklungsprozess, wo man an einem Kern System arbeitet, mit einem gemeinsamen Product Goal. Dass man dort Rollen hat, wie zum Beispiel ein Product Owner. Und wie wirkt sich das zum Beispiel in Kombination mit Low Code Development aus?

Speaker2: [00:17:21] Ich glaube, es hängt ein bisschen an der Größe des Projekts. Wenn es ein großes Projekt ist, dann denke ich, ist das nahezu identisch. Weil die Rollen sind ja nicht da, weil eine gewisse Technologie gewählt wurde, sondern weil man sich damit auch die entsprechenden Themen und Konflikte auch aufteilt. Und da ist es von der Plattform oder von der Technologie, die man drin hat, eigentlich nicht so abhängig, oder? Das Wichtige ist Wie kommen die Anforderungen von wo rein und wo werden die Risiken gemanagt? Das ist so die Hauptfrage eines Projekts und die ganzen Entwicklungsschritte, also wie das eigentliche Doing ist, ob ich jetzt da einen Commit mache und irgendeinen neuen Unit Tester einstelle und irgendwelche Zeilen schreibe dafür, oder ob ich einen Funktionsbaustein dazu schreibe oder Drag and Drop. Das ist dann weniger relevant. Ich glaube, je mehr Citizen Developer drin sind, desto weniger ist die Practice verfügbar oder bekannt, wie man Software entwickelt. Und da muss man sich wahrscheinlich sehr stark als Team auch so in eine Art Norming-Prozess begeben, um herauszufinden, ob man da mehr wie ein Software Projekt arbeiten möchte oder mehr wie eine Business Arbeitsgruppe arbeitet. Kanban, was auch immer. Und das hat viel mit Teambuilding zu tun. Ich denke, da ist die Plattform vielleicht Auslöser dafür, dass man sich das neu überlegen muss, aber nicht die Plattform selbst, sondern die Tatsache, dass andere Leute in dem Projektteam dann drin sind. Also einfach gesagt, wenn wir das Projekt dann von Hand coden würden, wäre das Äquivalent mit einem

einfach sehr cross-funktional aufgesetztem Projektteam, das schlussendlich auch ziemlich cool wäre.

Speaker1: [00:19:09] Wenn man darüber nachdenkt, dass man sagt, man hat dezentralisierte Entwickler den End-User, die an unterschiedlichen Themen arbeiten, inwiefern gehst du davon aus, dass ein Scrum Prozess geeignet wäre bzw falls du einen einen Scrum Prozess anwenden würdest: Welche Elemente würdest du infrage stellen?

Speaker2: [00:19:34] Ich glaube, gerade bei der dezentralen Entwicklung oder bei den Dingen, die dann so intrinsisch motiviert kommen, da habe ich den Verdacht, dass es nicht so den klassischen Ablauf gibt, dass von oben nach unten gesagt wird Ja, wir haben jetzt so und so viel Ressourcen und ihr habt so und so lange Zeit und das ist der ganze Scope. Ich glaube, das wäre dann eher so rum, dass von unten her gesagt wird Hey, wir geben, wir nehmen uns jetzt ein bisschen Zeit, wir probieren das mal aus. Das sind unsere ersten Ziele. Daran würden wir so ein bisschen messen wie wie das Ding erfolgreich ist. Oder das sind besonders kritische Dinge, die wir mal testen möchten. Und so denke ich, wäre wahrscheinlich ein SCRUM-Prozess, der ja auch mit estimation und burn down charts und was weiß ich was allem noch erweitert wird meistens wär dann wahrscheinlich schon ein bisschen zu zu stark strukturiert für diese experimentelle Phase. Sobald die Applikation dann von mehreren Leuten genutzt wird, auch wenn das noch in völligem Playground Modus ist und gestartet wird, dann lohnt sich sicher so ein ein regelmäßiger Rhythmus wo man sich anschaut okay was was läuft jetzt schon? Wer ist irgendwann Gold Platen und wer hat effektiv noch Ideen wie man das verbessern könnte? Bzw gibt es Blocker, die vielleicht auch gar nicht im Team gelöst werden können und da denke ich so ein Sprint-Rhythmus macht schon Sinn, auch einfach um irgendwo seinen Puls zu behalten. Auch ganz wichtig, denke ich, der SCRUM-Prozess definiert ja gerade einen Product Owner. Das heißt, jemand, der sich zuständig fühlt, über das ganze irgendwie Bescheid zu wissen und auch eine Art Richtung mindestens reporten zu können, wo sich das hinbewegt. Und ich glaube, das wäre dann auch die Knack-Nuss, wenn zu viele Leute, zu viele verschiedene Apps dann bauen, die dann mehr oder weniger versanden und die vielleicht auch durch Fluktuation oder andere Themen sich diese Ownership dann verschiebt, ohne dass da jemand nachkommt oder ohne dass geklärt ist, ob das jetzt auch entsprechend eingestampft werden kann. Das

wäre dann wirklich etwas, was durch einen vielleicht Scrum ähnlichen oder Mini Scrum Ansatz vielleicht ein bisschen sichtbarer wird.

Speaker1: [00:22:00] Ich habe hier noch einzelne Themen, die ich gerne kurz mit dir durchgehen möchte, einfach um zu schauen, wie du darauf reagierst oder wie du das Verbindung mit Low-Code-Development bringen kannst. Ist das Sammeln von Anforderungen jetzt wiederum mit dem Bild von dem User, der entwickelt: Wie würdest du sowas organisieren? Im klassischen Scrum Prozess hat man das Business, das die Anforderungen stellt und dann eine gemeinsame Priorisierung. Wie würdest du das im Rahmen von Low Code Development durch den User sehen? Siehst du da Abweichungen oder denkst du, man könnte es ähnlich machen?

Speaker2: [00:22:42] Also ähnlich geht immer, sage ich mal. Ich glaube die Low Code Plattformen eröffnen einen neuen Weg, der davon abhängt, wie stark die Leute auch Zeit und Ressourcen bekommen, sich damit auseinanderzusetzen. Also wenn ich mir vorstelle, dass jemand gesagt bekommt: Hey, du hast einen Nachmittag pro Woche Zeit, an dem Ding zu arbeiten, dann ist das ziemlich ähnlich wie ein Setup, in dem einfach an einer Exceltabelle früher gearbeitet wurde oder irgendein nützliches Fact Sheet, wo man ein bisschen Parameter, Dropdown usw eingeben kann und da kommt irgendeine sinnvolle Aussage raus, die man irgendwo oben bei einem Mailverkehr braucht. Ich glaube, das ist halt eher möglich als vorher, wo man eben nur Excels hatte oder eben nur Papier, um solche Ideen einfach irgendwie zu skizzieren. Man kann den Leuten sagen Hey, nimm dir Zeit, schau dir mal diesen Tutorial an, da kommst du mit der Plattform einigermaßen zurecht. Dann probiere es einfach aus. Und die die wichtigsten Fragen sind dann wie Wo kann sich jemand melden innerhalb einer Organisation, wenn er merkt Ja, ich bin jetzt mit der Plattform gestartet und ich hätte da noch ein paar Ideen, aber irgendwie stehe ich auch und ich weiß nicht, wie weiter. Und ich denke, da die richtigen Ansprechpersonen zu definieren, das wäre dann das Wichtigste in dem Ganzen.

Speaker1: [00:24:06] Wie siehst du das Thema Priorisierung? Würdest du befürworten, dass der Bereich selbst priorisiert? Macht es Sinn, dass es dennoch eine Plattform gibt, wo Vorhaben vorpriorisiert werden?

Speaker2: [00:24:25] Ich glaube, es muss oder was ich mir schön vorstelle und das muss eigentlich jemand aus dem Business beantworten, ist die Möglichkeit so eine Art Package zu buchen innerhalb der Firma, wo man sagt gut, ich kann onboarden, ich werde kurz eingeführt, wie die Plattform funktioniert. Ich bekomme einen Ansprechpartner, wenn ich irgendwo anstehe und umgekehrt fülle ich dafür eine Art Antrag oder eine Ideen-Skizze aus, wo ich auch gewisse Fragen mal beantworten muss, um sicherzustellen, dass ich mit dem Case, den ich mir da vorstelle, nicht gerade in eine Richtung ziele, die die Plattform gar nicht wirklich abdeckt oder die ich nur unter sehr großen Hürden dann irgendwie meistern könnte. Aber umgekehrt, dort dann zu sagen Jawoll, das ist möglich, so einen Rahmen kannst du bekommen, das stelle ich mir sehr, sehr gut vor. Und wie ich zu Beginn gesagt habe, je weniger riesen Struktur und je weniger große Prozesse und Approvals man dafür abholen muss und je mehr man da auch einfach einen Spielplatz mehr oder weniger zur Verfügung bekommt, desto einfacher.

Speaker1: [00:25:35] Ich gehe jetzt ein bisschen auf die Phasen durch. Oder wenn man dann die Entwicklung ansieht, welche Plattformen wären dort sinnvoll? Wir haben ja im Scrum das Daily, das haben wir kurz angesprochen. Oder es gibt zum Beispiel Reviews, Retros. Welche Gefäße siehst du als sinnvoll? Jetzt nicht zwingend auf Scrum bezogen, sondern eher angelehnt. Damit die Zusammenarbeit funktioniert.

Speaker2: [00:26:04] Was auch bei uns sehr effizient ist. Also wir haben ein paar Business Analysten, die sich mit der Plattform auseinandersetzen und da passiert es immer wieder mal, dass ich einfach sporadisch angefragt werde. Ich habe da was gebaut. Kannst du mal kurz deinen Blick drauf werfen und mir einen Tipp geben, wie ich das irgendwie verbessern könnte oder was ich da beachten muss, dass das irgendwie runterläuft. Was auch immer. Und so einen Sparringpartner, der vielleicht mehr die technische Sicht einbringt. Ich glaube, das wäre ziemlich gut so in Form einer Review. Die Checklisten hab ich schon erwähnt, also diese Quality Gates, wenn man sagt, man möchte den Maturity Level einer solchen Applikation hochschieben, dass die Leute auch verstehen. Okay. Um da hoch zu kommen, dass man sagt, das ist jetzt eine eine Scalable App in unserer Company, die auch längerfristig so ein Life Cycle bekommt. Das bedeutet, ich muss irgendwo angeben, welche Daten ich da überhaupt längerfristig speichern möchte, welche Identifier ich brauche oder auch nur schon welche APIs ich anziehe dafür, was da rauskommt oder auch welche Prozesse davon betroffen sind.

Solche Checklisten. Das hilft meistens, damit Leute, die sich dann mit der Idee vor allem auseinandergesetzt haben, seinen Raum bekommen oder auch sehen, was sie noch machen müssen, um das dann auch mit größerem Support betreiben zu können.

Speaker1: [00:27:30] Und was, was mich noch Wunder nimmt: Ich habe ja selbst etwas Erfahrung gemacht mit Power Automate. Ich habe auch eigene Dinge entwickelt. Für das ganze Thema Testing, Produktives Umfeld was sind die Grenzen von Low Code Applikationen in Bezug auf den End User in diesem Themenfeld. Wie siehst du das?

Speaker2: [00:27:53] Ja, also Testing ist eine ganz, ganz wichtige Sache und da unterscheiden wir auch sehr viele Plattformen anhand ihrer Maturität, weil die meisten Plattformen zu Beginn ihres Lebenszyklus da wenig Gewicht drauf legten, weil man möglichst schnell sichtbar, ja schön zu präsentierende Dinge zeigen möchte und gerade diese Qualitätsmaßnahmen dann meist zurück priorisiert werden. Je reifer eine solche Plattform ist, desto besser werden solche Dinge unterstützt. Gerade jetzt, wenn du Power Automate ansprichst, da gibt es auch kleine Features oben drin. Aber jetzt von Microsoft zum Beispiel kommt ja klar die Ansage, dass die Data Governance nicht in der Automate gelöst wird, sondern dort eben mit dem, was man sehen darf, darf man spielen sozusagen. Aber die Dinge, die auf Data Governance Ebene geklärt sein müssen, die sind weiter unten von der effektiven IT, dann zu definieren. Also dort werden die relativ feingranularen Rechte dann vergeben. Und das Schwierigste oder Dümmste, was man machen kann, ist, wenn eine Organisation dann einfach sagt Ja, lass uns Power Automate einsetzen und mal gucken, was passiert und einfach alle Rechte mal freigibt. So produziert man ziemlich schlimme Daten Salate. Das kann ziemlich schief gehen. Und ich glaube da, dass vorgängig zu klären, welche Daten man in so einen Citizen Flow reingeben möchte, das ist sehr wichtig zu beginnen und auch zu kommunizieren oder dass man sich darauf geeinigt hat. Beispiel Kern-Applikations-Daten dürfen genutzt werden, aber dürfen nicht irgendwohin gesendet werden. Zum Beispiel. Oder das wäre so eine einfache Policy, dass man das einfach verbietet oder sagt, das geht nicht outbound, was immer ihr da von unseren API's zur Verfügung gestellt bekommt.

Speaker1: [00:29:52] Und jetzt in dem klassischen Scrum Prozess existiert pro Task oftmals eine Definition of Done. Gibt es das in einem solchen Umfeld auch oder wie siehst du das in deiner Entwicklung?

Speaker2: [00:30:08] Also bei unserer Entwicklung betreiben wir das ganz normal mit einer Definition of Done und das hat dann auch irgendeine Abdeckung auf einem End-to-End-Test oder auch ein Unit Tests, wenn nötig. Code muss gerewied werden usw. Wir haben Branching. Ich glaube, das ist aber auch so, dass wir das ja dann auch als als Rahmen einer professionellen Softwareentwicklung sehen. Ich glaube jetzt, beim Kontext Citizen Developer ist eine Definition of Done viel weniger technisch, aber viel mehr darauf auszulegen: Hast du an die Dinge gedacht, die wichtig sind, auch zu beschreiben? Also nicht nur einfach App as it is, sondern auch den Intent, was möchtest du damit lösen? Wo siehst du die Weiterentwicklung? Was modelliert du da eigentlich genau? Ich glaube, ich sehe da schon eine Definition of Done oder auch so eine Art Guideline, dass gerade Citizen Developer sich die richtigen Fragen stellen und auch Antworten dazu finden können, für die sie nicht in jedem Schritt Guidance brauchen durch irgendeinen IT-Menschen, sondern einfach wissen müssen, dass sie diese Frage auch beantworten müssen.

Speaker1: [00:31:19] Ich muss ganz kurz eine Sekunde meinen Laptop einstecken.

Speaker2: [00:31:22] Kein Problem.

Speaker1: [00:31:35] Das nehme ich dann bei der Transkription raus.

Speaker2: [00:31:38] Kein Ding. Ich glaube, das wichtigste oder die wichtigste Frage, die eigentlich beantwortet werden muss von so einem Citizen Developer ist: Wie siehst du den Life Cycle von dem Ganzen? Also? Und das vielleicht auch repetitiv oder also regelmäßig wieder zusammen sagen: Hallo, du hast die und die App am Laufen, fühlst du dich dafür noch verantwortlich? Siehst du da noch einen Sinn dafür? Solche Dinge einfach irgendwie auch so eine Art Portfolio Management zu orchestrieren? Ich glaube, das wäre dann sehr wichtig. Die größte Gefahr wäre, dass Dinge, die einfach irgendwo rumliegen und so einen mezzo relevanten Charakter dann haben, im dümmsten Fall sogar die einen Daten sehr wichtig sind, die drinstecken, weil noch nicht woanders hin migriert. Und umgekehrt, eine große Ladung Blödsinn, weil nicht mehr nachgeführt, nicht mehr gepflegt und nicht mehr relevant.

Speaker1: [00:32:33] Das heißt die Rolle nach der Entwicklung: Unterscheidet sich diese von einer klassischen Software oder müsste diese noch definiert werden? Oder wie siehst du das?

Speaker2: [00:32:50] Ich würde stark dafür plädieren, dass man nicht versucht, zwischen Entwicklung und Betrieb zu unterscheiden. Sondern da möglichst auch diese Devops-Kultur zu leben. Dass man sagt, man hat eine Ownership über den Life Cycle und auch die aktive Nutzung von dem Ding und eine Ownership über die Entscheidung, was da reinkommt und was da nicht reinkommt. Fatal fände ich es, wenn es irgendwie so einen irgendeinen gibt, der dann so eine kleine App baut, dann einen Automate Flow, der sich dann aber nicht mehr darum kümmert, was damit gemacht wird und ob das überhaupt sinnvoll ist, was da drinsteht oder ob das up to date ist mit neuesten Weisungen. Ich glaube da, das wäre ziemlich schwierig. Und wenn so eine Person oder so ein Team nicht mehr vorhanden ist, dann muss es einen Mechanismus in der Organisation geben, dass auch die entsprechende App, dekommisioniert wird auf einen Graceful Shutdown hin, abgewürgt wird. Mehr oder weniger, um sicherzustellen, dass das Ding nicht einfach irgendwo rumsteht und Datenleichen produziert, die später niemand mehr aufräumen möchte.

Speaker1: [00:33:59] Dann unterscheidet sich dies eigentlich von der klassischen Softwareentwicklung. Da dort ja nicht der komplette Software Development Life Cycle oder die gesamte Software Lebenszyklus abgedeckt wird vom Entwickler, sondern er sich dann wiederum neuen Themen annimmt.

Speaker2: [00:34:19] Ja, es kommt ein bisschen auf den Setup drauf an, wenn ich mir einen Citizen Developer vorstelle, der sagt, ich bin Fachexperten für ein gewisses Thema. Ich arbeite an so einer kleinen App, die für unsere Abteilung oder sonst was irgendwas löst. Dann muss ja der nicht die ganze Zeit die einzige Person sein, die das weitertreibt. Aber wenn er merkt oder sie, dass sie nicht mehr zuständig ist dafür, dann muss es eine Übergabe geben. Und die kann sich nicht einfach darauf beschränken, dass man sagt, wie man das Ding verändert, sondern da muss es auch eine Übergabe geben, damit die ganzen, damit der Betrieb da auch gewährleistet ist. Oder dass klar ist, dass jetzt auch Leute, die die App nutzen, neuerdings eine andere Person anschreiben, die da irgendwas geändert haben möchten oder so was. Und vielleicht. Es kommt halt ein bisschen auf das Unternehmen an, oder es gibt Unternehmen, die sehr

stark getrennt haben zwischen dem Operations, also der ganzen Betrieb einer Applikation oder da wird nichts mehr geändert, höchstens hard fixes. Und der Entwicklung in einem Entwicklungsprojekt, das sind zum Teil auch ganz verschiedene Organisationsstrukturen, die das dann mit KPIs auch unterschiedlich gewichten. Was da passiert damit? Und gerade bei solchen Low Code Entwicklungen, die dann von Seite Business kommen, denke ich, ist dann wichtig, dass man sich dem bewusst ist, dass man das nicht zu stark trennen darf, weil sich die App dann irgendwie sehr komisch weiterentwickelt, voraussichtlich.

Speaker1: [00:35:51] Und welche Grundvoraussetzung muss dann ein End User Developer sozusagen mitbringen, um diese Rolle ausfüllen zu können? Wo startet man dort und wohin müsste man die Leute vielleicht auch bringen?

Speaker2: [00:36:11] Ich glaube es braucht so eine gewisse Motivation seinen Intrapreneurship Gedanken zu sagen: Ich möchte ein bestehendes Problem in meinem Umfeld, möchte ich jetzt technisch lösen oder einen Prozess sichtbar machen, den wir momentan einfach irgendwie über die Tischkante führen, was auch immer. Und diesen Spirit, den muss man mitbringen, weil die erste Hürde wird sein, dass man sich dann mit etwas auseinandersetzen muss dafür. Oder man kommt nicht einfach wie vielleicht in gewissen Promo Clips dazu, dass man einfach nur Plug and Play das Ding startet und dann in drei Minuten ist das Ding gebaut. Sondern man muss sich ja dann wirklich damit auseinandersetzen, wie man das gut macht. Man iteriert ja dann auch. Und das ist sicher auch gut, wenn das nicht eine Person allein macht, sondern auch irgendwo ein Team sich da Gedanken dazu macht das man sich nicht auf Kleinigkeiten beschränkt sondern sagt gut wo, wo sieht auch jemand anderes noch einen Mehrwert, wenn man das so lösen kann? Also dieses Bild, das ist das sind wir immer noch im Business Bereich. Also innerhalb von so einer Abteilung oder einem Interessens Kontext, dass das Ding wirklich einen business value hat. Und dann die technische Stabilität. Ich glaube, das ist dann auch eine Voraussetzung. Was ich vorhin gesagt habe, dass mit diesen Ansprechpartnern oder Hilfs Ressourcen. Guidelines, Checklisten, Links zu irgendwelchen Forum, vielleicht auch dem Anbieter der Plattform, dass da einfach viel da ist, wo sich die Leute selber helfen können. Ich glaube, das ist sehr wichtig, weil die ganze Top down Ausbildung und Orchestrierung, die dann sozusagen die Leute dazu verpflichtet, ja, du musst das jetzt können. Ich glaube das wirkt, wird eher das wird eher Resistenzen erzeugen als diesen Motivationscharakter

dann haben das die Leute auch einfach mal was machen dürfen. Auch ganz wichtig glaube ich, ist ein Hint Richtung Fehlerkultur. Also dass man bewusst sagt, das ist okay, wenn die App irgendwie sich komisch verhält in gewissen Situationen oder die und die Edgecases noch nicht gedeckt sind, weil sie ist ja in diesem Playground, wo das sein darf. Und ich glaube da das Bewusstsein zu schaffen, dass das Ziel wirklich ist, eine Idee, mal zu Ende zu denken und auch so eine Phase zu akzeptieren, wo man vielleicht auch qualitativ sagt, wir schränken das ein, was das alles abdecken muss, aber dafür sehen wir, ob wir damit wirklich Vorteile haben. Ich glaube, das ist auch sehr wichtig.

Speaker1: [00:38:51] Ich finde das Thema einschränken sehr spannend und wie du gesagt hast, es ja Verkaufsfolien, Plug and Play Gedanken. Wo siehst du die Grenzen von low code development durch den End user? Bzw wann braucht es dann wirklich eine höhere Instanz?

Speaker2: [00:39:15] Ich glaube, die Grenzen sind da, wo man die Plattform zutiefst verstehen muss, um sie weiter auszubauen. Also gerade dort, wo dann Custom Code reinkommt oder auch spezifische Adoptionen im UI erfolgen sollen. Je nach Plattform gibt es da unterschiedliche Möglichkeiten und die setzen dann immer wieder auch voraus, dass man versteht, wie man das auch Plattform seitig gedacht hat, dass man diese Extensions oder Hooks oder was auch immer dann nutzen kann. Ich glaube, dort ist dann ein Citizen Developer sehr eingeschränkt. Wenn man dann sagt, das ist nicht nur der Citizen Developer, sondern so ein gemischtes Team, dann sehe ich da ziemlich wenig Grenzen. Da ist es dann mehr eine Frage von: hat man strategisch ein gutes Team zusammengestellt, das wirklich auch dieselben Ziele hat, in dem Team was zu bewegen? Und dann ist mit so einer Low-Code-Plattform wirklich ein cooles Umfeld geschaffen, sage ich mal, wo man auch schnell vorankommt und in diesem Sinn dann auch Resultate zeigen kann, die man mit einem vielleicht eher STANDARD Setup sehr viel später erst wirklich sichtbar macht.

Speaker1: [00:40:28] Ich habe jetzt gehört Zusammenstellungen von zum Beispiel einem Citizen Developer mit professionellen Programmierern hast du jetzt erwähnt. Siehst du darin einen Vorteil?

Speaker2: [00:40:40] Ja. Also gerade wenn wir so über Apps sprechen, die so im Mature Scalable oder auch schon ziemlich ausgereiften Modus dann sind, denke ich,

die Kombination ist die stärkste. Weil dann befeuert sich das gegenseitig oder dann hat man auf einem gemeinsamen Arbeitsumfeld hat man plötzlich Leute, die Know how reinkriegen, die vielleicht sonst nicht zusammen gesprochen hätten. Und umgekehrt ist die Komplexität des ganzen Setups ein bisschen standardisiert. Das heißt, alle Beteiligten haben schon einen gewissen Wiedererkennungseffekt, der mit der Zeit läuft, wo man sich dann hoffentlich auch immer sicherer und wohler fühlt zu sagen: Gut, ich verstehe das zwar nicht, aber deshalb haben wir auch Person X im Team und die kann uns das, kann uns das zeigen oder auch gleich direkt modellieren, wie auch immer und so dann auch umsetzen.

Speaker1: [00:41:34] Du hast vor vorhin erwähnt: den Einbezug von lokalen Daten also mehr so on-Premise Kern Systemen über Schnittstellen. Siehst du da eine Grenze auch in Bezug auf Governance? Wenn man jetzt den End User Developer berücksichtigt, oder?

Speaker2: [00:41:54] Es kommt sehr darauf an, wie die Kern Systeme, also die Governance dieser Systeme schon aufgesetzt ist. Wenn die darauf darauf aus ist, dass sie immer nur von Custom-Projekten angezogen werden, dann sind sie natürlich auch in ihren Abläufen so gebaut, dass sie sich darauf verlassen, dass da in erster Linie mal eine Spezifikation kommt, technisch abgestempelt usw. Und auch in einem anderen Form als dann später ein Citizen Developer die vielleicht nutzen möchte. Verschiedene Plattformen gehen deshalb auch dazu über, dass sie eine Art API Layer definieren oder so ein Data Service Repository oder was auch immer, wo dann die Kern Systeme exponieren, was wirklich freigegeben ist, auch mit entsprechenden Rechten usw. Und sich darin eigentlich schon geklärt hat, welche Daten von wem genutzt werden dürfen.

Speaker1: [00:42:51] Okay.

Speaker2: [00:42:52] Ich glaube das ist bei der On-Prem-Sache geht es ja vor allem auch darum, welche Daten von welchen Plattformen überhaupt genutzt werden dürfen. Also da ist man schnell in diesem ganzen Data Privacy Kontext und dort ist es dann relativ egal, ob jetzt eine weitere App da auch noch Zugriff bekommt, sondern muss man die Frage insgesamt geklärt haben, ob zum Beispiel jetzt im Versicherungskontext Kunden aus der EU ihre Daten auf diesem System haben dürfen oder nicht oder? Das ist dann nicht eine Frage der einzelnen Low Code Apps, sondern auf der ganzen Low-

Code-Plattform. Die muss man am besten vorher, die klärt man am besten vorher und nicht einfach mittendrin.

Speaker1: [00:43:45] Gibt es aus deiner Sicht noch Punkte, welche für meine Forschung interessant sein könnten?

Speaker2: [00:43:52] Also du beschreibst jetzt vor allem den Scrum Prozess oder?

Speaker1: [00:43:56] Eher End-Use-Development.

Speaker2: [00:43:59] Ich glaube, die Frage, die man sich vielleicht noch eine Runde vorher stellen sollte, bevor man sagt: haben wir Citizen Developer im Unternehmen? Ist die Frage: Haben wir Leute im Unternehmen, die sich auch befähigt fühlen, Dinge zu bewegen? Und man kann das eine nicht ohne das andere haben, sage ich mal, wenn grundsätzlich das Gefühl da ist, das sind Leute, die in einer Fachabteilung arbeiten und die machen genau das, wozu sie beauftragt sind und die haben wenig Agency darüber, was, wie, ob sie das verändern dürfen, dann werden sie das auch nicht mit einer Low Code App später machen dürfen, weil sie das nicht können. Also nicht, weil sie nicht können können, sondern weil, weil ihre ganze Arbeitsstelle das irgendwie nicht erlaubt. Das heißt, wenn ich jetzt für ein größeres Unternehmen mal nachschauen müsste, wo würde so eine Low-Code-Entwicklung erfolgreich sein, dann müsste man jetzt schon mal durchgehen und überlegen, wo entstehen jetzt schon Ideen, die dann vielleicht mit Excel gebaut werden oder mit anderen lustigen Krücken? Und wo fühlen sich Leute auch in der Lage, solche Dinge dann im breiteren Kontext zu verteilen oder auch über verschiedene Organisationsformen dann sich da auszutauschen? Und wenn so eine Kultur da ist, dann denke ich, ist der Nährboden sehr gut, dass man auch eine Low-Code-Plattform dann nutzen kann und den Leuten sozusagen ein mächtigeres Werkzeug in die Hand gibt. Wenn das nicht da ist, dann ist das sehr schwierig. Dann sollte man vielleicht mit der Einführung der Low-Code-Plattform mehr noch andere Prozesse mitbedenken und auch mit einführen. Man kann sagen, das kann sich gegenseitig beeinflussen, es kann sich aber auch negativ beeinflussen. Oder wenn plötzlich Leute Angst bekommen. Und ich glaube, das ist noch vor der eigentlichen Delivery von einer App ist das sehr eine kulturelle Frage im Unternehmen, ob man da schon eine gewisse Reife entwickelt hat.

Speaker1: [00:45:54] Wie würdest du die Maturität eines Bereichs dann identifizieren? Du hast jetzt erwähnt Applikation, die bereits entwickelt sind. Gibt es dort noch weitere Merkmale, die du untersuchen würdest, bevor du einen Bereich berücksichtigst?

Speaker2: [00:46:16] Ich glaube. Vielleicht auch eine Art. Eine Art Untersuchung machen, ob Leute, die irgendwo arbeiten und sagen: Hey, hätte ich Budget und eine Entwickler-Bude zur Hand. Das würde ich dann in Auftrag geben. Aber wenn es gerade heute, wo viel über Mobile und Tablet abgewickelt werden kann, wo Gedanken rumstehen, ja wieso haben wir da eigentlich keine App dafür? Oder wieso? Wieso ist das bei uns immer noch so aufwendig? Wieso muss ich dieses Legacy System verwenden usw. Also wo solche Gedanken gedacht und auch besprochen, offen ausgesprochen werden können und auch vielleicht immer wieder mal Requests bei einer IT Abteilung ankommen, die man dann vielleicht aus Budgetgründen nicht umsetzen kann. Aus solchen Richtungen denke ich, an solchen Stellen würde ich nachschauen und sagen: Hey, gibt es da Leute, die so einen Drive haben, aber auch ein Interesse, wirklich auch ein neues Tool kennenzulernen und das mal auszuprobieren, im Wissen, dass es nicht alles löst, aber schlussendlich eben vielleicht etwas verbessern könnte, das sie so noch nicht verbessern konnten, weil eben die Ressourcen nicht verfügbar waren.

Speaker1: [00:47:31] Dann. Denke ich, haben wir wirklich viele Themen abdecken können. Es war für mich extrem spannend. Ich werde die Themen weiter aufbereiten, auch im Sinne der Transparenz. Du warst mein erstes Interview. Ich denke, ich werde meine Fragen entsprechend auch nochmals adaptieren. Eine Frage die sich mir noch stellt: Du hast bereits erwähnte bzw einen Kontakt weitergegeben von einer Person die auch noch interessant wäre zu interviewen. Denkst du das wäre was was ich in Angriff nehmen kann? Bzw wirklich auch. Ist dort bereits abgesprochen, dass ich dort drauf zugehen kann? Oder soll ich einfach mal anklopfen?

Speaker2: [00:48:12] Also das ist der Silvan Stich. Er hat zu mehr die Sicht Projektleitung. Und gerade wenn du dich für den Entwicklungsprozess oder auch den ganzen Flow aus Sicht Projektleitung dann interessierst, kann er das gut abgleichen, weil er sowohl mit Low-Code-Projekten als auch mit nicht nicht Low-Code-Projekten zu tun hat und dort dann auch gegebenenfalls andere Differenzen aufzeigen kann.

Speaker1: [00:48:38] Sehr schön. Ich bedanke mich recht herzlich für die investierte Zeit. Ich weiß wie es ist. Das ist ein zusätzlicher Termin. Der dann am Freitagnachmittag noch wahrgenommen wird. Deshalb an dieser Stelle ein herzliches Dankeschön. Ich werde die Inhalte aufbereiten, meine Thesis schreiben, falls du interessiert bist. Ich darf sie dir sehr gerne zustellen.

Speaker2: [00:48:59] Sehr gerne. Ja, also ich bin auch aktiv auch immer wieder auf der Suche, Leute zu finden, die sich mit den Fragen auseinandersetzen. Wir sehen da ziemlich viel Bewegung im Moment im Markt. Gerade auch mit der Frage oder was ist Make, was ist Buy? Da wird viel überlegt wo man sich strategische Vorteile holt und welche strategischen Risiken mit sich ins Haus holt, Wenn man einfach startet und sich nicht überlegt, was da für Konsequenzen damit dann irgendwann auf dem Tisch liegen. Das sind sehr spannende Fragen, die ich auch gerne immer wieder mal höre, aus anderer Sicht.