MMISP Frontend Implementierungsbericht

Implementierungsphase

MMISP Frontend Team

Table of contents

1. Einleitung

In der Implementierungsphase haben wir den zuvor angefertigten Entwurf umgesetzt und uns weiter am Pflichtenheft orientiert, um die dort gelisteten Anforderungen in angemessenem Maße zu erfüllen.

Im Verlaufe der Implementierung sind jedoch, wie bereits erwartet, einige Anpassungen am Entwurf notwendig geworden.

2. Änderungen an Design und Entwurf

2.1 Workflows

Ursprünglich war geplant, Workflows bei ihren Triggern anzuzeigen.

Wir haben es für sinnvoll erachtet, Workflows ihre eigene Liste und ihre eigene Einzelansicht zu geben. Dort findet sich auch der graphische Editor.

2.2 Neu hinzugefügt

An einigen Stellen hat es sich angeboten neue Elemente zum Design hinzu zu fügen, um die UX zu verbessern.

2.2.1 Mode lock

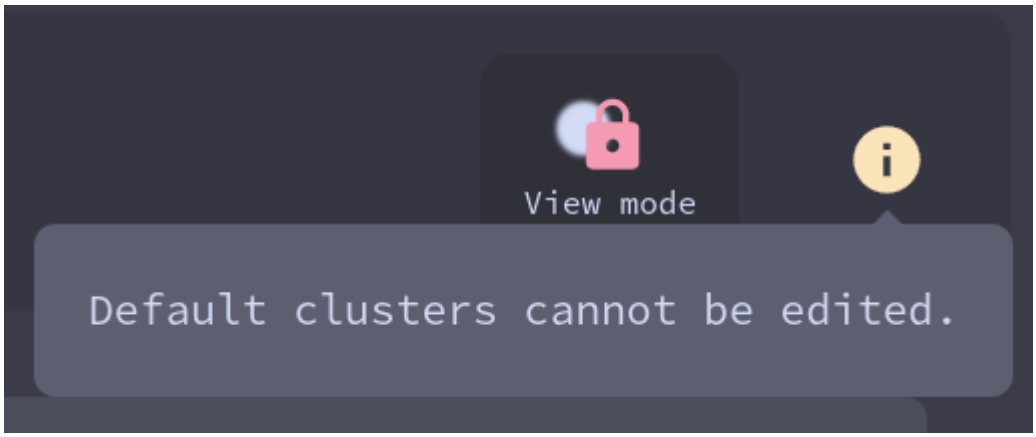
Das Umschalten zwischen view und edit mode wird in bestimmen Fällen gesperrt.

2.2.2 Context info

In bestimmten Fällen ist es für den Nutzer hilfreich kontextuelle Informationen zu erhalten. Zum Beispiel, wenn sich Oberfläche anders verhält als gewöhnlich.

Diese zusätzlichen Informationen können im oberen Menü, rechts neben der Aktionsleiste, mit einem Klick auf das Info-Symbol eingesehen werden.

Das Symbol ist standardmäßig hellgrau und wird farbig, wenn für den aktuellen Kontext Informationen zur Verfügung stehen.



3. Muss- und Wunschkriterien

Im Folgenden wird aufgeführt, in wie fern die im Pflichtenheft aufgeführten Muss- und Wunschkriterien implementiert wurden.

3.1 Musskriterien

In diesem Abschnitt werden Musskriterien besprochen, die **nicht** oder nur teilweise implementiert werden konnten.

3.1.1 API Verschuldet

Die MISP API verhält sich an vielen Stellen anders als spezifiziert oder anderweitig inkonsistent. In einigen Fällen, weicht das tatsächliche Verhalten der API so von unseren Erwartungen ab, dass das mit geplanten Funktionen leider nicht vereinbar war.

Remote Server

Die API gibt keinen Auth key zurück.

Users

Es existiert kein dedizierter Endpunkt zum Filtern von Usern. Zwar können Filter als Teil der URL mitgegeben werden, jedoch erfordert dies eine spezielle Formatierung der Parameter, die sich als ungeplanter Zusatzaufwand nicht mehr im Rahmen der Implementierungsphase umsetzen ließ.

3.2 Wunschkriterien

In diesem Abschnitt werden Wunschkriterien besprochen, die zusätzlich zu den Musskriterien umgesetzt werden konnten.

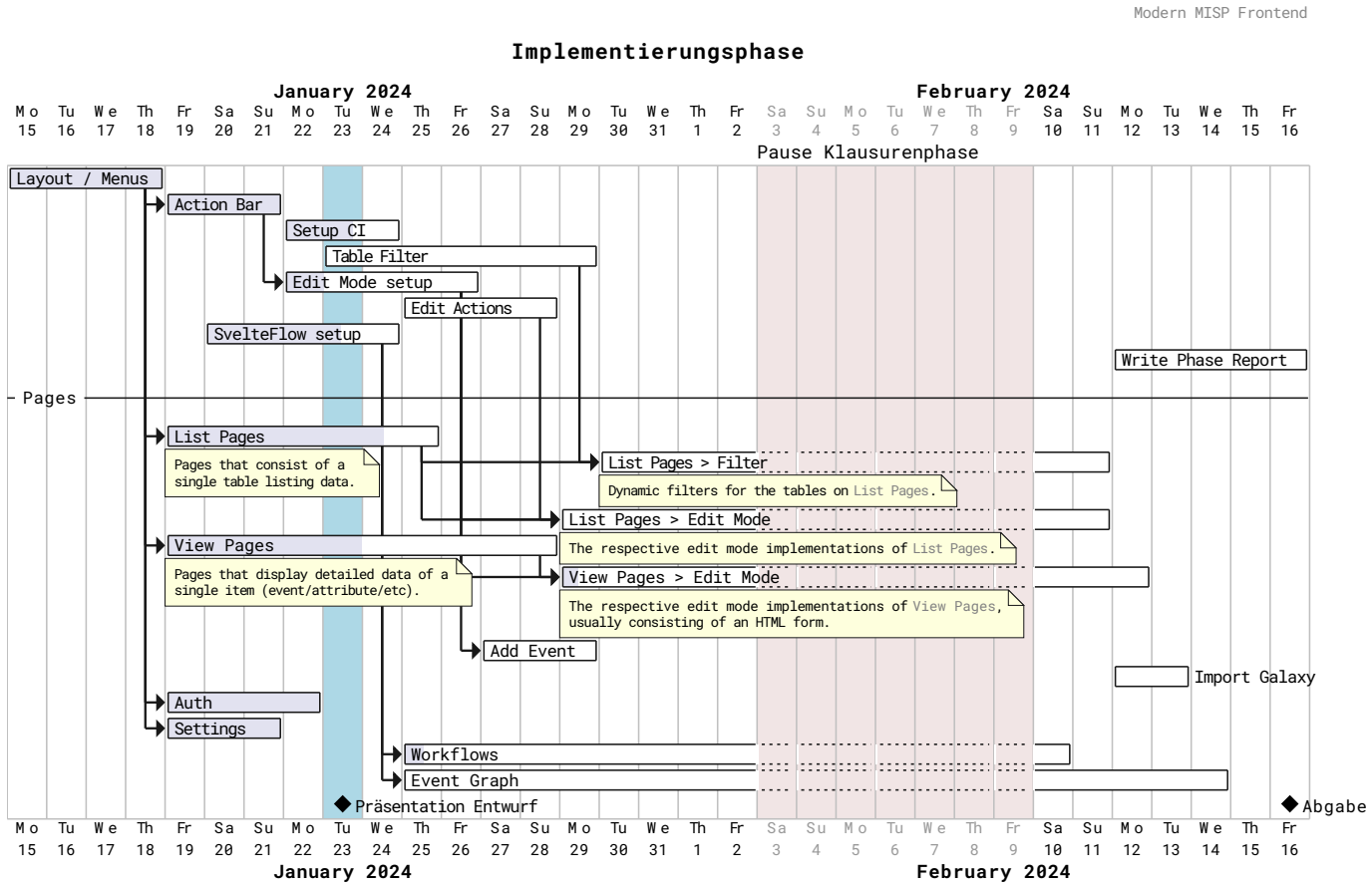
3.2.1 /F004/ MISP Token Auth

Es ist möglich, sich mit einem Auth Key anzumelden.

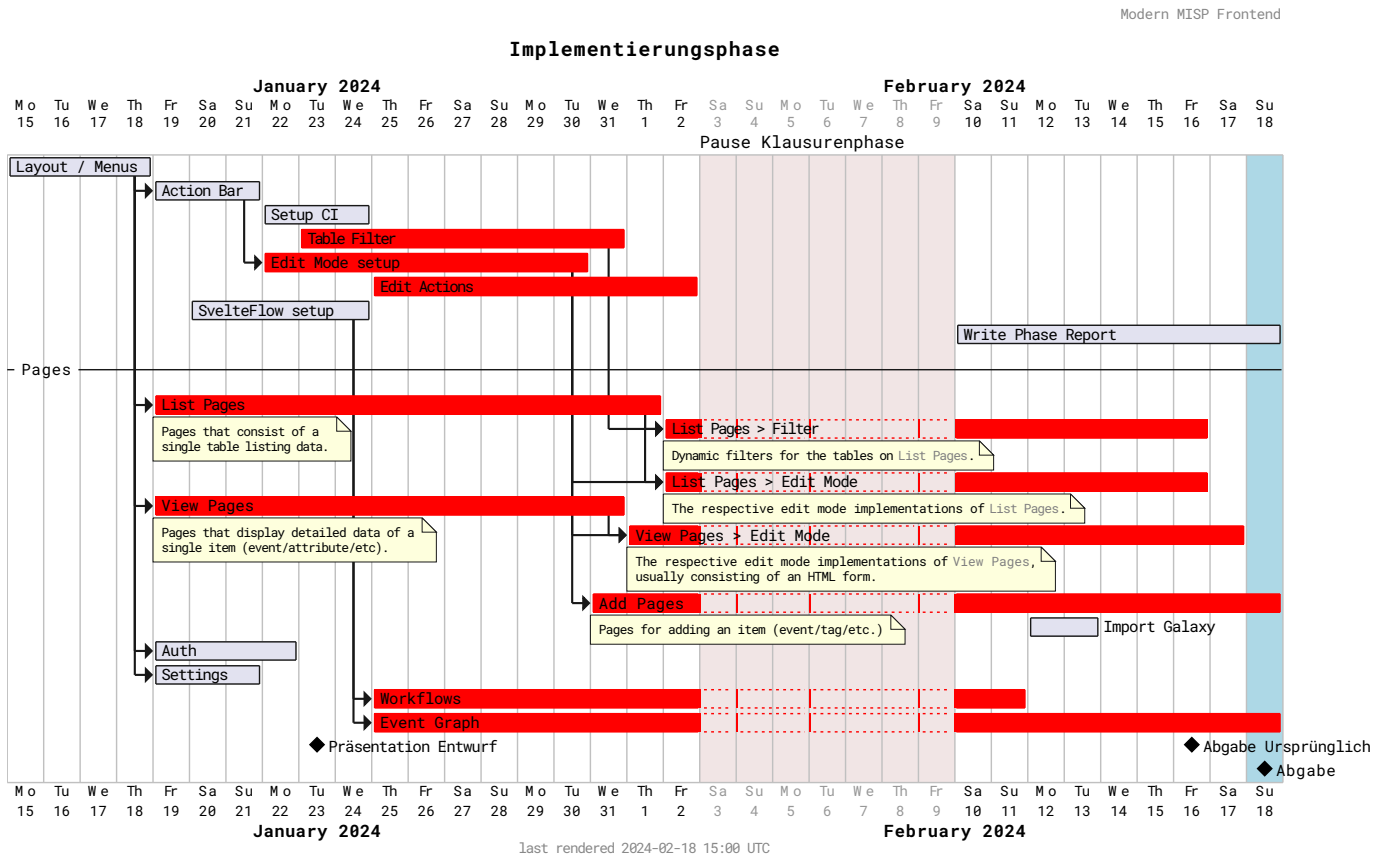
4. Ablauf und Verzögerungen im Zeitplan

In diesem Abschnitt wird der Verlauf der Phase analysiert.

4.0.1 Ursprünglicher Plan

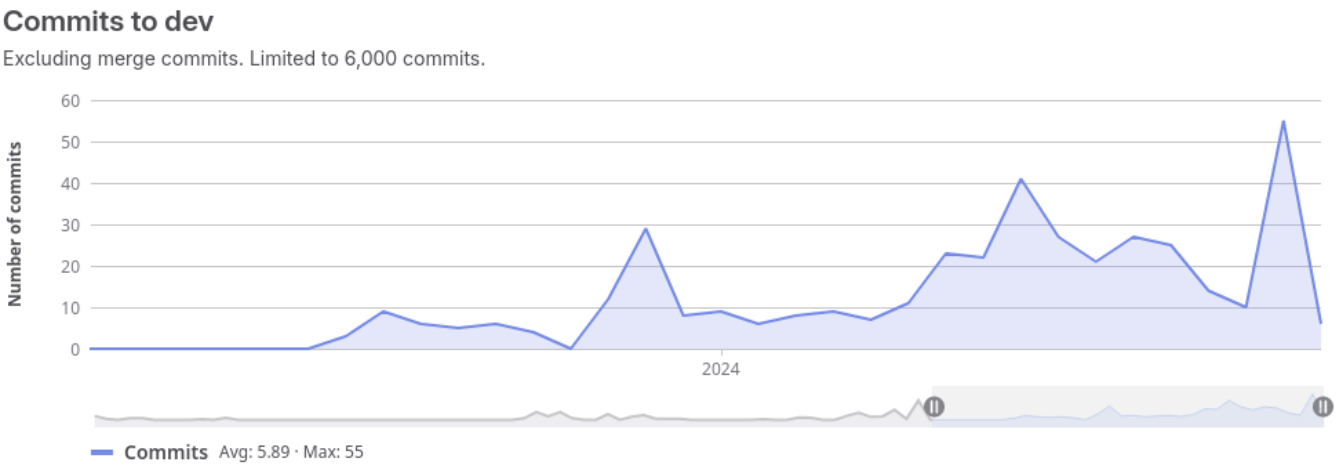


4.0.2 Tatsächlicher Verlauf



An einigen Stellen konnte unser Zeitplan leider nicht wie vorgesehen eingehalten werden.

Rote Balken symbolisieren, dass die entsprechende Aufgabe nicht zum geplanten Zeitpunkt vollständig abgeschlossen war.



4.1 Erste Phasenhälfte

Gerade in den ersten 1-2 Wochen ist die Implementierungsarbeit nur langsam in gang gekommen.

4.1.1 Planung

Die erste Woche war noch geprägt von der Vorbereitung der Entwurfspräsentation, sowie Planung der Implementierungsphase. In dieser Zeit ist das ursprüngliche Gantt-Diagramm entstanden.

Da wir die grundlegende Projektstruktur und einzelne Teile der Funktionalität bereits in der Entwurfsphase umgesetzt hatten, lagen wir zu diesem Zeitpunkt noch recht gut im Zeitplan.

4.1.2 Start der Implementierung

Die Verzögerungen am Anfang haben leider dazu geführt, dass sich auch der Beginn von darauffolgenden Aufgaben verzögert hat.

View mode

Bei den Seiten für den view mode gab es keine größeren Probleme, allerdings hat die Implementierung trotzdem etwas länger gedauert als geplant.

Workflows

Bei den Workflows hat es Anfangs Zeit gekostet, sich in SvelteFlow, die von uns verwendete Bibliothek zum realisieren das graphischen Editors, einzuarbeiten.

4.2 Zweite Phasenhälfte

4.2.1 API Probleme

Besonders als es an die Implementierung des edit mode und der Filter ging, haben sich die Probleme mit der MISP API bemerkbar gemacht.

Die teils falsche und unvollständige API Spezifikation, sowie inkonsistentes Verhalten und wenig aussagende oder irreführende Error-Nachrichten der API haben uns viel nicht eingeplante Zeit und Mühen gekostet.

So mussten wir unter anderem API Endpoints "reverse-engineeren", um sie richtig ansprechen zu können und eigene Typen für deren Rückgabewerte definieren.

4.2.2 Endspurt

In den letzten paar Tagen haben wir nochmal besonders viel gearbeitet, da die Zeit knapp wurde und wir noch alles fertig bekommen wollten.

5. Tests

Da wir eine reine Frontend-Software entwickeln und daher nicht viele Schnittstellen für tatsächliche, testbare Programmlogik haben, lassen sich auch klassische Unit Tests nur bedingt anwenden.

Aus diesem Grund haben wir uns entschieden als sinnvolle Alternative ein Storybook zu verwenden.

5.1 Storybook

Ein Storybook bietet die Möglichkeit alle Teilkomponenten des Frontends isoliert in sog. Stories dar zu stellen. Stories ermöglichen es, das Aussehen und Verhalten einer Komponente manuell zu überprüfen.

Gleichzeitig dienen sie auch als Dokumentation für andere Entwickler. Aus dem Storybook wird ersichtlich aus welchen Teilen sich die Oberfläche zusammensetzt und wie diese - auch intern - funktionieren.

Als Werkzeug zum Generieren unseres Storybooks nutzen wir Histoire.

5.2 Unit Tests

An einigen Stellen, an denen es sinnvoll machbar war, haben wir ergänzend auch noch normale unit tests verwendet.

5.2.1 Test suite

Für unsere unit tests haben wir Vitest genutzt.

5.2.2 Coverage

Die Tests beschränken sich auf wenige, rein logische Funktionen, für die kein aufwändiger DOM mock notwendig war.

Somit sind wir auf eine unit test coverage von knapp über 10% gekommen.

5.3 Weiteres

Um Fehler so frühzeitig wie möglich zu bemerken, haben wir neben unit tests außerdem linter, type checker, sowie Überprüfung auf Einheitliche Code-Formatierung direkt in unsere CI/CD Pipeline integriert und vor jedem Merge auf unseren Haupt-Entwicklungs-Branch erfolgreiche Jobs vorausgesetzt.