

Developer Experience Workshop: Nutzerzentrierte Gestaltung im Platform Engineering

Workshop Agenda

- Einführung in das Konzept eines Platform Teams
- Erläuterung von Developer Experience und User Experience
- Einführung in User Personas
- Hands-on: Erstellung von User Personas





Die vier Team-Typen nach Team Topologies

- Stream-Aligned Team
- Complicated-Subsystem Team
- Enabling Team
- Platform Team

Was ist überhaupt ein Platform Team?

- Ein Platform Team hat die Aufgabe, Stream-Aligned Teams so zu unterstützen, dass diese weitestgehend autonom arbeiten können.
- Die Stream-Aligned Teams sind vollumfänglich verantwortlich für die Entwicklung, den Betrieb und die Wartung ihrer Anwendungen
- Ein Platform Team stellt interne Dienste bereit, die die kognitive Belastung der Stream-Aligned Teams bei der Entwicklung und dem Betrieb ihrer Anwendungen reduzieren.

Welches Verhalten wird von einem effizienten Platform Team erwartet?

- Enge Zusammenarbeit mit Stream-Aligned Teams, um deren Bedürfnisse zu verstehen.
- Anwendung von Fast-Prototyping-Methoden und Einbeziehung von Stream-Aligned Teams für schnelles Feedback zur Nützlichkeit der Dienstleistungen.
- Hochwertige, zuverlässige Dienstleistungen, die regelmäßig auf Aktualität und Nutzbarkeit überprüft werden.
- Nutzung der eigenen Dienstleistungen, um die Benutzererfahrung zu verstehen und zu verbessern.

"I don't think, that you can ever do a good job of creating a platform unless you deeply understand the problem from the perspective of a stream aligned developer."

Dave Farley, Co-Autor von Continuous Delivery



Platform Engineering geht über reine technische Infrastruktur hinaus

- **Plattform als Produkt**

Die Plattform ist nicht nur Infrastruktur, sie ist ein Produkt, das den Bedürfnissen ihrer Nutzer gerecht werden sollte.

- **Nutzung von Produktentwicklungsmethoden**

Platform Engineering nutzt Techniken der Produktentwicklung, einschließlich User Experience Design und User Research, um nutzerzentrierte Produkte zu schaffen.

- **Platform Thinking und Community-Building**

Eine Plattform wird als Gemeinschaft gesehen, nicht nur als technisches Produkt. Sie fördert Zusammenarbeit, Wissensaustausch und gegenseitige Unterstützung.

Was ist Developer Experience?

- Developer Experience (DX) bezieht sich auf die Gesamterfahrung, die ein Entwickler hat, wenn er mit einem Produkt oder einer Plattform interagiert.
- Es ist ein Teilbereich der User Experience (UX), spezialisiert auf die Bedürfnisse und Anforderungen von Softwareentwicklern.
- Ziel ist es, Entwicklern eine reibungslose, intuitive und produktive Erfahrung zu bieten.

"The Developer Experience (DX) describes the experience developers have while using or working on your product. It is a package of positive and also negative feelings."

Merkmale einer guten Developer Experience (DX) können sein:

- **Dokumentation:**

Eine vollständige, klar strukturierte und leicht verständliche Dokumentation. Enthält beispielsweise eine gut gepflegte README-Datei, Tutorials, Anleitungen zur Fehlerbehebung und Beispiele für gängige Anwendungsfälle.

- **API-Design:**

Intuitive und konsistente APIs, die den Prinzipien des sauberen Designs folgen.

- **Fehlermeldungen:**

Hilfreiche und präzise Fehlermeldungen, die Entwicklern dabei helfen, Probleme schnell zu identifizieren und zu beheben.

- **Tooling:**

Leistungsstarke und benutzerfreundliche Tools, die die Entwicklung erleichtern und beschleunigen.

- **Community und Support:**

Aktive Entwicklergemeinschaft und ansprechbarer Support. Möglichkeiten zur Interaktion und zum Austausch mit anderen Entwicklern.

Zeichen von schlechterer Developer Experience (DX) können sein:

Developer Experience

- **Fehlende oder veraltete Dokumentation:**

Fehlende Anleitungen, veraltete Informationen oder unklare Anweisungen, die die Entwickler im Stich lassen.

- **Unkonsistentes API-Design:**

APIs, die nicht intuitiv zu verwenden sind oder die gängige Prinzipien verletzen.

- **Nichtaussagende Fehlermeldungen:**

Fehlermeldungen, die unklar sind oder die Ursache des Problems nicht genau identifizieren.

- **Komplexe Tools:**

Tools, die schwer zu erlernen und zu verwenden sind oder die nicht gut mit anderen Tools integrieren.

- **Mangelnder Support und Community:**

Fehlende Unterstützung und eine nicht engagierte Community, die nicht bei der Lösung von Problemen hilft.

Was sind User Personas?

- User Personas sind fiktive Charaktere, die typische Nutzer und deren Eigenschaften, Bedürfnisse, Motivationen und Verhaltensweisen repräsentieren.
- Sie basieren auf den Erkenntnissen aus der Nutzerforschung und dienen als konkrete, greifbare Darstellungen von Nutzersegmenten.
- User Personas helfen, die Perspektive der Nutzer einzunehmen und Produktentscheidungen an ihren Bedürfnissen auszurichten.

Nutzen von User Personas

- **Klare Ausrichtung**

Helfen, ein gemeinsames Verständnis der Nutzer zu schaffen und sich besser auf deren Bedürfnisse auszurichten.

- **Empathie schaffen**

Ermöglichen es, sich besser in die Nutzer hineinzuversetzen und deren Perspektive einzunehmen.

- **Entscheidungshilfe**

Erleichtern Entscheidungen, indem sie dabei helfen, Optionen aus der Perspektive der Nutzer zu bewerten.

- **Kommunikation erleichtern**

Dienen als gemeinsame Referenzpunkte für die Kommunikation mit anderen Teammitgliedern und Stakeholdern.

User Persona: Python Pete - Data Scientist



Demografie

Leitender Data Scientist, 7 Jahre Erfahrung, spezialisiert auf Python und Maschinelles Lernen

Bevorzugte Tools & Methoden

Python, TensorFlow, Jupyter Notebook, Scikit-learn, Spark, CI/CD.

Zitat

"Ich möchte, dass meine Modelle mit hoher Geschwindigkeit und hoher Leistung laufen, ohne dass ich ein Kubernetes-Zertifikat benötige, um das zu erreichen."

Ziele und Bedürfnisse:

- Wünscht eine robuste Plattform, um Datenverarbeitungsjobs und ML-Modelle zu hosten.
- Sucht nach einer einfacheren Methode, um ML-Modelle zu trainieren, zu testen und zu implementieren.
- Macht sich Sorgen um die Performanz seiner Modelle und will dafür eine effiziente und zugleich zuverlässige Infrastruktur.

Frustrationen:

- Hat mehr "Data" als "Science" im Job, da er viel Zeit damit verbringt, Infrastrukturprobleme zu beheben, anstatt Modelle zu trainieren.
- Leidet unter der Komplexität von Kubernetes und findet, dass es schwer ist, all die Details zu lernen und gleichzeitig ein vollzeit Data Scientist zu sein.
- Hat Schwierigkeiten, seine Modelle von der lokalen Entwicklungsumgebung in die Kubernetes-Umgebung zu übertragen.
- Sieht Kubernetes als eine notwendige Hürde auf dem Weg zur erfolgreichen Modellimplementierung.