

Master-Thesis

zur Erlangung des akademischen Grades
Master of Science (M.Sc.)

an der Hochschule für Technik und Wirtschaft des Saarlandes
im Studiengang Praktische Informatik
der Fakultät für Ingenieurwissenschaften

**Konzeption und Umsetzung einer interaktiven Anwendung zur
Darstellung von Sensordaten unter besonderer Berücksichtigung
der User Experience**

vorgelegt von
Johann Chopin

betreut und begutachtet von
Prof. Dr. Markus Esch

Berlin, Tag. Monat Jahr

Selbständigkeitserklärung

Ich versichere, dass ich die vorliegende Arbeit (bei einer Gruppenarbeit: den entsprechend gekennzeichneten Anteil der Arbeit) selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Ich erkläre hiermit weiterhin, dass die vorgelegte Arbeit zuvor weder von mir noch von einer anderen Person an dieser oder einer anderen Hochschule eingereicht wurde.

Darüber hinaus ist mir bekannt, dass die Unrichtigkeit dieser Erklärung eine Benotung der Arbeit mit der Note „nicht ausreichend“ zur Folge hat und einen Ausschluss von der Erbringung weiterer Prüfungsleistungen zur Folge haben kann.

Berlin, Tag. Monat Jahr

Johann Chopin

Zusammenfassung

Kurze Zusammenfassung des Inhaltes in deutscher Sprache, der Umfang beträgt zwischen einer halben und einer ganzen DIN A4-Seite.

Orientieren Sie sich bei der Aufteilung bzw. dem Inhalt Ihrer Zusammenfassung an Kent Becks Artikel: <http://plg.uwaterloo.ca/~migod/research/beckOOPSLA.html>.

Danksagung

An dieser Stelle möchte ich mich bei all denjenigen bedanken, die mich während der Anfertigung dieser Bachelorarbeit unterstützt und motiviert haben.

Zuerst gebührt mein Dank Herr Prof. Markus Esch die meine Masterarbeit betreut und begutachtet hat. Für die hilfreichen Anregungen und die konstruktive Kritik bei der Erstellung dieser Arbeit möchte ich mich herzlich bedanken.

Ebenfalls möchte ich mich bei meinen Kollegen und Kolleginnen der Firma Dryad bedanken (insbesondere mein Kollege Dinesh Priyashantha und mein Vorgesetzter Cherian Mathew), die mir während der Schreibphase meiner Masterarbeit wertvollen Input gegeben haben und mir Einblicke in die Unternehmenskultur gewährt haben.

Schließlich wäre es nachlässig, meine Familie nicht zu erwähnen, insbesondere meine Eltern. Ihr Glaube an mich hat mich während dieses Prozesses bei Laune und Motivation gehalten.

Chopin Johann

Inhaltsverzeichnis

1 Einleitung	1
1.1 Motivation	1
1.2 Aufgabenbeschreibung und Ziele	1
1.3 Aufbau der Thesis	1
2 Grundlagen	3
2.1 Silvanet-Technologie	3
2.2 Anwendungsentwicklung mit Webtechnologien	4
2.2.1 Single Page Application (SPA)	5
2.2.2 TypeScript	5
2.2.3 Webpack	6
2.2.4 Angular	7
2.2.5 Npm	7
2.2.6 SASS	8
2.2.7 Angular Routing	8
2.2.8 Design System (PrimeNg)	9
2.3 Tools und Arbeitsabläufe	10
2.3.1 Scrum	11
2.3.2 Tools	11
2.3.3 Developer Experience	13
2.4 Ergonomie und User Experience	14
2.4.1 Ergonomie	14
2.4.2 User Experience	15
3 Analyse	17
3.1 Zielsetzung	17
3.2 Ergonomische Inspektion	17
3.2.1 Guidance	18
3.2.2 Workload	22
3.2.3 Explicit Control	23
3.2.4 Adaptability	24
3.2.5 Error Management	26
3.2.6 Consistency	27
3.2.7 Schlussfolgerungen	28
3.3 Benutzertests	28
3.3.1 Festlegung der Testabdeckung	28
3.3.2 Methoden für Usability-Tests	32
3.4 Benutzerumfrage über die Feuerwehr-Community	32
4 Konzeption	33
4.1 Verwenden eines Dashboards für schnelle Entscheidungsfindung	33
4.2 Verhalten einer interaktiven Karte beim Zoomen und Auszoomen	33
4.3 Präsentation von Daten in Stresszeiten	33

5	Implementierung	35
5.1	Entwicklung eines Benutzers Fragebogens	35
5.2	Geschichtete Darstellung auf interaktiver Karte	35
6	Evaluation	37
7	Diskussion	39
8	Zusammenfassung und Ausblick	41
8.1	Zusammenfassung	41
8.2	Ausblick	41
	Literatur	43
	Abbildungsverzeichnis	45
	Tabellenverzeichnis	46
	Listings	46
	Abkürzungsverzeichnis	47
.1	Ergebnis des zweiten Ideationszyklus mit der Lotus-Blossom-Methode . .	51
.2	Notizen aus der ergonomischen Inspektion der Webschnittstelle (Englisch)	54

1 Einleitung

1.1 Motivation

Jedes Jahr und immer häufiger brechen auf der ganzen Welt immer intensivere und verheerendere Waldbrände aus. Nur in Europa brannten 2021 mehr als 1.113.464 ha Wald in mehr als 39 Ländern [5]. Um das Risiko von Waldbränden zu verringern, hat das junge Start-up-Unternehmen Dryad Sensoren entwickelt, die eine ultraschnelle Erkennung von Waldbränden ermöglichen.

Diese Sensoren, die über mehrere Aktionsstandorte verteilt sind, erfassen verschiedene Arten von Daten wie Temperatur, Luftfeuchtigkeit und das Vorhandensein von Gasen, die bei einer Verbrennung entstehen. Ein Webinterface ist verfügbar, um die verschiedenen Daten, die von diesen Sensoren gesendet werden, zu sehen. Da die Kunden jedoch nicht alle Computerspezialisten sind, muss die Benutzeroberfläche so intuitiv wie möglich sein.

Derzeit kommt das Produkt mit einer Anleitung, die erklärt, wie man die Webanwendung nutzt. Dies deutet auf ein großes Problem mit der Nutzererfahrung hin. Meine Aufgabe in dieser Arbeit ist es daher, die Schwachstellen der aktuellen Schnittstelle zu identifizieren und Verbesserungen vorzuschlagen und zu entwickeln.

1.2 Aufgabenbeschreibung und Ziele

In meiner Thesis bei Dryad geht es darum, die entscheidenden Bereiche der Webanwendung, die die verschiedenen Sensoren verwaltet, zu erkennen, die als schlechte Benutzererfahrung angesehen werden. Die Schwerpunkte der Arbeit lassen sich wie folgt beschreiben:

- Identifizierung von ergonomischen Mängeln basierend auf einer Auditierung der gesamten Schnittstelle.
- Entwicklung und Durchführung von Benutzertests, die es ermöglichen, User-Stories so zu definieren, dass sie nach Prioritäten geordnet werden können.
- Forschung und Entwicklung zu den verschiedenen interaktiven Karten der Anwendung, die es dem Nutzer erleichtern, seine Standorte und Sensoren sowie deren Status zu entdecken.
- Forschung zur Verbesserung der Datenpräsentation in Stresssituationen

1.3 Aufbau der Thesis

In Kapitel 2 werden zunächst die grundlegenden Konzepte vorgestellt, die für das Verständnis der These hilfreich sind. In Kapitel 3 werden dann die Anforderungen vorgestellt, die für diese Thesis definiert wurden.

[TODO]

2 Grundlagen

2.1 Silvanet-Technologie

Das Startup namens Dryad¹ mit Sitz in Berlin hat sich zum Ziel gesetzt, die Zahl der Waldbrände zu verringern, indem es sie so schnell wie möglich entdeckt und die Behörden warnt. Ein Team für die Entwicklung von Hardware und eingebetteten Systemen beschäftigt sich mit der Entwicklung kleiner solarbetriebener Sensoren, die Veränderungen in der Zusammensetzung der Umgebungsluft erkennen können. Wenn eine Veränderung festgestellt wird, wird ein Scan initiiert und eine eingebaute künstliche Intelligenz bestimmt, ob es sich um die Verbrennung handelt. Der Vorteil der künstlichen Intelligenz besteht darin, dass sie mehrere Variationen als nur die einfache Verbrennung erkennen kann. So kann man feststellen, ob es sich nur um einen Diesel-LKW handelt, der in der Nähe vorbeifährt, oder ob es sich um ein Feuer aus einem bestimmten Holz handelt, etc.

Diese Sensoren werden mithilfe der LoRaWAN-Technologie miteinander verbunden, die es ermöglicht, ein großes Netzwerk mit geringem Stromverbrauch aufzubauen. Die LoRaWAN-Netzwerkarchitektur wird in einer sternförmigen Topologie eingesetzt, in der Gateways Nachrichten zwischen Endgeräten und einem zentralen Netzwerkservier weiterleiten

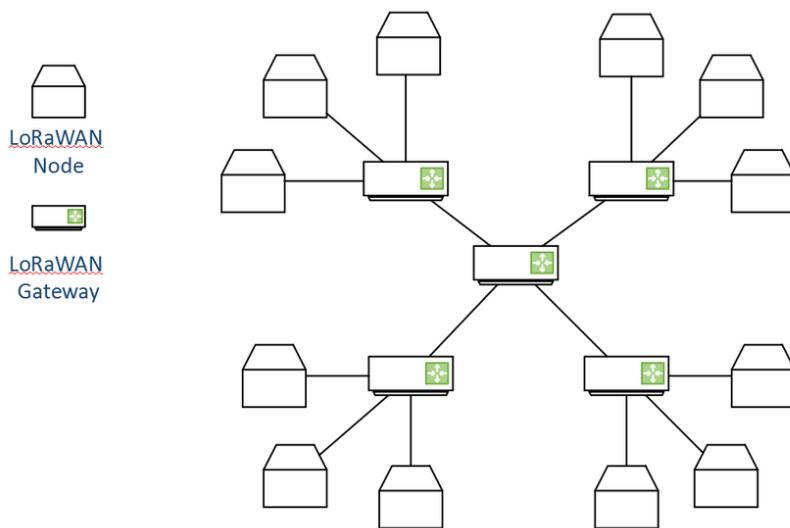


Abbildung 2.1: Schematisierung einer Star-of-Stars-Netzwerk-Architektur, bestehend aus *Nodes* und *Gateways*, die vom LoRaWAN-Standard verwendet wird[10]

In der Silvanet-Architektur von Dryad entsprechen die Nodes den Sensoren, die Waldbrände erkennen sollen. Wir nennen sie Sensors. Silvanet verwendet dann 2 Arten von Gateways, um die Informationen von den Sensoren nach außerhalb des Netzwerks zu leiten (Dryads Cloud-Backend-Service). Der erste Typ heißt *Mesh-Gateway* und ermöglicht die Netzwerkabdeckung über große Entfernung zwischen den Sensoren und andere

¹<https://www.dryad.net/>

2 Grundlagen

Mesh-Gateways. Die zweite Art von Gateways, die *Borders-Gateways*, übertragen die Daten der Mesh Gateways nach außerhalb des LoRaWAN-Netzwerks, indem sie mit dem Backend-Service von Dryad kommunizieren. Eine Ansammlung von *Sensoren*, *Border-Gateways* und *Mesh-Gateways* wird als *Site* bezeichnet.

Nach der Verarbeitung können die verschiedenen Informationen der zu Standorten zusammengefassten Sensoren abgerufen und in der Webanwendung von Dryad angezeigt werden. Mit dieser Anwendung kann sich ein Benutzer anmelden, die Installation von Sensoren an einem Standort planen, die verschiedenen gesammelten Daten in Echtzeit verfolgen und die Art einer Waldbrandwarnung im Detail abrufen.

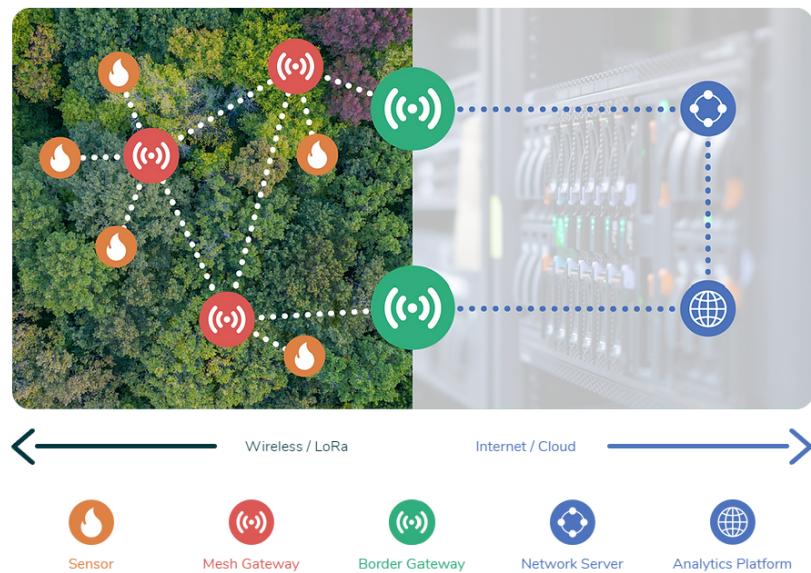


Abbildung 2.2: Schematische Darstellung des LoRaWAN-Netzwerks, das die verschiedenen Sensoren mit dem Cloud-System verbindet.



Abbildung 2.3: ein luftschnüffelnder Sensor, der ein Silvanet-Netzwerk bildet

2.2 Anwendungsentwicklung mit Webtechnologien

Die Sensorverwaltungsanwendung von Dryad wird mithilfe von Webtechnologien (html, css, javascript) und zahlreichen Tools entwickelt, um eine Webanwendung zu erstellen, die auf mehreren Plattformen (Browser, ios, Android) läuft und dabei nur einen einzigen Quellcode hat. Es wurde beschlossen, dass die erste Version der Anwendung eine Single Page Application (SPA) sein sollte. Derzeit gibt es viele Tools, die das Erstellen einer SPA

erleichtern. Innerhalb von Cloud-Team von Dryad wurde beschlossen, die folgenden zu verwenden:

2.2.1 Single Page Application (SPA)

Eine Single Page Application ist ein Website-Design-Ansatz, bei dem der Inhalt jeder neuen Seite nicht durch das Laden neuer HTML-Seiten, sondern dynamisch durch die Fähigkeit von JavaScript, die Document Object Model (DOM)-Elemente auf der bestehenden Seite selbst zu manipulieren, generiert wird.

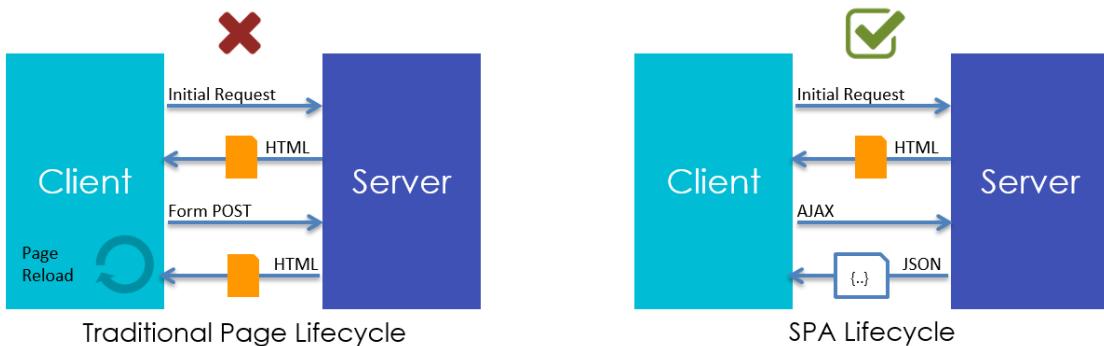


Abbildung 2.4: <https://www.tothenew.com/blog/optimization-of-angularjs-single-page-applications-for-web-crawlers/>

Pros:

- Die traditionellen HTML-Seiten nehmen beim Hin- und Herbewegen zum Server viel Zeit in Anspruch. Dies erforderte das Nachladen der großen Menge ähnlicher Daten, während das neue SPA die Notwendigkeit der Kommunikation mit HTML-Tags zum Server mildert. SPA lädt nur einmal mit den HTML-Tags, während die nächsten Anfragen auf das teilweise Laden der Seite mittels AJAX und JSON erfolgen. Dies führt zu einer Einsparung von Bandbreite.
- Die SPA-Seite wird schneller geladen und benötigt aufgrund der geringeren Datentransaktion weniger Bandbreite. Das Seitendesign und UX werden besser und funktionieren erstaunlich gut mit den langsamen Verbindungen.

Cons:

- Die traditionellen HTML-Seiten sind gut für den SEO, während die SPA schwer von den Benutzern gecrawlten werden kann, weil die Webcrawlers nicht wissen, wie sie mit dem Javascript umgehen sollen. Die Lösung besteht darin, die roboterspezifische HTML-Seite zu kodieren, was wiederum zu Wartungsproblemen führen kann.
- Manchmal braucht die SPA viel Zeit, um das Bündel aus CSS und Javascript zu laden, was das erste Laden der Seite verzögert.

2.2.2 TypeScript

Die Entwicklung einer Webanwendung kommt heutzutage ohne Javascript kaum noch aus. Die Hegemonie dieser Sprache wird nicht mehr angezweifelt, und ihre Gemeinschaft ist eine der wichtigsten in der Welt der Webentwicklung, was sicherlich auf ihre inhärente Flexibilität zurückzuführen ist. Es hat jedoch einige Einschränkungen, wenn es um die

2 Grundlagen

Entwicklung komplexer Anwendungen geht, und aus diesem Grund kommt TypeScript ins Spiel.[8]

TypeScript² ist eine Programmiersprache, die von Microsoft im Jahr 2012 entwickelt wurde. Sein Hauptanliegen ist es, die Produktivität der komplexen Anwendungsentwicklung zu verbessern.

Es handelt sich um eine Open-Source-Sprache, die als höhere Schicht von Javascript entwickelt wurde. Jeder in Javascript gültige Code ist auch in TypeScript gültig. Die Sprache führt jedoch optionale Funktionen wie Typisierung oder objektorientierte Programmierung ein. Um diese Funktionen nutzen zu können, ist keine Bibliothek erforderlich, aber nur das Tool für die TypeScript-Kompilierung. Somit wird der ausgeführte Code ein Javascript-Äquivalent des kompilierten TypeScript-Codes sein.

Dieser Sicherheitstyp hat ihn bei Javascript-Entwicklern sehr beliebt gemacht, wie die Stackoverflow Survey 2022³ beweist:

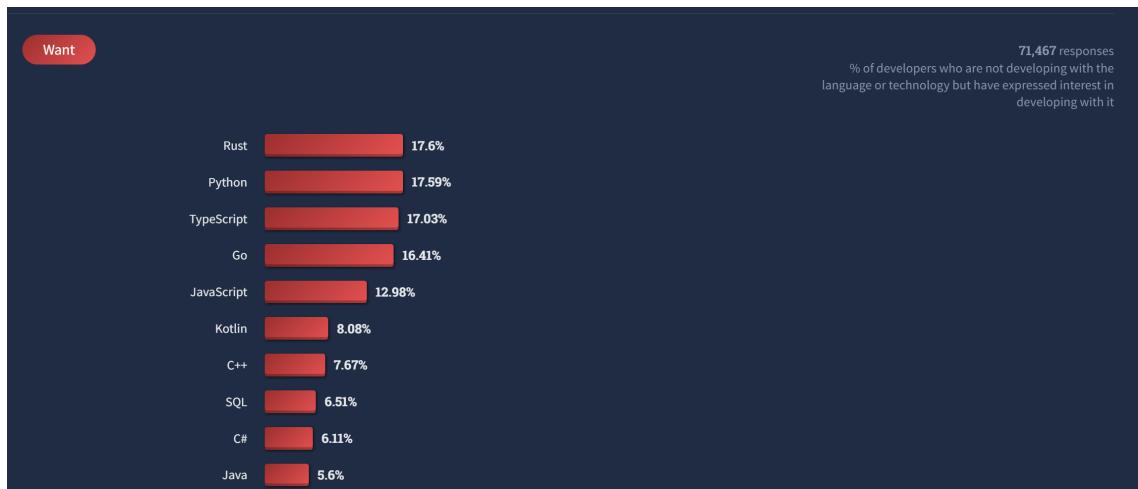


Abbildung 2.5: Liste der beliebtesten Programmier-, Skript- und Auszeichnungssprachen im Jahr 2022

2.2.3 Webpack

Webpack⁴ ist ein Modul-Bündler. Sein Hauptzweck besteht darin, JavaScript-Dateien für die Verwendung in einem Browser zu bündeln, aber es ist auch in der Lage, so gut wie jede Ressource oder jedes Asset zu transformieren, zu bündeln oder zu paketieren⁵.

Es gibt viele verschiedene Bündler wie Parceljs⁶, Rollupjs⁷ oder Browserify⁸. Das von uns verwendete Web-Framework (das im nächsten Abschnitt vorgestellt wird) kommt jedoch fertig mit einer webpack-Konfiguration. So wird webpack neben seiner Reife, seinen zahlreichen Plug-ins und seiner umfangreichen Konfiguration auch für den Aufbau unserer Javascript-Anwendung verwendet.

²<https://www.typescriptlang.org/>

³<https://survey.stackoverflow.co/2022/#most-loved-dreaded-and-wanted-language-want>

⁴<https://github.com/webpack/webpack>

⁵Offizielle Webpack-Dokumentation: <https://github.com/webpack/webpack#webpack>

⁶<https://parceljs.org/>

⁷<https://github.com/rollup/rollup>

⁸<https://github.com/browserify/browserify>

2.2.4 Angular

Mit Javascript können wir ein HTML DOM leicht manipulieren. Das Hinzufügen, Ändern oder Löschen eines Node zur Laufzeit, ohne die Seite neu zu laden, bietet dem Benutzer eine sehr dynamische und leichte Schnittstelle, die dem ähnelt, was er in einer nativen Anwendung sehen kann.

Leider kann die Entwicklung einer Webanwendung allein mithilfe von Javascript sehr viel Zeit in Anspruch nehmen und schwer skalierbar werden. Aus diesem Grund haben sehr marktführende Unternehmen wie Google oder Facebook ihr Javascript-Framework entwickelt, um dynamische Schnittstellen einfach zu erstellen.

Ein Framework ist eine semi-vollständige Applikation. Es stellt für Applikationen eine wiederverwendbare, gemeinsame Struktur zur Verfügung. Die Entwickler bauen das Framework in ihre eigene Applikation ein und erweitern es derart, dass es ihren spezifischen Anforderungen entspricht. Frameworks unterscheiden sich von Toolkits dahingehend, dass sie eine kohärente Struktur zur Verfügung stellen, anstatt einer einfachen Menge von Hilfsklassen[6].

Einige der beliebtesten sind React⁹ von Facebook, Angular¹⁰ von Google und Vue¹¹.



Abbildung 2.6: Stackoverflow Survey 2022: Beliebteste Web-Frameworks

Angular wurde aufgrund seines umfassenden Designs, seiner großen Nutzergemeinde und der Tatsache, dass das Team bereits über fundierte Kenntnisse in diesem Tool verfügte, als Framework ausgewählt.

2.2.5 Npm

Es ist derzeit undenkbar, eine Webapplikation von Grund auf neu zu entwickeln. Es gibt eine Vielzahl von Open-Source-Bibliotheken, die von verschiedenen Entwicklern entwickelt wurden und mit denen unsere Anwendung schnell und einfach erstellen werden kann. Deshalb ist es notwendig, ein Tool zu benutzen, um die verschiedenen Bibliotheken, die verwendet werden, zu organisieren.

Node Package Manager (NPM) ist das Zentrum der gemeinsamen Nutzung von JavaScript-Code und mit mehr als einer Million Paketen und 11 Millionen Entwickler weltweit die größte Software-Registry der Welt¹². Aufgrund dieses guten Renommes beschloss das Team, NPM zur Verwaltung der verschiedenen Bibliotheken einzusetzen.

⁹<https://github.com/facebook/react/>

¹⁰<https://github.com/angular/angular>

¹¹<https://github.com/vuejs/vue>

¹²Npmjs offizielle Website: <https://www.npmjs.com/>

2 Grundlagen

2.2.6 SASS

Angular erlaubt es, das DOM einer Webanwendung leicht zu manipulieren, bleibt nur noch, sie zu stylen. Es wird dafür der Stylesheet-Kernsprachen des World Wide Webs verwenden, Cascading Style Sheets (CSS).

Leider kann CSS schnell überflüssig und schwierig zu handhaben werden, wenn es darum geht, mehrere Dutzend Elemente gleichzeitig zu bearbeiten. Um dieses Problem zu überwinden, wurde der CSS-Präprozessor Syntactically Awesome StyleSheets (SASS) geschaffen, der die Verwendung von Variablen, Schleifen und Funktionen erlaubt, die in CSS umgesetzt werden.

Die folgende Abbildung zeigt die gleiche Style-Deklaration mit CSS und SASS:

```
ul {  
    background-color: red;  
}  
  
ul.hide {  
    opacity: 0;  
}  
  
ul.hide li {  
    text-decoration: underline;  
}
```

Listing 2.1: Beispiel für die Gestaltung einer Liste mit css

```
ul {  
    background-color: red;  
  
    &.hide {  
        opacity: 0;  
  
        li {  
            text-decoration: underline;  
        }  
    }  
}
```

Listing 2.2: Beispiel für die Gestaltung einer Liste mit scss

Es wurde daher beschlossen, diesen css-Prozessor im Projekt zu verwenden. Außerdem bietet Angular eine bereits fertige Sass-Kompilation an.

2.2.7 Angular Routing

Der Antrag wird also von einer einzigen Seite generiert, was aber nicht bedeutet, dass ein Wechsel der URLs nicht möglich ist. Tatsächlich kann eine SPA dank Javascript auf verschiedenen URLs laufen. Angular bringt ein eigenes Routing-System namens Angular Routing¹³ mit, das es ermöglicht, bestimmte Komponenten auf der Grundlage der URL anzuzeigen. Das Projekt wird diese Lösung verwenden, um das Routing der Anwendung zu verwalten.

¹³<https://angular.io/guide/routing-overview>

2.2.8 Design System (PrimeNg)

Eine Benutzeroberfläche besteht aus vielen kleinen Elementen, die als User Interface (UI)-Komponenten bezeichnet werden. Diese sind oft mit Schaltflächen, Popups oder Eingabefeldern versehen, die in der gesamten Anwendung wiederverwendet werden können. Die Organisation einer Sammlung dieser wiederverwendbaren Komponenten, so dass ihre Verwendung durch klare Standards geleitet wird, die es ermöglichen, viele verschiedene Anwendungen zu bauen, nennt man ein *Design System*[4].

Zu den bekanntesten Design Systems, die wir finden können, gehören:

- Atlassian Design System¹⁴
- Carbon¹⁵: IBM's open-source Design System
- Polaris¹⁶: Shopify's Design System

Dryad hat auch im Blick, ein Designsyste zu entwerfen, das an die verschiedenen zukünftigen Schnittstellen angepasst ist. So sind die wichtigsten Richtlinien für Ikonografie, Illustrationen und Markenimage bereits festgelegt. Jetzt fehlt nur noch die Sammlung von UI-Komponenten, die diese Richtlinien anwenden. Da Dryad aus einem kleinen Team mit engen Deadlines bestand, war es nicht möglich, Komponenten aus dem Nichts zu entwickeln. Deshalb greift das Team auf eine Sammlung bereits entwickelter Komponenten zurück, die eine nahezu sofortige Entwicklung der Schnittstelle ermöglichen. Die Wahl fiel auf die PrimeNg¹⁷ Library, die mehr als 80 UI-Komponenten, 200 Icons, eine große Auswahl an Templates und vor allem eine enge Kompatibilität mit dem verwendeten Framework Angular bietet.

```
<p-paginator
  [rows]="numberOfRows"
  [totalRecords]="totalRecords"
  (onPageChange)="pageChange($event)"
></p-paginator>
```

Listing 2.3: Beispiel für die Verwendung der Komponente *p-paginator*, mit der eine Schnittstelle mit Paging einfach verwaltet werden kann.



Abbildung 2.7: Rendern auf die Schnittstelle einer *p-paginator*-Komponente, wie in Abbildung 2.3 deklariert.

¹⁴<https://atlassian.design/>

¹⁵<https://github.com/carbon-design-system/carbon>

¹⁶<https://polaris.shopify.com/>

¹⁷<https://www.primefaces.org/primeng/>

2 Grundlagen

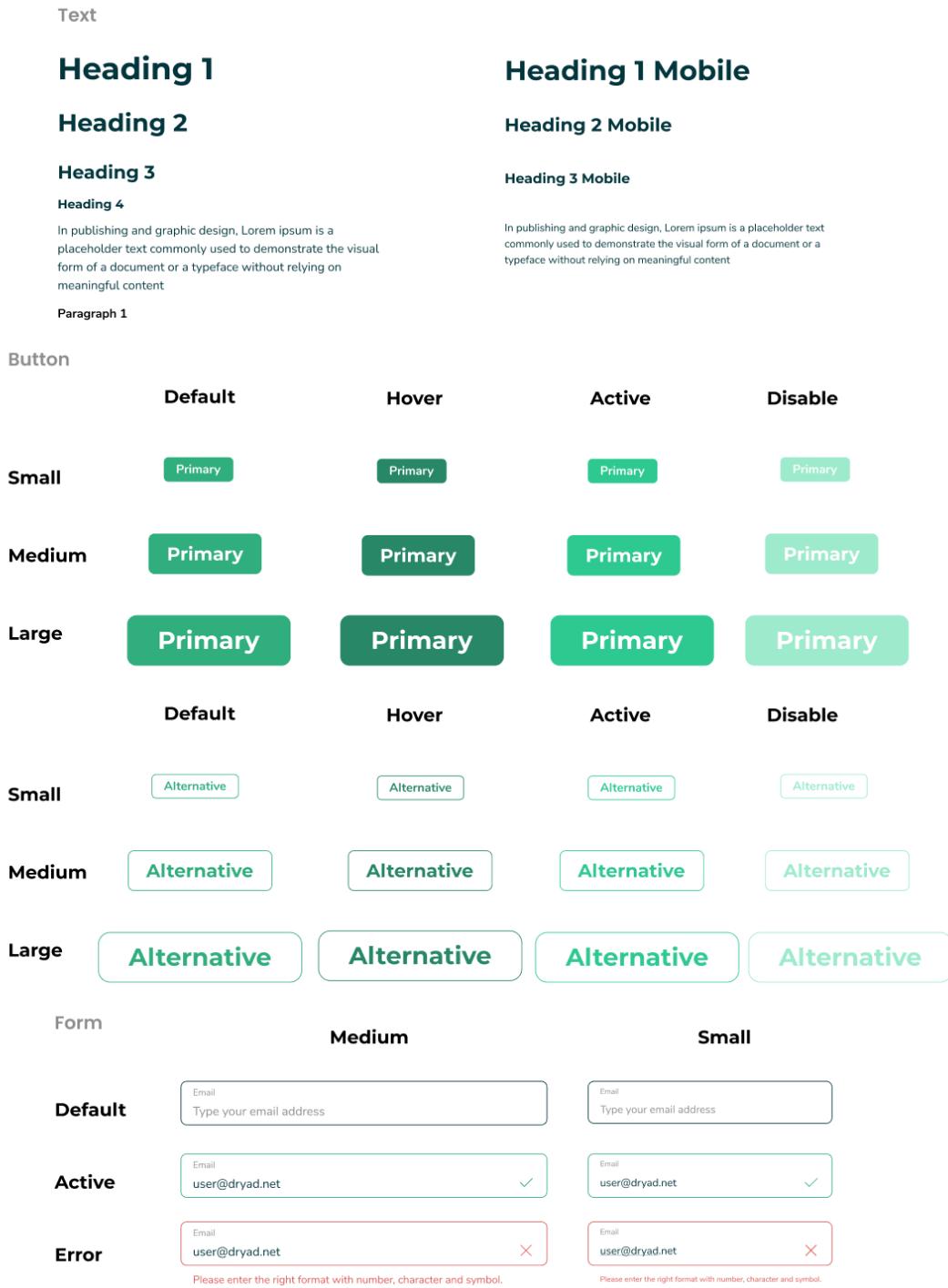


Abbildung 2.8: Überblick über einige UI-Komponenten des Designsystems von Dryad entwickelt auf der Designanwendung Figma

2.3 Tools und Arbeitsabläufe

Innerhalb des Cloud-Team wird jedes IT-Projekt agil nach der Scrum-Methode entwickelt. Scrum ist ein Framework für die Entwicklung komplexer Softwareprodukte. Sie wird von ihren Schöpfern als "ganzheitlicher, iterativer Rahmen definiert, der sich auf gemeinsame Ziele konzentriert, indem er produktiv und kreativ Produkte von höchstmöglichen Wert

liefert"[11]. Es basiert auf der Unterteilung eines Projekts in "Sprint Boxen", sogenannte Sprints, die von ein paar Stunden bis zu einem Monat dauern können (in unserem Team dauert es eine Woche)

2.3.1 Scrum

Das Cloud-Team wendet mehrere Scrum-Methoden zur Verbesserung der Software-Entwicklung an:

2.3.1.1 Sprint Meeting

Zu Beginn jedes Sprints trifft sich das Cloud-Team, um die verschiedenen Aufgaben für den nächsten Sprint zu planen und diese in Tickets zu organisieren. Sie finden jeden Montag um 10 Uhr morgens statt und dauern durchschnittlich 30 Minuten.

2.3.1.2 Retro Meeting

Diese Treffen finden einmal pro Woche statt und ermöglichen jedem Teamentwickler, die guten und schlechten Punkte der vergangenen Woche zum Ausdruck zu bringen. Dies gibt Rückmeldung über die Stimmung im Team. Sie finden jeden Donnerstag um 13.00 Uhr statt und dauern durchschnittlich 45 Minuten.

2.3.1.3 Kanban board

Die Verfolgung unserer Sprint-Tickets erfolgt über eine Kanban-Tafel. Es handelt sich dabei um ein agiles Projektmanagement-Tool, das dazu dient, die Arbeit zu visualisieren, die laufende Arbeit zu begrenzen und die Effizienz zu maximieren. Unsere Tabelle besteht aus 6 Spalten:

- **Backlog:** Tickets geplant, aber nicht bearbeitet
- **Open:** Tickets, die im aktuellen Sprint erledigt werden müssen
- **On Hold:** Geplante, aber für unbestimmte Zeit blockiertes Ticket
- **In Progress:** Die Tickets, die realisiert werden
- **Review:** Tickets, die derzeit einer Code Review unterzogen werden
- **Closed:** Tickets, die validiert und im Entwicklungszweig zusammengeführt wurden

Jedes Ticket muss begleitet sein von:

- **Ein Story point:** es handelt sich um eine Zahl zwischen 1 und 8, die der geschätzten Anzahl der Stunden entspricht, die für die Erstellung des Tickets benötigt werden
- **eine Priorität:** dem Ticket muss Vorrang eingeräumt werden (low, medium oder high)

2.3.2 Tools

Wir verwalten all diese Prozesse sowie die Verwaltung des Codes mit verschiedenen Tools:

2 Grundlagen

2.3.2.1 Git

Git¹⁸ ist ein freies und quelloffenes verteiltes Versionskontrollsysteem. Es handelt sich um ein einfaches und leistungsfähiges Tool, dessen Hauptaufgabe darin besteht, die Entwicklung der Inhalte einer Baumstruktur zu verwalten.

Git ist das Tool, aber um ein kollaboratives Projekt richtig zu verwalten, braucht das Team einen Arbeitsablauf. Das Team hat sich für Gitflow¹⁹ entschieden. Dieses Workflow definiert ein striktes branch-modell, das um eine Projektfreigabe herum entworfen wurde²⁰.

Das innerhalb des Unternehmens verwendete Modell ist recht einfach. Es besteht aus einem "mainBranch", der die für die Code-Produktion fertige Version mit den verschiedenen Versionen enthält, den "developBranch", der dem Zweig entspricht, der sich auf alle "featureBranch" bezieht:

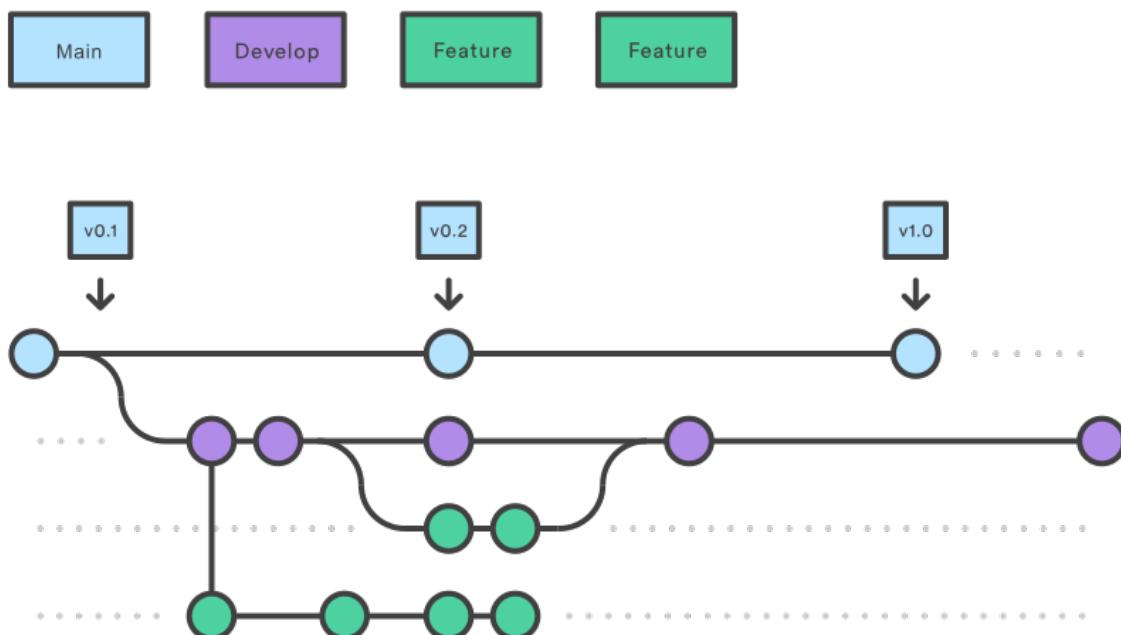


Abbildung 2.9: Schematische Darstellung eines Git-Baums, der dem Standard von gitflow entspricht

Das Cloud-Team nahm eine Änderung am Schema vor und entfernte den branch develop, der angesichts der geringen Größe des Teams nicht relevant war.

2.3.2.2 Slack

Die Kommunikation innerhalb der Firma erfolgt mithilfe der Slack application²¹. Damit können wir einander leicht anrufen und unsere Sofortnachrichten auf verschiedene Kanäle organisieren.

¹⁸<https://git-scm.com/>

¹⁹<https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow>

²⁰Stackoverflows git-flow tag Beschreibung: <https://stackoverflow.com/questions/tagged/git-flow>

²¹<https://slack.com/>

2.3.2.3 ClickUp

ClickUp²² ermöglicht es uns, unseren agilen Prozess zu virtualisieren: Tickets, Kanban-Tafel, Sprint-Planung, Roadmap, ...

2.3.2.4 Google Workspace

Die Dryad-Teams verwenden Google Workspace-Tools, um die externe Kommunikation über Google Mail, die Finanzen über Google Sheets, die Speicherung verschiedener Dateien und Bilder mit Google Drive und die Bearbeitung von Textdateien mit Google Doc zu verwalten.

2.3.2.5 GitHub

Die Verwendung von GitHub ermöglicht es, die Git-Projekte des Unternehmens online zu hosten. Es ermöglicht uns die einfache Verwaltung unseres Quellcodes in Zusammenarbeit mit verschiedenen Features wie Code-Reviews, Veröffentlichung von npm-Paketen und kontinuierlicher Integration.

2.3.3 Developer Experience

Da der Tech Stack definiert ist, ist es möglich, das Projekt zu starten und etwas aufzubauen. Es kann jedoch Bedenken geben, wenn mehrere Entwickler gleichzeitig am gleichen Quellcode arbeiten (Konflikte, Code-Formatierung, Sicherstellung der Gültigkeit des Codes, ...). Deshalb ist es wichtig, bestimmte Regeln und Tools einzuführen, die es dem Entwickler ermöglichen, sich auf die eigentliche Aufgabe zu konzentrieren. Für die Schaffung dieses Projekts war es wesentlich, die folgenden Tools/Prozesse zu implementieren, um sich in einer einfachen und homogenen Entwicklungsumgebung leicht weiterentwickeln zu können.

2.3.3.1 Linter

Ein Linter ist ein AnalyseTool für den statischen Code, das zur Kennzeichnung von Programmierfehlern, Bugs, stilistischen Fehlern und verdächtigen Konstrukten verwendet wird. Es wurde beschlossen, den eslint²³ Javascript-Linter wegen seiner Popularität und seiner verschiedenen, leicht konfigurierbaren Plugins zu verwenden (Angular²⁴, TypeScript²⁵, ...).

Eslint erlaubt es, den von jedem Entwickler geschriebenen Code zu homogenisieren. So ist die Einrückung für alle gleich, ebenso wie Zeilenumbrüche, Leerzeichen, variable Notation, usw...

2.3.3.2 TypeScript

TypeScript ist ein Tool, das entwickelt wurde, um Entwicklern die einfache Realisierung von JavaScript-Projekten zu erleichtern. Da es aber sehr anpassungsfähig ist, muss jedoch bekannt sein, was von diesem Tool erwartet wird und wie es verwendet werden soll.

²²<https://clickup.com/>

²³<https://github.com/eslint/eslint>

²⁴<https://github.com/angular-eslint/angular-eslint>

²⁵<https://github.com/typescript-eslint/typescript-eslint>

2 Grundlagen

```
> NX Ran target lint for project silvanet-web (7s)
  With additional flags:
    --fix=true

  ✘ 1/1 failed
  ✓ 0/1 succeeded [0 read from cache]

marker.ts
  45:7 error  'justAnUnusedVariable' is assigned a value but never used  @typescript-eslint/no-unused-vars
✖ 1 problem (1 error, 0 warnings)
```

Abbildung 2.10: Beispiel für die Linter-Ausführung, die einen Fehler im Terminal zurückgibt

Es wurde daher beschlossen, TypeScript mit einer hohen Anforderung zu konfigurieren. Daher muss alles typisiert werden (Variablen, Funktionen, Parameter), damit jeder Entwickler leichter verstehen kann, was an jeder Stelle im Code geschieht. Darüber hinaus ist der "beliebige" Typ verboten, da er die Verwendung von TypeScript überflüssig macht.

Dies mag dem DX-Prinzip zuwiderlaufen, da es anfangs viele Einschränkungen mit sich bringt, aber nach einer kurzen Anpassungszeit kommen die Vorteile eines stark typisierten Codes schnell zum Tragen und verbessern die Erfahrung des Entwicklers erheblich.

2.3.3.3 Drone CI

In jedem Beruf ist es sehr ärgerlich, immer wieder die gleichen Aufgaben auszuführen. In der Entwicklungswelt müssen wir zum Beispiel sicherstellen, dass der zur Produktion geschickte Code dem Linters-Standard entspricht, sich problemlos kompilieren lässt und vorher alle Tests besteht. Ständig darauf achten zu müssen, sie nicht zu vergessen, belastet die Entwickler unnötig mental.

Aus diesem Grund verwendet das Unternehmen ein kontinuierliches Integrations-Tool, Drone CI²⁶. Es ist sehr kompatibel mit GitHub und ermöglicht, bestimmte Aufgaben auszuführen, wenn Code auf GitHub gepusht wird.

Seine große Stärke ist die Einfachheit der Konfiguration. Es muss nämlich nach der Verknüpfung von travis mit dem GitHub-Projekt nur noch eine einfache `.drone.yml`-Datei an der Root des Projekts erstellt und bearbeitet werden.

2.4 Ergonomie und User Experience

Es gibt heute viele Möglichkeiten, eine Benutzeroberfläche zu präsentieren. Viele Schnittstellen vermitteln jedoch den Eindruck, dass sie schwer zu erforschen und zu nutzen sind, und es ist schwierig, das ursprüngliche Ziel zu erreichen. So mussten Konzepte erstellt werden, um einerseits eine Schnittstelle zu entwerfen und andererseits eine Schnittstelle, die von einem Menschen genutzt wird, so zu bewerten, dass sie effektiv, effizient und mit denen die Nutzer zufrieden sein werden. So entstanden zwei Designkonzepte, die seither allgemein verwendet werden: Ergonomie und User Experience (UX) von Schnittstellen.

2.4.1 Ergonomie

Ergonomie kann definiert werden als "Ausmaß, in dem ein Produkt von bestimmten Benutzern verwendet werden kann, um bestimmte Ziele mit Effektivität, Effizienz und Zufriedenheit in einem bestimmten Nutzungskontext zu erreichen"[7]. Da es sich um

²⁶<https://www.drone.io/>

eine recht umfassende und allgemeine Definition handelt, kann man sie in zwei Konzepte unterteilen.

Eine der Konzeptionen ist, dass sich die Ergonomie auf Maßnahmen konzentrieren sollte, die mit der Erfüllung der übergeordneten Ziele der Aufgabe zusammenhängen. Dies beinhaltet Techniken zur summativen Bewertung oder messbasierten Bewertung, sowohl objektive als auch subjektive Schnittstellen.

Die andere Konzeption ist, dass sich die Praktiker auf die Erkennung und Beseitigung von Problemen der Benutzerfreundlichkeit konzentrieren sollten.

2.4.2 User Experience

Während die Ergonomie also technisch und wissenschaftlich orientiert ist, um einen Benutzer bei der Ausführung einer Aufgabe zu bewerten, ist die UX emotionaler.

So sind instrumentelle Attribute wie Effizienz und Effektivität immer noch wichtig, aber hauptsächlich in dem Maße, in dem sie emotionale Ergebnisse wie Zufriedenheit, Vertrauen und wahrgenommene Schönheit beeinflussen, mit daraus resultierenden Auswirkungen auf das Ergebnisverhalten wie z. B. wiederholte Käufe und Weiterempfehlung an andere Nutzer.

UX nach Geoffrey Crofte [3] ist daher ganzheitlich und zeitlich orientiert, d. h. vor, während und nach der Nutzung der Schnittstelle. Daher konzentriert sich UX auf die Faktoren, die auf die Nutzung einer Schnittstelle folgen:

- das soziale Image der erwarteten Erfahrung
- die Erinnerung an vergangene Erlebnisse
- das soziale Teilen von Erfahrungen
- die Kontinuität des Service
- die Erinnerung an die durchgeführte Erfahrung

3 Analyse

3.1 Zielsetzung

Das Ziel dieser Thesis ist die methodische Analyse der Webschnittstelle zur Steuerung eines Silvanet-Systems, um dem Benutzer eine möglichst benutzerfreundliche, effiziente und zugängliche Erfahrung zu bieten. Der Nutzer sollte in der Lage sein, die Funktionsweise der Schnittstelle schnell und einfach zu erforschen und zu verstehen, abhängig von seinem Wissen über das System oder die Welt der Informatik im Allgemeinen und seiner Kultur.

Dazu kommt die Forschungs- und Entwicklungsarbeit zu interaktiven Karten, die in der Anwendung massiv eingesetzt werden, um Sensoren zu lokalisieren und schnell den Status eines Ortes zu erfahren. So kann der Nutzer schnell und in Echtzeit den Status eines bestimmten Standorts oder mehrerer Standorte je nach Zoomstufe der Karte verfolgen. Jeder Standort kann aus Tausenden von Sensoren bestehen, die über mehrere Hektar verteilt sind. Die Karte muss diese Punktdichte so verwalten, dass sie lesbar bleibt und dem Nutzer einfach die relevanten Daten liefert.

Schließlich wird eine Forschungsarbeit durchgeführt, um das Krisenmanagement zu verbessern. Dies betrifft insbesondere das Szenario eines Waldbrandes, bei dem der Nutzer die Behörden warnen und gleichzeitig eine Überpanik vermeiden muss. Die Schnittstelle muss daher in der Lage sein, den Benutzer schnell und effektiv über die Art der Gefahr zu informieren und ihn bei der Ausführung seiner Rolle zur Behebung der Gefahr zu leiten.

3.2 Ergonomische Inspektion

Eine der relevanten Aufgaben, die zu Beginn der Analyse einer Schnittstelle durchgeführt werden muss, ist eine ergonomische Inspektion der Schnittstelle. Diese Aufgabe erfordert nur das Wissen, das für diese Bewertung erforderlich ist, keine Drittparteien oder externen Benutzer.

Für diese Inspektion werde ich mich hauptsächlich auf die ergonomischen Kriterien stützen, die von J. M. C. Bastien und D. L. Scapin in ihrem Artikel *Ergonomic Criteria for the Evaluation of Human-Computer Interfaces*[1]. Diese Kriterien, die in acht Punkte unterteilt sind, wurden entwickelt, um die Dimensionen der Funktionalität einer Schnittstelle zu definieren und zu operationalisieren.

Sie können folgendermaßen zusammengefasst werden:

3 Analyse

Name	Beschreibung	Unterkriterien
Guidance	Mittel, die zur Verfügung stehen, um die Benutzer während ihrer Interaktion mit einem Computer zu beraten, zu orientieren, zu informieren, anzuweisen und zu leiten	Prompting, Grouping/Distinction of Items, Immediate Feedback, Legibility.
Workload	betrifft alle Schnittstellenelemente, die eine Rolle bei der Verringerung der wahrnehmungsbedingten oder kognitiven Belastung der Nutzer und bei der Steigerung der Dialogeffizienz spielen.	Brevity, Information Density
Explicit Control	Verarbeitung expliziter Benutzeraktionen durch das System und die Kontrolle, die die Benutzer über die Verarbeitung ihrer Aktionen durch das System ausüben.	Explicit User Action, User Control
Adaptability	Fähigkeit, sich kontextabhängig und entsprechend den Bedürfnissen und Vorlieben der Nutzer zu verhalten	Flexibility, User Experience
Error Management	Vorbeugung und Reduzierung von Fehlern und Behebung von Fehlern, wenn sie auftreten	Error Protection, Quality of Error Messages, Error Correction
Consistency	die Art und Weise, in der Entscheidungen für das Schnittstellendesign in ähnlichen Kontexten beibehalten werden und in unterschiedlichen Kontexten anders ausfallen	
Significance of Codes	Beziehung zwischen einem Begriff und/oder einem Zeichen und seiner Referenz.	
Compatibility	Übereinstimmung zwischen den Merkmalen der Benutzer und den Merkmalen der Aufgabe sowie die Organisation der Ausgaben, Eingaben und des Dialogs für eine bestimmte Anwendung,	

Tabelle 3.1: Zusammenfassung der ergonomischen Kriterien von Bastien und Scapin

Die Inspektion wurde über einen Zeitraum von zwei Wochen durchgeführt. Die Methodik bestand darin, für jedes Kriterium alle Seiten der Benutzeroberfläche durchzugehen und Mängel zu identifizieren. Dies führte zu einem Inspektionsbericht (pdf-Version in Englisch .2), in dem jedes Problem detailliert und mit einem Lösungsansatz beschrieben wurde. Angesichts der umfassenden Liste mussten die Aufgaben priorisiert werden. Dies geschah in Absprache mit dem Cloud-Team. Die kritischsten Punkte werden in den folgenden Abschnitten nach Kriterien gegliedert.

3.2.1 Guidance

Ein häufig anzutreffender Mangel in Schnittstellen ist das Fehlen von Labels in den Eingaben eines Formulars oder das Verwenden nur eines Icons. Dies passt zum Unterkriterium *Prompting* und stellt ein echtes Verständnisrisiko für den Nutzer dar. Obwohl viele Icons von der Mehrheit der Menschen als üblich angesehen werden, bleibt es ein Risiko, sie zu verwenden, vor allem, wenn die Schnittstelle für mehrere Kulturtypen oder mehrere

3.2 Ergonomische Inspektion

Ebenen der Computerkenntnisse auf Seiten der Benutzer bestimmt ist. P. Pappachan und M. Ziefle haben in ihrer Studie *Cultural Influences on the Comprehensibility of Icons in Mobile-Computer-Interaction*[9] nachgewiesen, dass Icons in allen Kulturen im Allgemeinen als das erkannt werden, was sie darstellen, aber nicht als das, was sie bedeuten, was jedoch für die Verständlichkeit von Icons von großer Bedeutung ist. So ist es notwendig, dass die Benutzeroberfläche Daten mit einer Erklärung präsentiert, die nicht nur ikonografisch ist, wie es auf dem folgenden Screenshot leider der Fall ist.

Last sending alert sensor		
	03.05.2022	
	10:07:40	
	27b2	
	52.85683	
	13.760759	

Abbildung 3.1: Darstellung mit Icons als einzigem Label

Doch es ist einfach, Rückfragen oder Fehlinterpretationen seitens des Nutzers zu vermeiden, indem man einfach eine Textbeschriftung neben dem Symbol hinzufügt.

Last Notification		
	Date	29/08/2022
	Time	16:34:40
	Sensor ID	2039
	Latitude	
	Longitude	
	Location	

Abbildung 3.2: Hinzufügen von Textlabels zusätzlich zu den Icons, um die Verwechslung von 3.1 einzuschränken

Ein weiteres Unterkriterium betrifft die ortsbezogene Gruppierung der Komponenten einer Schnittstelle nach ihrer Zugehörigkeit und Hierarchie. In der Tat wird der Benutzer *die verschiedenen Elemente leichter erkennen, wenn sie in einer geordneten Weise präsentiert werden*[1]. Dies hat den Vorteil, dass die Items einer Schnittstelle schneller erlernt und leichter behalten werden können. Die Silvanet-Webschnittstelle ist voll von Stellen, an denen die Hierarchie der Komponenten betont werden könnte. Ein gutes Beispiel ist das folgende Filtermenü:

3 Analyse

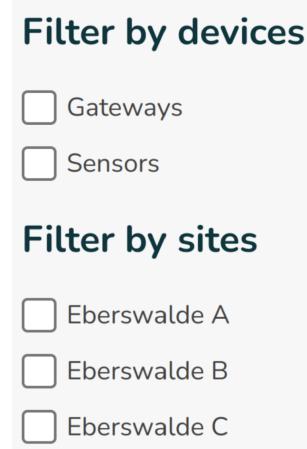


Abbildung 3.3: Menü mit geringer hierarchischer Unterscheidung

Doch schon das Hinzufügen von etwas mehr Padding auf der Ebene der Items hilft dem menschlichen Auge, eine Hierarchie zu visualisieren, in der die Items zu dem Titel über ihnen gehören. Darüber hinaus kann ein Kollaps-System verwendet werden, um die Zugehörigkeit zu einer Kategorie weiter zu spezifizieren. Dieses System kann einfach durch ein Chevron-Symbol verdeutlicht werden.

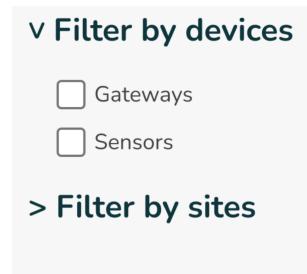


Abbildung 3.4: Neustrukturierung des Menüs 3.3, um die Sichtbarkeit der Hierarchie zu erhöhen

Ähnlich wie das Unterkriterium Unterscheidung nach Ort gibt es auch das Unterkriterium Unterscheidung nach Formatierung. Ein gutes Beispiel dafür, wie dieses Kriterium angewendet werden könnte, ist die Anzeige von Namen und Kennungen von Einheiten wie Sensoren oder Gateways. In der folgenden Abbildung sehen wir eine Liste von Geräten, die mit einem Namen identifiziert werden, der eine Art Identifikator enthält. Da die Kennung eines Geräts nur als Text angezeigt wird, ist die Identifizierung langsamer. In dieser Form ahnt der Nutzer nicht, dass es möglich ist, mit diesem Text zu interagieren, indem er ihn anklickt. Dies ist jedoch der Fall, da es möglich ist, auf die Detailseite eines Geräts zu navigieren, indem man auf diese Art von Kennung klickt.

Device	Site	Device type
Eberswalde BG xBvvX	Eberswalde A	Gateway
Eberswalde SN ofUAI	Eberswalde A	SensorNode

Abbildung 3.5: Anzeige von Geräteentitäten ohne Differenzierung der Formatierung

Das Hinzufügen eines einfachen/reinen Designs eines Tags oder einer Schaltfläche (Hintergrund mit schlichter Farbe und einem abgerundeten, ausgeprägten Rand) zu den

3.2 Ergonomische Inspektion

Gerätekennungen je nach Gerätetyp hilft den Nutzern, eine dichte Benutzeroberfläche schneller zu lesen.

Device	Site	Device type
Eberswalde BG xBvvX	Eberswalde A	Gateway
Eberswalde SN ofUAI	Eberswalde A	SensorNode

Abbildung 3.6: Anzeige von Geräteentitäten mit Differenzierung der Formatierung

Dies erleichtert dem Benutzer das Erlernen der Fähigkeit, diese Art von Entitäten über die gesamte Schnittstelle zu erkennen.

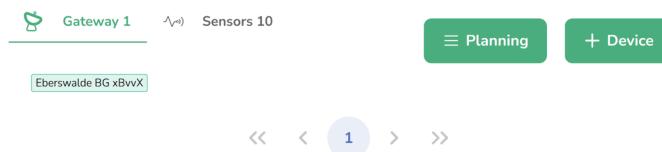


Abbildung 3.7: Gemeinsame Nutzung der Formatierung der Entität Gerät

Eine der Säulen der Benutzerführung zur Aufgabenerledigung ist das *sofortige Feedback* nach einer Aktion. In allen Fällen sollte eine schnelle Antwort des Schnittstelle mit Informationen über die angeforderte Transaktion und ihr Ergebnis erfolgen. Leider verarbeitet die Silvanet-Anwendung nur sehr wenig Feedback zu den Aktionen des Benutzers. Aufgrund der geringen Effektivität und des Zeitmangels bei der Entwicklung der Schnittstelle wird dem Nutzer nur ein kleiner Teil der Erfolge gezeigt. Bei den meisten Knöpfen, die angeklickt werden, erhält der Nutzer z. B. kein Feedback. In den meisten Fällen wird die Aktion relativ schnell ausgeführt und der Nutzer schließt daraus auf einen Erfolg. Es kann jedoch vorkommen, dass die Aktion einige Zeit in Anspruch nimmt (z. B. bei einer schlechten Verbindung), und der Nutzer sollte darüber informiert werden, dass die Anfrage bearbeitet wird.

Es ist daher von entscheidender Bedeutung, dass eine angeklickte Schaltfläche, die eine Aktion ausführt, dem Benutzer eine Schnittstelle präsentiert, die den Ladevorgang veranschaulicht. Dies kann leicht durch die Verwendung einer UI-Komponente namens Spinner erreicht werden.



Abbildung 3.8: Beispiel einer Schaltfläche, die nach dem Anklicken einen Spinner darstellt, der eine laufende Aktion anzeigen

Ein ähnliches Problem ist bei der Anzeige von Sensoren auf der interaktiven Karte zu erkennen. Tatsächlich werden die verschiedenen Geräte an einem Standort im Hintergrund abgerufen, sobald der Nutzer auf eine bestimmte Ebene zoomt. Wenn die Internetverbindung jedoch nicht schnell genug ist oder der Benutzer schnell zoomt, hat der Benutzer nicht das Gefühl, dass im Hintergrund eine Abfrage stattfindet, da nichts passiert, und sucht daher vergeblich nach Sensoren auf der Karte.

3 Analyse

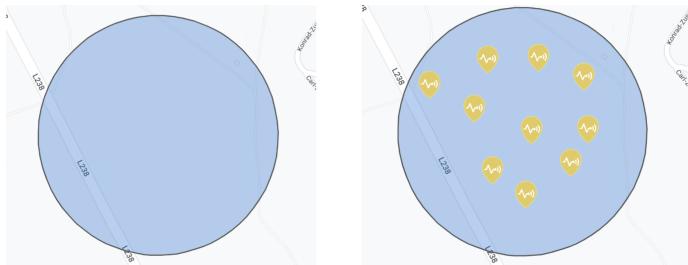


Abbildung 3.9: Bei einer langsamen WLAN-Verbindung zoomt der Benutzer in eine Site (blauer Bereich), aber es scheint nichts zu passieren. Erst nach einigen Sekunden sind die Sensoren sichtbar.

Dasselbe Problem findet sich in der überwiegenden Mehrheit der Schnittstelle, wo die meisten Abfragen (fetch/XMLHttpRequest (XHR)) dem Benutzer nicht angezeigt werden. Sehr oft kann man mit einer langsamen Verbindung auf einen Bereich der Schnittstelle zugreifen, der leer ist, weil die Daten gerade geladen werden. Es gibt jedoch viele UI-Komponenten, die einem Benutzer mitteilen, dass eine Aktion im Hintergrund abläuft. Eine der am besten geeigneten Komponenten für diese Art von Feedback ist das Skeleton, das einerseits den Benutzer über das Laden einer Seite oder eines Teils einer Seite benachrichtigt, andererseits aber auch die wahrgenommene Wartezeit des Benutzers verkürzt. Die Studie von Bill Chung in seinem *Artikel Everything you need to know about skeleton screens*[2] zeigt, dass Skeleton-Komponenten am besten gegen einen einfachen Spinner oder eine leere Seite abschneiden, was die Reduzierung der Wartezeitwahrnehmung betrifft.

Das in der Silvanet-Schnittstelle verwendete Library, PrimeNg, bietet eine Skeleton-Komponente out of the box und wird eine einfache Implementierungslösung für dieses Issue ermöglichen.



Abbildung 3.10: Beispiel für UI-Komponenten-Skeletons, die vom Library PrimeNg bereitgestellt werden

3.2.2 Workload

Das Unterkriterium *Brevity / Concision* besagt, dass eine Schnittstelle den wahrnehmungsbezogenen und kognitiven Aufwand für einzelne Eingaben oder Ausgaben begrenzen sollte, da die kurzfristige Erinnerungsfähigkeit eines Nutzers begrenzt ist. Leider erfüllt die Silvanet-Anwendung dieses Kriterium nicht, wenn es darum geht, dem Benutzer metrische Daten anzuzeigen. Die Anwendung erfordert, dass Flächen mehrfach angezeigt werden, sei es die Fläche einer Site oder die sensorische Abdeckung eines Sensors. Wenn jedoch keine Skalierung der Daten vorgenommen wird, wird es für einen Nutzer sehr schnell schwierig, diese Daten zu bewerten, wie man im nächsten Screenshot sehen kann:

Covered area of each site

Eberswalde A 77012 m²

Eberswalde B 216557 m²

Eberswalde C 1963 m²

Abbildung 3.11: Messung von nicht skalierten Bereichen, die die kognitive Belastung der Schnittstelle verstärken

Das Unterkriterium *Brevity / Minimal Actions* weist auf ein allgemeines Problem der Silvanet-Anwendung hin, nämlich den Mangel an eindeutigen URL-Zugriffen auf einen Teil der Anwendung. Die Anwendung erfolgt hauptsächlich durch das Öffnen und Schließen von Popups mittels Knopfdruck. In vielen Szenarien muss der Nutzer daher mehrere Klicks ausführen, um zum Stadion zurückzukehren, z. B. vor einer Seitenaktualisierung. Mit diesem Mangel lässt die Anwendung dem Nutzer nicht viel Freiheit, einfach einen bestimmten Teil der Anwendung zu teilen.

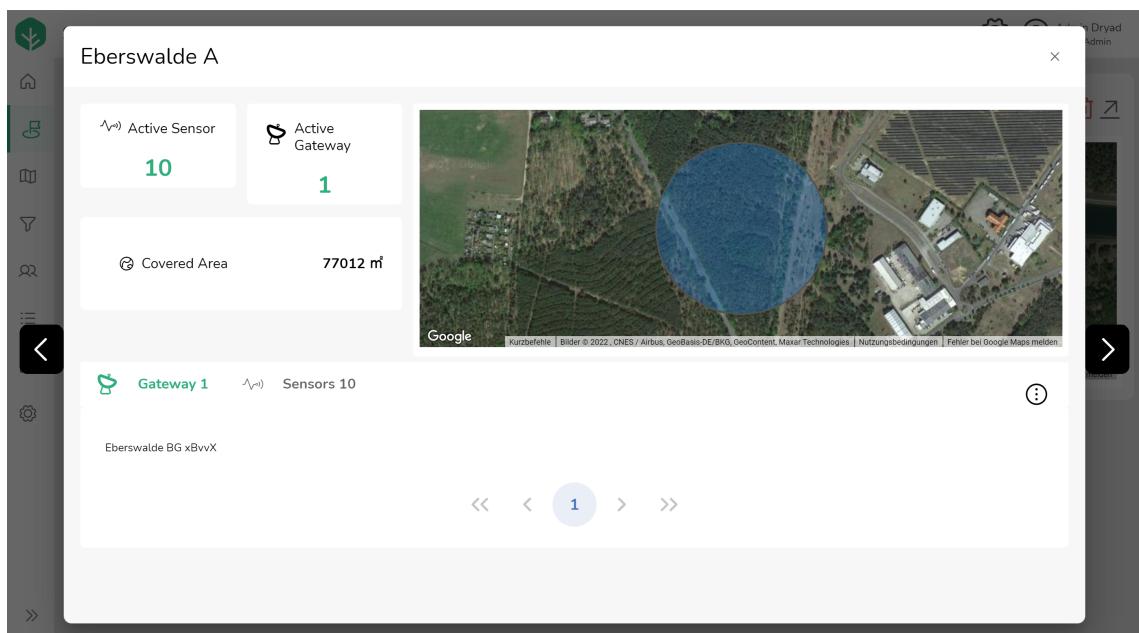


Abbildung 3.12: Popup, das die Details einer *Site* präsentiert

Eines der markantesten Beispiele ist die Seite mit den Details einer *Site*, die in einem Popup geöffnet wird. Ein Nutzer kann diese Details nicht direkt mit einem anderen Nutzer teilen, da es keinen Link zum Öffnen des Popups gibt.

3.2.3 Explicit Control

Einer der wichtigsten Aspekte der Silvanet-Anwendung, wenn es um das Kriterium *Explicit Control* geht, sind die Aktionen in Popups. Wenn ein Popup vom Benutzer geschlossen wird und sich darin ein Formular befindet, wird das Formular so gesendet, als ob es vom Benutzer bestätigt worden wäre. Dies ist in den Augen eines Benutzers sehr verwirrend,

3 Analyse

denn eine Schnittstelle sollte "immer einen Benutzer auffordern, eine explizite ENTER-Aktion durchzuführen, um die Verarbeitung eingegebener Daten einzuleiten"[1].

Das Kriterium *User Control* bezieht sich auf die Tatsache, dass die Benutzer immer die Kontrolle über die Systemverarbeitung haben sollten (z. B. Unterbrechen, Abbrechen, Anhalten und Fortsetzen). Jede mögliche Aktion eines Benutzers sollte antizipiert werden und es sollten entsprechende Optionen zur Verfügung gestellt werden. Die silvanet-Anwendung hat an vielen Stellen riskante Aktionen, wie z. B. das Löschen einer *Site* oder das Festlegen, dass eine Waldbrandwarnung ein Fehlalarm ist. Obwohl diese Aktionen bereits durch einen Bestätigungsdialog geschützt sind, könnte es sinnvoll sein, die Fehlerbehandlung zu verstärken, indem man eine *Vorgang abbrechen*-Aktion hinzufügt. Dies wird dem Benutzer ein Gefühl der Zuversicht vermitteln, wenn er die Schnittstelle manipuliert, da keine Aktion endgültig ist. Ein gutes Beispiel für diese Art der Rückgabeaktion ist auf der Gmail-Oberfläche mit dieser Komponente namens *Toast* zu sehen, die es dem Nutzer ermöglicht, das Löschen einer E-Mail rückgängig zu machen:

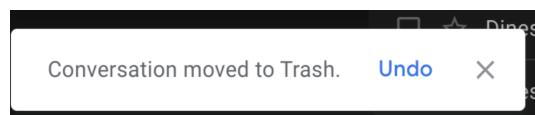


Abbildung 3.13: *Toast*nachricht von Gmail, mit der das Löschen einer E-Mail rückgängig gemacht werden kann

3.2.4 Adaptability

Nach den Kriterien von Bastien und Scapin muss eine Schnittstelle, um ergonomisch zu sein, eine Anpassungsfähigkeit an den Benutzer bieten. Dies wird im Kriterium *Flexibility* hervorgehoben, das sich in der Fähigkeit einer Schnittstelle ausdrücken lässt, sich an die Bedürfnisse eines Nutzers anzupassen. So wäre es relevant, dass die Schnittstelle von Silvanet ein gewisses Maß an Anpassungsmöglichkeiten bietet. Ein guter Kandidat wäre die Seite, auf der die verschiedenen *Sites* eines Nutzers vorgestellt werden. Derzeit besteht diese Liste aus Card-Komponenten, die den Namen des *Sites* und eine Ansicht des *Sites* auf einer interaktiven Karte anzeigen.

A screenshot of a web application titled 'Site Management'. On the left, there is a vertical sidebar with icons for home, site management, and other navigation items. The main area shows two cards side-by-side. Each card has a title ('Eberswalde A' and 'Eberswalde B'), a map view with a large circular overlay, and a set of four small icons for settings, user, trash, and edit. At the bottom of each card, there is a 'Google' logo and a row of small text links: 'Kurzbefehle', 'Kartendaten', 'Nutzungsbedingungen', and 'Fehler bei Google Maps melden'. The overall interface is clean and modern, using a light gray background and a mix of blue and green colors.

Abbildung 3.14: Seite mit der Liste der *Sites* eines Nutzers

Auf dieser Seite wäre es möglich, das Auffinden einer bestimmten *Site* in der Liste zu erleichtern, indem man dem Benutzer die Möglichkeit gibt, Änderungen an der UI vorzunehmen, um so die Erinnerung an die Schnittstelle zu fördern. Eine interessante Idee ist

3.2 Ergonomische Inspektion

es, den Nutzer die Liste nach seinen Vorlieben ordnen zu lassen. Um das schnelle Erkennen eines Listeneintrags zu erleichtern, könnte es möglich sein, die Hintergrundfarbe der Card-Komponente zu ändern. Im gleichen Sinne könnte der Nutzer die interaktive Karte durch ein Bild seiner Wahl ersetzen.

Ein interessanter Aspekt der Silvanet-Anwendung und ihre Absicht, weltweit nutzbar zu sein. Dryads Lösung zur Erkennung von Waldbränden sollte allen potenziellen Kunden auf der ganzen Welt angeboten werden. So muss die Anwendung unabhängig von der Sprache oder der Kultur des Nutzers nutzbar sein. Aus ergonomischer Sicht ist dies mit einer Vielzahl von Abhängigkeiten verbunden, darunter auch die der Internationalization (I18n) und der Localization (L10n). Die Einführung dieser beiden technischen Verfahren wird als Globalisierung bezeichnet[13].

3.2.4.1 I18n

Die Internationalisierung umfasst die Planungs- und Vorbereitungsphasen für ein Produkt, in denen es von vornherein für die Unterstützung globaler Märkte konzipiert wird. Dieser Prozess bedeutet, dass alle kulturellen Annahmen entfernt werden und alle länder- oder sprachspezifischen Inhalte außerhalb des Produkts gespeichert werden, so dass es leicht angepasst werden kann. Die Internationalisierung besteht in erster Linie darin, die Funktionalität eines Produkts von einer bestimmten Kultur, Sprache oder einem bestimmten Markt zu abstrahieren, damit die Unterstützung für bestimmte Märkte und Sprachen problemlos integriert werden kann. In der Welt der Software gibt es viele Beispiele für Designfehler bei der Internationalisierung, wie z. B. nicht übersetzte Texte in Grafiken, nicht übersetzte Screenshots der Anwendung in der Dokumentation oder Telefonnummern, die im Ausland nicht verwendet werden können.

3.2.4.2 L10n

Die Lokalisierung bezieht sich auf die tatsächliche Anpassung des Produkts an einen bestimmten Markt. Sie umfasst die Übersetzung, die Anpassung von Grafiken, die Übernahme lokaler Währungen, die Verwendung korrekter Formulare für Daten, Adressen und Telefonnummern und viele andere Details, einschließlich der physischen Struktur von Produkten in einigen Fällen. Wenn diese Details in der Internationalisierungsphase nicht vorgesehen waren, müssen sie während der Lokalisierung korrigiert werden, was das Projekt zeitlich und finanziell belastet.

Die Globalisierung kann also in einen Kreislauf zwischen I18n und L10n übersetzt werden, den das Team von Dryad bei der Erstellung ihrer Produkte anwenden könnte.

3 Analyse



Abbildung 3.15: Darstellung des Globalisierungszyklus eines Produkts

3.2.5 Error Management

Wie in den Empfehlungen zum *Error Management* erwähnt, ist es besser, "Fehler vor der Validierung als danach zu erkennen". Daher sollte die Schnittstelle erkennen und verhindern Dateneingabefehler, Befehlsfehler oder Aktionen mit zerstörerischen Folgen. Dies gilt insbesondere für die verschiedenen Formulare von Silvanet, wo die meisten Inputs nicht angeben, welches Format erlaubt ist, und nach dem Absenden des Formulars einen Fehler zurückgeben. Es soll die Benutzererfahrung verbessern, wenn Sie wissen, dass Sie in dieser Eingabe nur Zahlen eingeben dürfen, dass Leerzeichen verboten sind oder dass Kommazahlen im Format xx.xxx geschrieben werden.

Es wird daher empfohlen, dass die Fehlermeldungen für den Benutzer klar sind, wenn eine Aktion nicht wie erwartet verläuft. Die Anwendung, die sich in einem unfertigen Stadium befindet, zeigt immer wieder Fehlermeldungen an, die für den Benutzer irrelevant sind, wie *Error* oder *Ein Fehler ist aufgetreten*.



Abbildung 3.16: Screenshot von Push-Benachrichtigungen, die nach einer Fehlbedienung an den Benutzer gesendet werden

Es ist jedoch notwendig, eine ausgezeichnete Qualität der Fehlermeldungen anzubieten, die das Lernen der Nutzer aus den Systemen fördert, indem sie ihnen die Gründe für ihre Fehler und deren Art mitteilen und ihnen beibringen, wie sie ihre Fehler verhindern oder beheben können.

3.2.6 Consistency

Die Kriterien besagen, dass die UI-Elemente einer Benutzeroberfläche von Bildschirm zu Bildschirm und von Sitzung zu Sitzung stabil und konsistent sein müssen. Unter diesen Bedingungen ist das Computersystem berechenbarer, das Lernen wird erleichtert und die Anzahl der Fehler wird reduziert. Ein auffälliges Beispiel für eine solche Inkohärenz findet sich jedoch in einem entscheidenden Element der Anwendung: dem Warnknopf, der bei der Entdeckung eines Waldbrandes angezeigt wird. Dieser zeigt dem Benutzer den Text "Fire alert", dem das rote Icon eines Sensors vorangestellt ist. Im gesamten Rest der Anwendung wird ein Feuer jedoch mit dem Icon einer Flamme dargestellt.

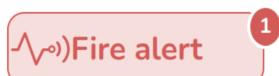


Abbildung 3.17: Icon eines Sensors, der auf der Feueralarmtaste verwendet wird



Abbildung 3.18: Icon, das in einer Stecknadel verwendet wird, um den Standort eines Feuers auf der Karte anzuzeigen

Damit ein Benutzer einfach einen Zusammenhang zwischen einem Waldbrand und einem Icon herstellen kann, um die Benutzeroberfläche zu entlasten, muss das verwendete Icon für jede Situation einzigartig sein.

Im gleichen Sinne hat die Schnittstelle Schwierigkeiten, die Konsistenz bei der Anzeige der Anzahl von Entitäten zu wahren. Tatsächlich ist das Design des Betrags nicht immer gleich, was dem Nutzer nicht hilft, ihn leicht als Betragswert zu identifizieren. Ein Beispiel ist die Differenz zwischen dem Betrag für die Geräte und dem Betrag für den Waldbrandalarm. Eine wird als Abzeichen dargestellt und die andere zeigt einfach den Betrag neben der Entität, die durch einen vertikalen Balken getrennt ist.

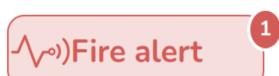


Abbildung 3.19: Darstellung eines Betrags als Badge

Sensor | 2104 Gateway | 232

Abbildung 3.20: Darstellung eines Betrags als einfache Zahl neben der betreffenden Entität

In diesem Fall sollte die *UI* vereinheitlicht werden, indem eine Entscheidung darüber getroffen wird, welche Anzeige bei der Präsentation eines Betrags verwendet werden soll. Dies wird es dem Benutzer ermöglichen, auf einer komplexen Oberfläche die Beträge einer Entität leicht von anderen Zahlen zu unterscheiden.

3.2.7 Schlussfolgerungen

Die Analyse der verschiedenen Seiten der Silvanet-Schnittstelle mithilfe der ergonomischen Kriterien von Bastien und Scapin hat bereits viele Mängel aufgedeckt, die die Leistung eines Benutzers beim Verstehen, Lernen und Beherrschen des vorgeschlagenen Systems beeinträchtigen können. Diese Probleme wurden mithilfe von Tickets im ClickUp-Tool¹ in verschiedene Aufgaben aufgeteilt, die unter den verschiedenen Entwicklern aufgeteilt, nach Relevanz und Schwierigkeit der Implementierung priorisiert und schließlich im Scrum-Flow² verwendet wurden. Ein Teil der Aufgaben, die sich aus dieser Forschungsarbeit ergeben und die mir zugewiesen wurden, werden im Teil Konzeption³ der Thesis vorgestellt.

3.3 Benutzertests

Wie im Dokument "Ergonomic Criteria for the Evaluation of Human-Computer Interfaces" spezifiziert, stellt eine kriterienbasierte Bewertungsmethode lediglich einen analytischen Ansatz dar. Als solche sind die Kriterien nicht dazu gedacht, andere Bewertungsmethoden zu ersetzen. Daher sind Benutzertests weiterhin notwendig, um die Leistung und Nutzbarkeit einer Schnittstelle zu gewährleisten. Aus diesem Grund wurden Benutzertests entwickelt, um allgemeine Usability-Probleme zu identifizieren und potenziell Probleme zu erkennen, die nicht von den Kriterien abgedeckt werden.

Das Ziel dieses Tests ist es, die Effektivität, Effizienz und Zufriedenheit mit der Silvanet-Webanwendung mithilfe von echten Nutzern zu quantifizieren. Zunächst muss jedoch definiert werden, was getestet werden soll und vor allem, wie getestet werden soll.

3.3.1 Festlegung der Testabdeckung

Um relevante und schnelle Tests für die Benutzer durchführen zu können, mussten wir festlegen, auf welche Bereiche der Benutzeroberfläche sich die Tests konzentrieren sollten. Es erschien uns sinnvoll, eine Gruppenaktivität mit dem Cloud-Team durchzuführen, um Ideen zu sammeln und die am meisten erwähnten auszuwählen. Um die Brainstorming-Sitzung zu strukturieren und zu vermeiden, dass man sich zu sehr verzettelt, basiert die Aktivität auf der Ideationstechnik namens *Lotus Blossoms*, die es ermöglicht, eine Idee nach der anderen gründlich zu erforschen. Diese von S. Tatsuno 1990[12] entwickelte Methode der Ideenfindung lässt sich leicht durch das folgende Flussdiagramm schematisieren.

¹2.3.2.3

²2.3.1

³4

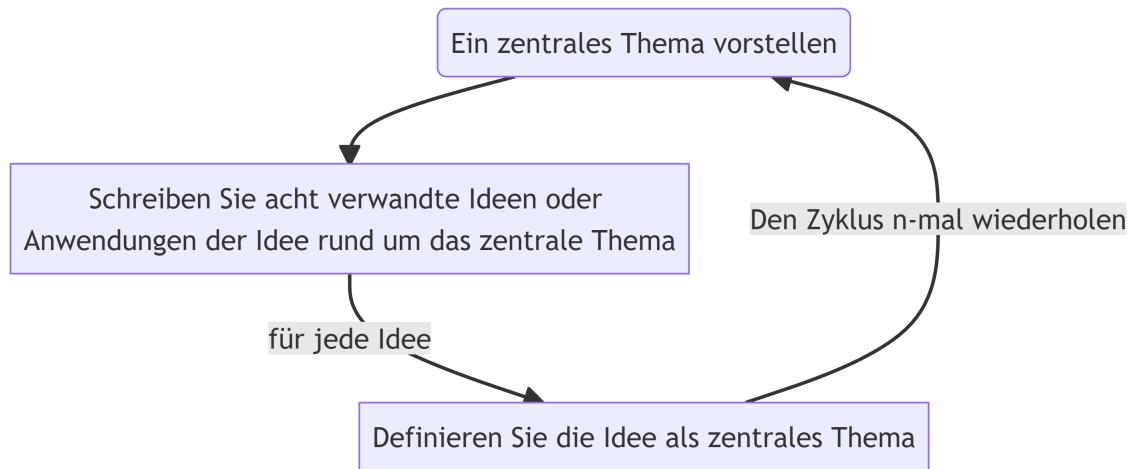


Abbildung 3.21: Darstellung eines Ideationszyklus mit der Lotus-Blossom-Methode

In diesem Fall sollte eine Iteration von zwei Zyklen ausreichen, um die wesentlichen Funktionen der Anwendung zu definieren. Die erste zentrale Frage wurde im Vorfeld mit der Frage "Was sind die Ziele des Gesprächs mit dem Kunden über die Anwendung?" festgelegt. Aus dieser Frage ergaben sich 6 verschiedene andere Themen:

- Welche Informationen über die Geräte sind von Interesse?
- Ist die Planung eines *Site* funktionsfähig und die Schnittstelle funktionstüchtig?
- Welche kundenorientierten Funktionen sollten den Nutzern zur Verfügung gestellt werden?
- Welche Art von Informationen würden Sie im Falle eines Feueralarms interessieren?
- Welche Art von Informationen möchten Sie auf dem Dashboard sehen?
- Funktioniert die Kartendarstellung gut?

3 Analyse

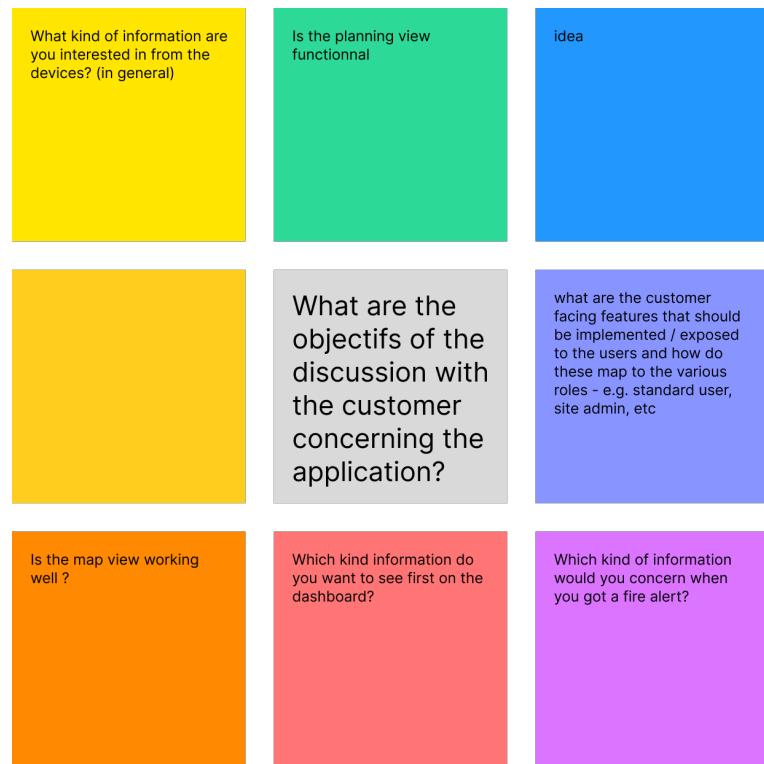


Abbildung 3.22: Durchführung des ersten Ideationszyklus durch das Cloud-Team von Dryad mit der Lotus-Blossom-Methode

In der zweiten Iteration wurden mehr Fragen zu den Funktionen der Benutzeroberfläche gestellt, die die genannten Themen beantworten oder näher erläutern können.

3.3 Benutzertests

How do we present device management features (internal firmware update, ML, model updates, inventory management) ?	How relevant is it to have data from a single device compared to a group of devices?	What kind of data representation do you naturally understand easily? Is it more individual or grouped based?	Is the user flow of the planning view intuitive ?	Do you achieve planning an efficient sylvanet network?	Can you easily split the site deployment areas between different members?	Idea	Idea	Idea
How do we connect the device information to meaningful actions a user could perform ?	What kind of information are you interested in from the device(s) (in general)?	How in which format user want to see the provided data?	Which problems user face while creating the planning task (flow, interface, UI)?	Is the planning view functional	should we create a separate page for the planning view with more specific features added?	Idea	Idea	Idea
How do we present information related to the health of the devices ?	How do we present the history data related to devices ? (ie: battery levels, message sent,etc)	Do user want to see every thing in detail (with the number) or the data can be presented in a more summarized visually?		Without the planning feature, how would you manage the sensors deployment?	Are you able to manage the sensors stock?	Idea	Idea	Idea
Idea	Idea	Idea	What kind of information are you interested in from the device(s) (in general)?	Is the planning view functional	Idea	what is the level of technical detail to expose to the users	what are the admin features (permissions) to expose to users based on the various roles	How does the user understand the status of the entire system
Idea	Idea	Idea	What are the objectives of the discussion with the customer concerning the application?			Idea	what are the customer facing features that should be implemented / exposed to the users and if so these map to the various roles - e.g. standard user, site admin, etc	How much users will have admin privilege
Idea	Idea	Idea	Is the map view working well?	which kind of information do you want to see first on the dashboard?	What kind of information would you concern when you get a fire alert?	Idea	Is our User-Flow of all screens (special Alert Page and Planning Page) now clear with the user?	Idea
Can we make the usage of the map view more intuitive?	Should display the sensors in default zoom level?	Which format view of map should be default (graphic, statewide?)	How do we present the user with minimal information to start interacting and allow user to click-over to discover more information?	How do we present the status of global health of the system?	Should it be customizable? Should we allow user to select the widgets they need?	Idea	How would you like to be notified? Email, message, call, push alert, custom alarm hardware	Do you want to handle the emergency call? If multiple people get the notification what one is supposed to make the call?
Can we make the user input interactions smoother and streamlined?	Is the map view working well?	Do you easily find the location of a specific sensor (value within the map view (sensor offline for example))	How do we use the dashboard as a basic view with connections to all other views?	Which kind of information do you want to see first on the dashboard?	Should we make the dashboard more interactive? And Should we group the global data in one place and then separate for easy-view?	Idea	Which kind of information would you concern when you get a fire alert?	How we arrange the information to notify effectively? (ex: what should be displayed first)
How do we implement zoom level based interactions?	Do we need clustering for markers?	Is the clicking behaviour for each function clear with user?	Should/How do we add Call To Actions to relevant information on the dashboard?	Do you find every information/data you are looking for if not what's missing?	How do we design an easy workflow for the user to resolve fire alerts?	Idea		

Abbildung 3.23: Durchführung des zweiten Ideationszyklus durch das Cloud-Team von Dryad mit der Lotus-Blossom-Methode

Die Einzelheiten zu den verschiedenen Blütenblättern der Lotus Blossom des zweiten Zyklus können im Anhang .1 eingesehen werden.

Diese Aktivität ermöglichte es, die Usability-Tests schließlich auf die folgenden Features zu konzentrieren:

- **Planung der Sensoren eines Site UX:** Die Anwendung bietet dem Nutzer eine Seite, auf der er auf einer interaktiven Karte die neuen Sensoren an einem Site so positionieren kann, dass die Erfassungsabdeckung der Sensoren optimiert wird. Dies führt den Nutzer dann durch den Wald zu den Koordinaten, an denen er einen Sensor hoch oben an einem Baum befestigen muss.
- **Relevanz der Information in Alarmsituationen:** Wenn ein Feuer vom System erkannt wird, hat der Benutzer Zugang zu einer Alarmseite, die die Situation mithilfe von Daten und einer interaktiven Karte detailliert darstellt. Es sind auch schnelle Aktionen möglich, wie z.B. die Kontaktaufnahme mit den örtlichen Behörden. Dies

3 Analyse

betrifft auch Informationen, die per E-Mail an den Nutzer gesendet werden, um ihn über den Alarm zu benachrichtigen.

- **Usability der interaktiven Karte:** Die Anwendung bietet eine Seite, die nur aus einer interaktiven Karte besteht, auf der Sie die verschiedenen *Sites* und Sensoren anhand ihrer geografischen Lage einsehen können.
- **Relevanz von Sensordaten:** In der Anwendung werden die Sensordaten auf unterschiedliche Weise dargestellt. Es wäre interessant, die Strategie zu definieren, die verwendet wird, um diese Daten für den Benutzer relevant zu präsentieren.

3.3.2 Methoden für Usability-Tests

3.4 Benutzerumfrage über die Feuerwehr-Community

4 Konzeption

4.1 Verwenden eines Dashboards für schnelle Entscheidungsfindung

4.2 Verhalten einer interaktiven Karte beim Zoomen und Auszoomen

4.3 Präsentation von Daten in Stresszeiten

5 Implementierung

5.1 Entwicklung eines Benutzers Fragebogens

5.2 Geschichtete Darstellung auf interaktiver Karte

6 Evaluation

7 Diskussion

8 Zusammenfassung und Ausblick

8.1 Zusammenfassung

8.2 Ausblick

Literatur

- [1] J. M. C. Bastien und D. L. Scapin. „Ergonomic Criteria for the Evaluation of Human-Computer Interfaces“. In: (1993).
- [2] Bill Chung. „Everything you need to know about skeleton screens“. In: *UX Collective* (Aug. 2018).
- [3] Geoffrey Crofte. „Ergonomie des interfaces et Expérience Utilisateur“. In: (2017).
- [4] Will Fanguy. *A comprehensive guide to design systems*. Techn. Ber. Inside Design, 2019.
- [5] San-Miguel-Ayanz J, Durrant T, Boca R, Maianti P, Liberta' G, Artes Vivancos T, Jacome Felix Oom D, Branco A, De Rigo D, Ferrari D, Pfeiffer H, Grecchi R und Nuijten D. „Advance report on wildfires in Europe, Middle East and North Africa 2021“. In: KJ-NA-31028-EN-N (online) (2022). ISSN: 1831-9424 (online). DOI: 10.2760/039729 (online).
- [6] Ralph E. Johnson und Brian Foote. „Designing Reusable Classes“. In: *Journal of Object-Oriented Programming* (1988).
- [7] J. R. Lewis und J. Sauro. „Usability and User Experience: Design and Evaluation“. In: (2021).
- [8] Christopher Nance. „TypeScript Essentials“. In: *Packt Publishing Ltd* (2014), S. 27–31.
- [9] P. Pappachan und M. Ziefle. „Cultural influences on the comprehensibility of icons in mobile-computer interaction“. In: *Behaviour Information Technology* 27 (2008).
- [10] Riegsecker und Austin. „Measuring Environmental Effects on LoRa Radios in Cold Weather Using 915 MHz“. Diss. Dez. 2017.
- [11] Ken Schwaber und Jeff Sutherland. „The Scrum Guide“. In: (2010).
- [12] S. Tatsuno. „Created in Japan: from imitators to world-class innovators.“ In: (1990).
- [13] *What is Globalization?* Techn. Ber. LISA: Localization Industry Standards Association, 2011.

Abbildungsverzeichnis

2.1	Schematisierung einer Star-of-Stars-Netzwerk-Architektur, bestehend aus <i>Nodes</i> und <i>Gateways</i> , die vom LoRaWAN-Standard verwendet wird[10]	3
2.2	Schematische Darstellung des LoRaWAN-Netzwerks, das die verschiedenen Sensoren mit dem Cloud-System verbindet.	4
2.3	ein luftschnüffelnder Sensor, der ein Silvanet-Netzwerk bildet	4
2.4	https://www.tothenew.com/blog/optimization-of-angularjs-single-page-applications-for-web-crawlers/	5
2.5	Liste der beliebtesten Programmier-, Skript- und Auszeichnungssprachen im Jahr 2022	6
2.6	Stackoverflow Survey 2022: Beliebteste Web-Frameworks	7
2.7	Rendern auf die Schnittstelle einer <i>p-paginator</i> -Komponente, wie in Abbildung 2.3 deklariert.	9
2.8	Überblick über einige UI-Komponenten des Designsystems von Dryad entwickelt auf der Designanwendung Figma	10
2.9	Schematische Darstellung eines Git-Baums, der dem Standard von gitflow entspricht	12
2.10	Beispiel für die Linter-Ausführung, die einen Fehler im Terminal zurückgibt	14
3.1	Darstellung mit Icons als einzigm Label	19
3.2	Hinzufügen von Textlabels zusätzlich zu den Icons, um die Verwechslung von 3.1 einzuschränken	19
3.3	Menü mit geringer hierarchischer Unterscheidung	20
3.4	Neustrukturierung des Menüs 3.3, um die Sichtbarkeit der Hierarchie zu erhöhen	20
3.5	Anzeige von Geräteentitäten ohne Differenzierung der Formatierung	20
3.6	Anzeige von Geräteentitäten mit Differenzierung der Formatierung	21
3.7	Gemeinsame Nutzung der Formatierung der Entität Gerät	21
3.8	Beispiel einer Schaltfläche, die nach dem Anklicken einen Spinner darstellt, der eine laufende Aktion anzeigt	21
3.9	Bei einer langsamen WLAN-Verbindung zoomt der Benutzer in eine <i>Site</i> (blauer Bereich), aber es scheint nichts zu passieren. Erst nach einigen Sekunden sind die Sensoren sichtbar.	22
3.10	Beispiel für UI-Komponenten-Skeletons, die vom Library PrimeNg bereitgestellt werden	22
3.11	Messung von nicht skalierten Bereichen, die die kognitive Belastung der Schnittstelle verstärken	23
3.12	Popup, das die Details einer <i>Site</i> präsentiert	23
3.13	<i>Toastr</i> nachricht von Gmail, mit der das Löschen einer E-Mail rückgängig gemacht werden kann	24
3.14	Seite mit der Liste der <i>Sites</i> eines Nutzers	24
3.15	Darstellung des Globalisierungszyklus eines Produkts	26
3.16	Screenshot von Push-Benachrichtigungen, die nach einer Fehlbedienung an den Benutzer gesendet werden	26
3.17	Icon eines Sensors, der auf der Feueralarmtaste verwendet wird	27

3.18	Icon, das in einer Stecknadel verwendet wird, um den Standort eines Feuers auf der Karte anzuzeigen	27
3.19	Darstellung eines Betrags als Badge	27
3.20	Darstellung eines Betrags als einfache Zahl neben der betreffenden Entität	27
3.21	Darstellung eines Ideationszyklus mit der Lotus-Blossom-Methode	29
3.22	Durchführung des ersten Ideationszyklus durch das Cloud-Team von Dryad mit der Lotus-Blossom-Methode	30
3.23	Durchführung des zweiten Ideationszyklus durch das Cloud-Team von Dryad mit der Lotus-Blossom-Methode	31
1	51
2	51
3	52
4	52
5	53
6	53

Tabellenverzeichnis

3.1	Zusammenfassung der ergonomischen Kriterien von Bastien und Scapin	18
-----	--	----

Listings

2.1	Beispiel für die Gestaltung einer Liste mit css	8
2.2	Beispiel für die Gestaltung einer Liste mit scss	8
2.3	Beispiel für die Verwendung der Komponente <i>p-paginator</i> , mit der eine Schnittstelle mit Paging einfach verwaltet werden kann.	9

Abkürzungsverzeichnis

CSS Cascading Style Sheets

DOM Document Object Model

I18n Internationalization

L10n Localization

NPM Node Package Manager

SASS Syntactically Awesome StyleSheets

UI User Interface

UX User Experience

XHR XMLHttpRequest

Anhang

.1 Ergebnis des zweiten Ideationszyklus mit der Lotus-Blossom-Methode

.1 Ergebnis des zweiten Ideationszyklus mit der Lotus-Blossom-Methode

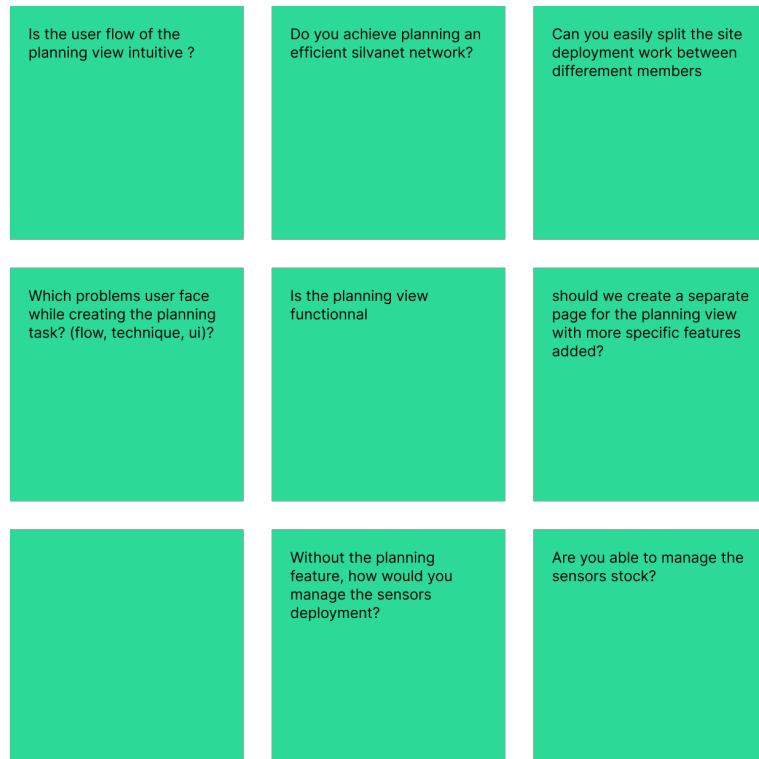


Abbildung 1



Abbildung 2



Abbildung 3

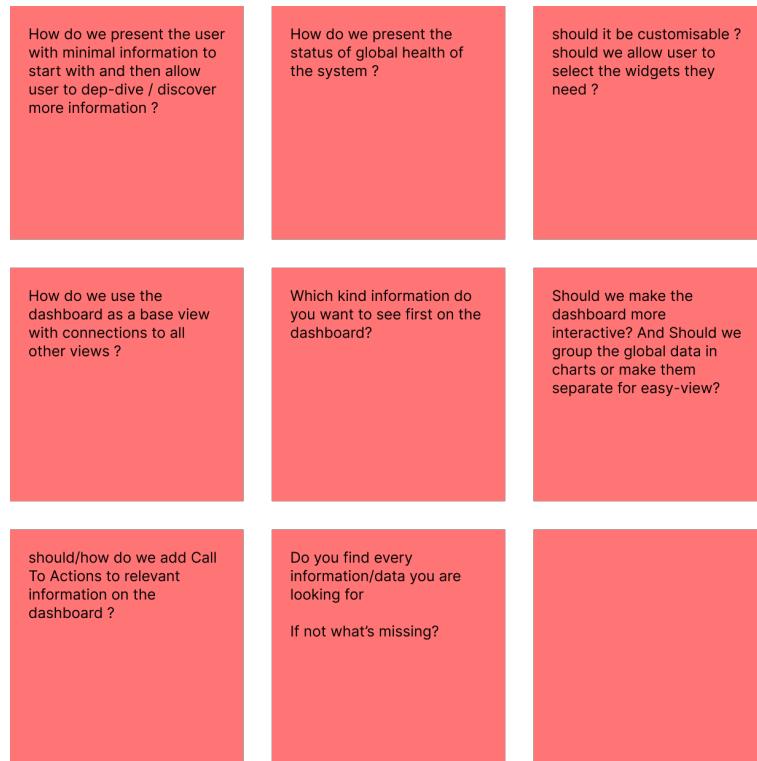


Abbildung 4

.1 Ergebnis des zweiten Ideationszyklus mit der Lotus-Blossom-Methode

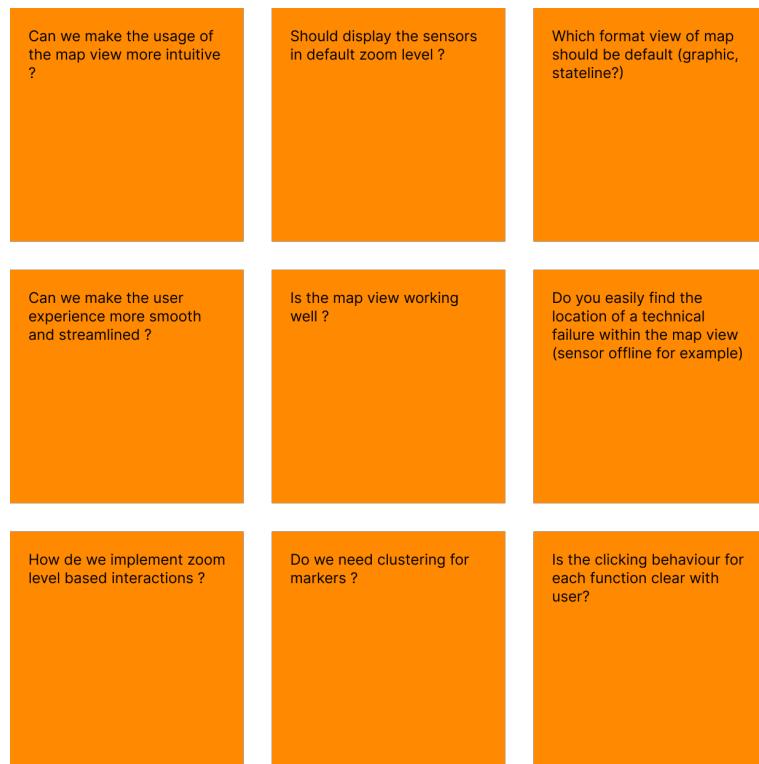


Abbildung 5

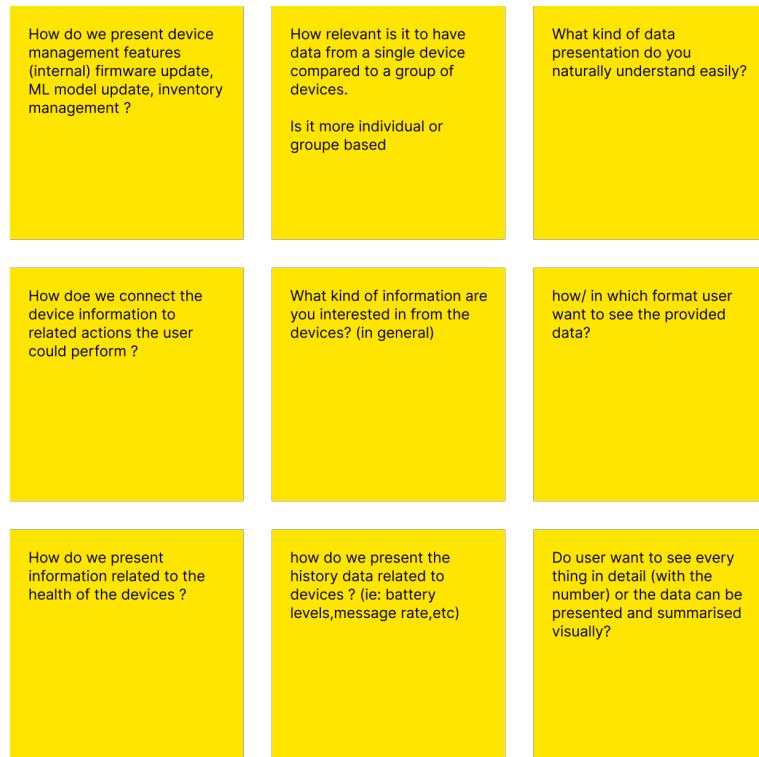


Abbildung 6

.2 Notizen aus der ergonomischen Inspektion der Webschnittstelle (Englisch)

Ergonomic inspection

This document refers to inspection on the dryad web application using the criteria of the research "[Ergonomic criteria for the evaluation of Human-computer Interfaces](#)" by BASTIEN and SCAPIN. The flow is the following. I will take all criteria in order and will check that the current interface respects them and if not propose a way to fix it.

1. Guidance	5
1.1. Prompting	5
1.1.1. Specify the use of the asterisk in the forms	5
1.1.1.1. <input checked="" type="checkbox"/> Problem	5
1.1.1.2. <input checked="" type="checkbox"/> Solution	5
1.1.2. Clear password format requirements	5
1.1.2.1. <input checked="" type="checkbox"/> Problem	5
1.1.2.2. <input checked="" type="checkbox"/> Solution	5
1.1.3. Use clear form labels	5
1.1.3.1. <input checked="" type="checkbox"/> Problem	5
1.1.3.2. <input checked="" type="checkbox"/> Solution	6
1.1.4. Avoid icons only labels	6
1.1.4.1. <input checked="" type="checkbox"/> Problem	6
1.1.4.2. <input checked="" type="checkbox"/> Solution	6
1.1.5. Define latitude/longitude data format in inputs label	6
1.1.5.1. <input checked="" type="checkbox"/> Problem	6
1.1.5.2. <input checked="" type="checkbox"/> Solution	6
1.1.6. Define weight data format in inputs label	7
1.1.6.1. <input checked="" type="checkbox"/> Problem	7
1.1.6.2. <input checked="" type="checkbox"/> Solution	7
1.1.7. Define min/max length on textarea with current status	7
1.1.7.1. <input checked="" type="checkbox"/> Problem	7
1.1.7.2. <input checked="" type="checkbox"/> Solution	7
1.1.8. Provide a title for each window	8
1.1.8.1. <input checked="" type="checkbox"/> Problem	8
1.1.8.2. <input checked="" type="checkbox"/> Solution	8
1.1.9. Disable next step in stepper	8
1.1.9.1. <input checked="" type="checkbox"/> Problem	8
1.1.9.2. <input checked="" type="checkbox"/> Solution	9
1.2. Grouping/Distinction by Location	9
1.2.1. Organize "Active sensors" in the dashboard total sensors area	9

1.2.1.1. Problem	9
1.2.1.2. Solution	9
1.2.2. Better localisation for filters menu	9
1.2.2.1. Problem	9
1.2.2.2. Solution	10
1.3. Grouping/Distinction by Format	10
1.3.1. Use common edition flow	10
1.3.1.1. Problem	10
1.3.1.2. Solution	11
1.3.2. Use unique design for Gateway/Site/SensorNode name	11
1.3.2.1. Problem	11
1.3.2.2. Solution	11
1.3.3. Add UI design for a specific role	12
1.3.3.1. Problem	12
1.3.3.2. Solution	12
1.4. Immediate Feedback	12
1.4.1. Better feedback for submit button	12
1.4.1.1. Problem	12
1.4.1.2. Solution	13
1.4.2. Add loading animation to site area	13
1.4.2.1. Problem	13
1.4.2.2. Solution	13
1.4.3. Add loading animation when fetching data	13
1.4.3.1. Problem	13
1.4.3.2. Solution	13
1.5. Legibility	14
1.5.1. Avoid line return in text	14
1.5.1.1. Problem	14
1.5.1.2. Solution	14
2. Workload	14
2.1. Brevity / Concision	15
2.1.1. Use proper area units	15
2.1.1.1. Problem	15
2.1.1.2. Solution	15
2.2. Brevity / Minimal Actions	15
2.2.1. Be able to request a particular page directly	15
2.2.1.1. Problem	15
2.2.1.2. Solution	15
2.2.2. Add routing functionality to the map	16
2.2.2.1. Problem	16

2.2.2.2. <input checked="" type="checkbox"/> Solution	16
2.3. Information Density	16
2.3.1. Reduce/improve density of a site overview	16
2.3.1.1. <input type="checkbox"/> Problem	16
2.3.1.2. <input checked="" type="checkbox"/> Solution	17
2.3.1.3. <input type="checkbox"/> Problem	18
2.3.1.4. <input checked="" type="checkbox"/> Solution	18
3. Explicit Control	19
3.1. Explicit User Action	20
3.1.1. Don't save data on alert modal close	20
3.1.1.1. <input type="checkbox"/> Problem	20
3.1.1.2. <input checked="" type="checkbox"/> Solution	20
3.2. User Control	20
3.2.1. Add an Undo button when changing a sensor fire alert state	20
3.2.1.1. <input type="checkbox"/> Problem	20
3.2.1.2. <input checked="" type="checkbox"/> Solution	20
4. Adaptability	21
4.1. Flexibility	22
4.1.1. Customize sites design	22
4.1.1.1. <input type="checkbox"/> Problem	22
4.1.1.2. <input checked="" type="checkbox"/> Solution	22
4.1.2. Customize what happened in case of fire	22
4.1.2.1. <input type="checkbox"/> Problem	22
4.1.2.2. <input checked="" type="checkbox"/> Solution	22
4.1.3. Localisation	22
4.1.3.1. <input type="checkbox"/> Problem	22
4.1.3.2. <input checked="" type="checkbox"/> Solution	22
4.1.4. Customize the units and format	23
4.1.4.1. <input type="checkbox"/> Problem	23
4.1.4.2. <input checked="" type="checkbox"/> Solution	23
4.1.5. Customize sidebar layout	23
4.1.5.1. <input type="checkbox"/> Problem	23
4.1.5.2. <input checked="" type="checkbox"/> Solution	23
4.2. User Experience	24
4.2.1. Add advanced search bar	24
4.2.1.1. <input type="checkbox"/> Problem	24
4.2.1.2. <input checked="" type="checkbox"/> Solution	24
5. Error Management	25
5.1. Error Protection	26
5.1.1. Disable draggable gateway in "edit Gateway"	26

5.1.1.1. Problem	26
5.1.1.2. Solution	26
5.1.2. Better input validation	26
5.1.2.1. Problem	26
5.1.2.2. Solution	26
5.2. Quality of Error Messages	26
5.2.1. Improve error messages	26
5.2.1.1. Problem	26
5.2.1.2. Solution	26
5.2.2. Add report a bug option when "Unknown error" occurs in request	27
5.2.2.1. Problem	27
5.2.2.2. Solution	27
5.3. Error Correction	27
6. Consistency	28
6.1. Consistency	28
6.1.1. Keep fire icons consistent	28
6.1.1.1. Problem	28
6.1.1.2. Solution	28
6.1.2. Keep action buttons format consistent	28
6.1.2.1. Problem	28
6.1.2.2. Solution	28
6.1.3. Use common design for entity amount	29
6.1.3.1. Problem	29
6.1.3.2. Solution	29
7. Significance of Codes	30
7.1. Significance of Codes	30
8. Compatibility	30
8.1. Compatibility	31
8.1.1. Update lat/long together	31
8.1.1.1. Problem	31
8.1.1.2. Solution	31

1. Guidance

1.1. Prompting

1.1.1. Specify the use of the asterisk in the forms

1.1.1.1. ✗ Problem

It's actually not obvious for every users what (*) means in a form.

1.1.1.2. ✓ Solution

Simply add an explanation message at the top like “Fields with the symbol (*) are required!”.

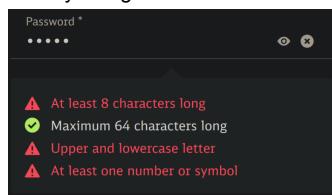
! * fields are required !

1.1.2. Clear password format requirements

1.1.2.1. ✗ Problem

1.1.2.2. ✓ Solution

Show to user what's missing or not by using a list:



1.1.3. Use clear form labels

1.1.3.1. ✗ Problem

Some forms labels are not clear for new clients like:

- In add user form: **Ns role, role, client**
- In add experiment form: **Hotplate Temp**

1.1.3.2. Solution

If the label is not clear → add a little explanation of what is expected in a simple helper text or a helper tooltip:

- Example of input with helper text



- Example of input with helper tooltip



1.1.4. Avoid icons only labels

1.1.4.1. Problem

The icons can be interpreted in different ways from one user to another. Here is an example of confusing interface:

Last sending alert sensor	
	03.05.2022
	10:07:40
	27b2
	52.85683
	13.760759

Sensor data presented in the alert center at <http://localhost:4200/alert-centre>

1.1.4.2. Solution

In order to avoid interrogation or misinterpretation by the user, it is recommended to add a textual label next to the icon.

1.1.5. Define latitude/longitude data format in inputs label

1.1.5.1. Problem

Some forms input like for latitude/longitude, the desired format is not specified.

1.1.5.2. Solution

Specify in the label what the required format is like:

1.1.6. Define weight data format in inputs label

1.1.6.1. ✗ Problem

The Burn Material Amount input is not clear:

1.1.6.2. ✓ Solution

Simplest solution would be to specify in the label what the required format is like:

Another solution is to provide an input with a dropdown addon like:

1.1.7. Define min/max length on textarea with current status

1.1.7.1. ✗ Problem

Long text inputs have probably a min or max length required. The user needs to be informed about it which is currently not the case on several textareas like:

- Edit gateway popup at <http://localhost:4200/messages>

1.1.7.2. ✓ Solution

Specify in the label or in a helper text below the rules and the current status like for example:

Message *

 Lorem ipsum dolor

Minimum 50 expected characters (currently 17)

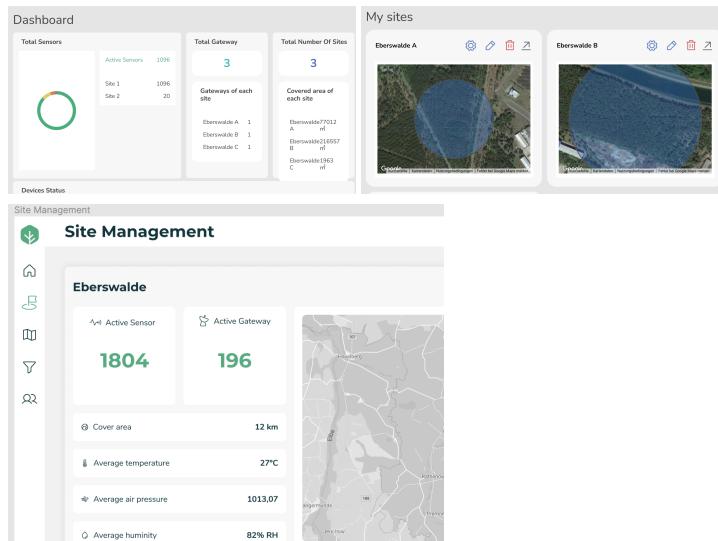
1.1.8. Provide a title for each window

1.1.8.1. ✗ Problem

Good prompting guides the users to navigate in the application. Currently there is no title on each app window. For a new user it could be confusing what the current opened window is.

1.1.8.2. ✓ Solution

Adding a title (h1) on each windows help the user to easily keep track on his location:



1.1.9. Disable next step in stepper

1.1.9.1. ✗ Problem

When using a stepper component (see in “add a device” in <http://localhost:4200/sites>), the next step or other steps are clickable.

1.1.9.2. Solution

If some steps are required make the following disabled to keep consistency with the “next” button.

1.2. Grouping/Distinction by Location

1.2.1. Organize “Active sensors” in the dashboard total sensors area

1.2.1.1. Problem

An hierarchical list should be organized according to a defined parameter which is not clear when observing the mockup.

1.2.1.2. Solution

Order the sites alphabetically or according to the number of sensors. The best case would be to let the user sort the sites when clicking on the label and symbolize the order with a simple arrow like:

STATUS ↓	PROJECT	TEAM
----------	---------	------

1.2.2. Better localisation for filters menu

1.2.2.1. Problem

The filter options are on the same level as the section title. It's good enough for tiny lists but if it grows then it can be hard to see the hierarchy.

Filter by devices

Gateways

Sensors

Filter by sites

Eberswalde A

Eberswalde B

Eberswalde C

1.2.2.2. ✓ Solution

Just adding a tiny more padding helps the human eyes to visualize a hierarchy. Additionally to that, a system of collapse can be used with the help of a chevron icon to improve the presence of a hierarchy to the user.

v Filter by devices

Gateways

Sensors

> Filter by sites

1.3. Grouping/Distinction by Format

1.3.1. Use common edition flow

1.3.1.1. ✗ Problem

In some areas in the app, the “edition” action flow is done in several ways. That should be normalized. Here are some non-normalized modal:

- In the “Update payload format” at <http://localhost:4200/sites>
-

1.3.1.2. Solution

Use the more common flow which is to open a modal with a submit button.

1.3.2. Use unique design for Gateway/Site/SensorNode name

1.3.2.1. Problem

On multiple views there is the display of names of devices. However the names are presented as pure text and in some cases (like in <http://localhost:4200/messages>) it's really hard to identify what the name relates to.

1.3.2.2. Solution

Adding a simple/lean design to device names according to its type will help the user to read a dense interface.

Device	Site	Device type
Eberswalde BG xBvvX	Eberswalde A	Gateway
Eberswalde SN ofUAI	Eberswalde A	SensorNode

List of devices as pure text

Device	Site	Device type
Eberswalde BG xBvvX	Eberswalde A	Gateway
Eberswalde SN ofUAI	Eberswalde A	SensorNode

List of the same devices with it's matching device type format

The chosen format can be then used across the different pages/views:





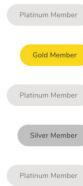
1.3.3. Add UI design for a specific role

1.3.3.1. ✗ Problem

The users table at <http://localhost:4200/users> is just composed of string without any visual help.

1.3.3.2. ✓ Solution

The table can be improved by using an additional UI design for the roles like the following:



Example from <https://dribbble.com/shots/8573883-Restoranku-Discount-Member-Dashboard>

1.4. Immediate Feedback

1.4.1. Better feedback for submit button

1.4.1.1. ✗ Problem

When the submit button is clicked, no feedback is provided to the user. For most cases the action is done pretty quickly but the case can occur that the action takes time (bad connexion for example) and the user needs to be informed that the application is processing. That concerns:

- Edit Site popup at <http://localhost:4200/sites> when clicking submit
- Link new payload popup at <http://localhost:4200/sites> when clicking submit

1.4.1.2. Solution

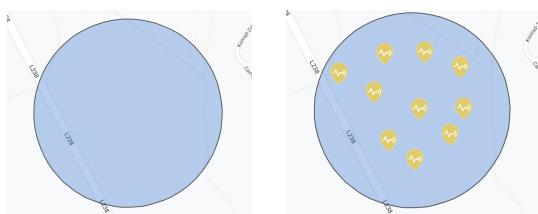
After a click on a submit button, an animation should be triggered on the button or an overlay until the processing is done. Here is an example of a loading animation on a button:



1.4.2. Add loading animation to site area

1.4.2.1. Problem

The different devices of a site are fetched at a specific zoom level. However, if the internet connection is not fast enough, the user has no feedback that a request takes place in the background.



With slow wifi connection, the user zooms in the site (blue area) but nothing seems to happen. Only after several seconds the sensors are visible.

1.4.2.2. Solution

Adding a loading animation on the site area will indicate to the user that something is processing. An UI example can be waves coming from the center of the circle zone like this <https://cdn.dribbble.com/users/26878/screenshots/3777480/39-location.gif>.

1.4.3. Add loading animation when fetching data

1.4.3.1. Problem

There is no loading feedback during the fetch of

- The sites in <http://localhost:4200/sites>
- The sites in the filters in <http://localhost:4200/messages>
- The devices in the table at <http://localhost:4200/messages>
- The users in the table at <http://localhost:4200/users>
- The settings in the table at <http://localhost:4200/settings>
- The gateways and sites at <http://localhost:4200/dashboard>
- The Contact SOS modal at <http://localhost:4200/alert-centre>

1.4.3.2. Solution

Add some nice loading like skeletons (see <https://primefaces.org/primeng/skeleton>):



Example of a card skeleton component

1.5. Legibility

1.5.1. Avoid line return in text

1.5.1.1. Problem

From the recommendation: "When space for text display is limited, display a few long lines of text rather than many short lines of text".

1.5.1.2. Solution

Avoid as much possible to have line return in the interface like in:

- Site descriptions at <http://localhost:4200/alert-centre>
- Table header in site planning at <http://localhost:4200/packet/18>
- Values in table at <http://localhost:4200/settings>
- Values in "Update payload format" table at <http://localhost:4200/sites>

2. Workload

2.1. Brevity / Concision

2.1.1. Use proper area units

2.1.1.1. Problem

Area size are displayed in the interface but not always with an according unit that is easy to read:

Covered area of each site	
Eberswalde A	77012 m ²
Eberswalde B	216557 m ²
Eberswalde C	1963 m ²

2.1.1.2. Solution

Use proper units like km², ha, ... according to the order of size (also a good idea would be to let the user choose between m² and ha). See in:

- In the dashboard at <http://localhost:4200/dashboard>
 - In a site at <http://localhost:4200/sites>
 - In the map in the site dialog (the unit used is "sq.m")
-

2.2. Brevity / Minimal Actions

2.2.1. Be able to request a particular page directly

2.2.1.1. Problem

It's really important to limite as much as possible the steps users must go through in the application. Therefore all "important" page sections need to be accessible by a shareable URL.

2.2.1.2. Solution

Provide an URL endpoint to directly access the following app sections:

- In the dashboard, the current "devices status" type should be kept in the URL as a query string to avoid the user to always click to go to the non default one

- A specific site at <http://localhost:4200/sites> so that the user doesn't need to click again on the toggle icon. A good endpoint could be `/sites/{siteid}`
- Create a site. Example of an endpoint is `/sites/_create` (the use of the underscore in `_create` is to denote that `create` is an action and not an entity id following the REST convention)
- Edit a specific site → `/sites/{siteid}/_edit`
- Delete a specific site → `/sites/{siteid}/_delete`
- In the `/messages` endpoint be sure to keep the user filters as query strings in the URL. That will allow him to share it easily to others

2.2.2. Add routing functionality to the map

2.2.2.1. Problem

There is no routing functionality in the map which makes it difficult to explore properly and share to other users. If a user jumps to a specific site or sensor, an accidental refresh of the page will forget its location on the map and the user will have to do all the travel again.

2.2.2.2. Solution

The map should be reactive to routing change in the scope of the `/map` endpoint. An example would be that the endpoint `/map/sites/42` makes the map jump to the given site. Here are some ideas of endpoints:

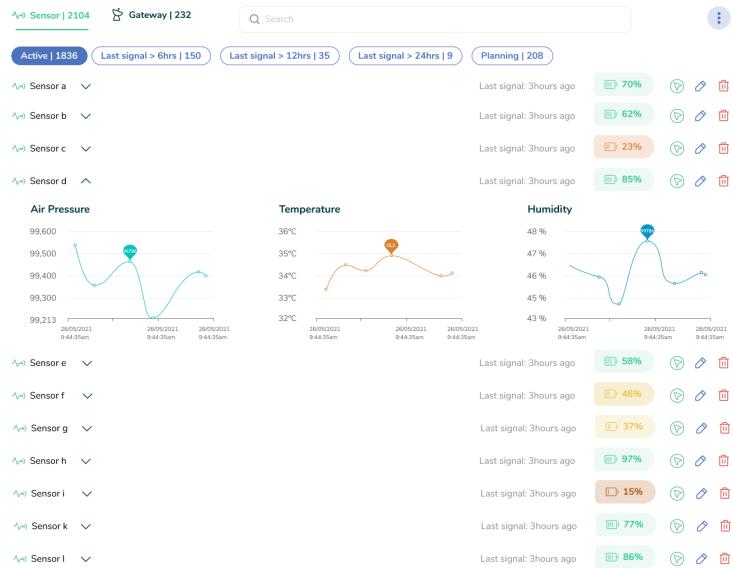
- `/map/sites/{siteid}`
 - `/map/sites/{siteid}/sensors/{sensorid}`
 - `/map/sites/{siteid}/gateways/{gatewayid}`
-

2.3. Information Density

2.3.1. Reduce/improve density of a site overview

2.3.1.1. Problem

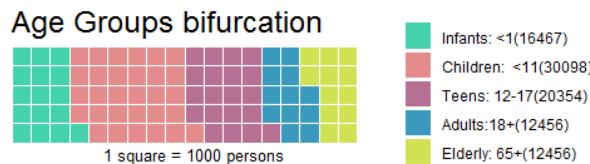
The user should be able to see all the sensors on the site. However, representing all of them as a table takes a lot of space on the interface and this without necessarily adding much advantage to the user.



Current design of the sensors section in the dashboard.

2.3.1.2. Solution

The main goal of this section should be to group the sensors according to several filters. In this case it would be nice to present the filtered devices in a way that the user can easily see some groups emerging. An interesting representation for that would be a heatmap like:

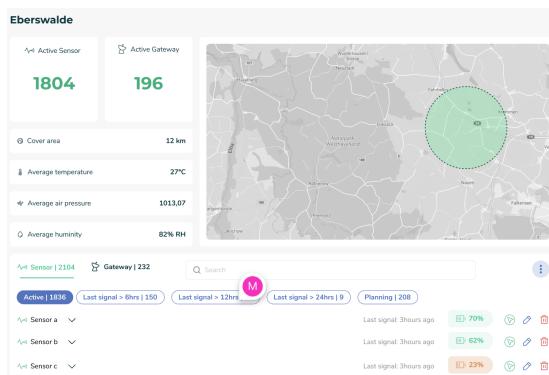


The heatmap should be able to group the sensors according to the “last signal”, “battery”, “air pressure”, “temperature” and “humidity”. The user would be then be able to select a unique sensor (one of the squares of the heatmap) to check its data more in details:

- Name
- Time since last signal
- Battery level
- Air pressure
- Temperature

- Humidity

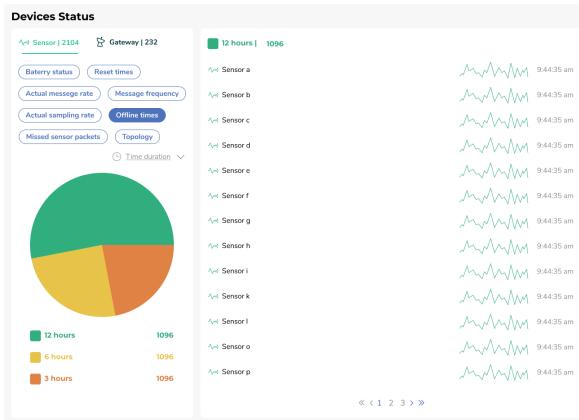
The user will be able to also select directly a group of sensors or multiple sensors. The result of that will be a similar dashboard to the one for a single sensor but with “all data for each sensor”. To do so, the “air pressure”, “temperature” and “humidity” charts are going to be a multiline chart. The representation of the “Time since last signal” and “Battery level” could then be a simple pie chart. Also when selecting a single or multiple sensors, the map above the section could be updated to present only the selected sensors:



2.3.1. Reduce/improve density of a dashboard sensors overview

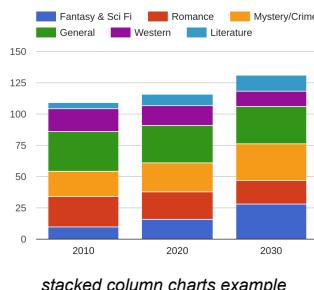
2.3.1.3. X Problem

It's exactly the same issue as the one above. Here the user can't really digest all the information and draw conclusions:



2.3.1.4. Solution

The different sensors data could be represented across a “stacked column charts” where every column is a site like:



stacked column charts example

3. Explicit Control

3.1. Explicit User Action

3.1.1. Don't save data on alert modal close

3.1.1.1. Problem

If you edit a gateway in a site in <http://localhost:4200/sites>, a modal opens. If you change the value of the gateway and close the popup, there is a toast message that indicates that the gateway has been updated. That's probably not what the user expected.

3.1.1.2. Solution

As mentioned in the "Ergonomic Criteria for the Evaluation of Human-Computer Interfaces" document, an interface should "*Always require a user to take an explicit ENTER action to initiate processing of entered data*". In this case, closing a modal shouldn't have a side effect. The problem also occurs in:

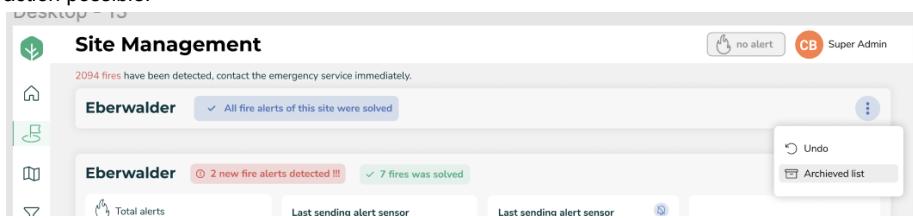
- The edit device modal at <http://localhost:4200/messages>

3.2. User Control

3.2.1. Add an Undo button when changing a sensor fire alert state

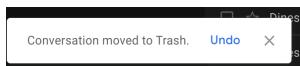
3.2.1.1. Problem

When there is a fire, the alert center page at <http://localhost:4200/alert-centre> enables the user to change a state of sensors to "false alarm" for example. However when done there is no undo action possible.



3.2.1.2. Solution

Add an Undo action button that return to the previous state like you can see on gmail when deleting a mail:



Example of Undo button in gmail interface

4. Adaptability

4.1. Flexibility

4.1.1. Customize sites design

4.1.1.1. Problem

It's recommended to give the user as much flexibility as possible to customize an interface. Currently all the sites are listed at <http://localhost:4200/sites>. However there is no way given to the user to distinguish one site from another.

4.1.1.2. Solution

Adding the possibility to customize the visual of a site in this list will definitely help users to easily distinguish each site. Here are some customization ideas:

- Change the background color of a site panel with given colors
- Change the order of the sites with a draggable sortable list
- Change the map view to a custom picture

4.1.2. Customize what happened in case of fire

4.1.2.1. Problem

In case of fire, it's a good idea to let the user customize what should happen for a specific site.

4.1.2.2. Solution

When creating a site and in the settings page, the interface should let the user configure some actions for a given site. Here are some ideas:

- Send a push notification
- Send a sms to given numbers
- Send an email to given emails

4.1.3. Localisation

4.1.3.1. Problem

Dryad wants to sell products worldwide. However the current interface is in english only but all the users are not native english speakers.

4.1.3.2. Solution

We should give the user the possibility to choose in which language the interface should be displayed.

4.1.4. Customize the units and format

4.1.4.1. Problem

Worldwide some units and formats are not the same so the interface shouldn't force the user to use non common ones.

4.1.4.2. Solution

The interface should let the user choose in which units/format some data should be displayed.
Here some concerned data:

- Date
- Time
- Distance (metric vs Customary system)
- Area
- Temperature
- Pressure
- Float number
- Volume?
- Weight and mass?

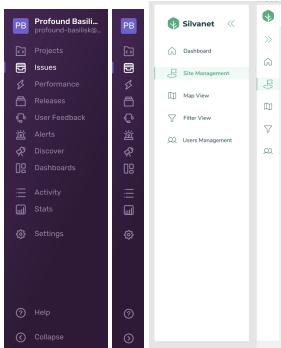
4.1.5. Customize sidebar layout

4.1.5.1. Problem

Currently the sidebar only presents icons which can be confusing for some users that are not sure what a flag icon refers to.

4.1.5.2. Solution

A simple solution is to add labels to each icons that you can collapse just like:



Example of a collapsible sidebar from the Sentry dashboard.

4.2. User Experience

4.2.1. Add advanced search bar

4.2.1.1. X Problem

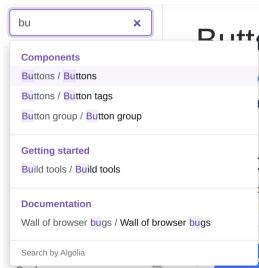
Currently there is almost only one way to go to a specific page in the app. However, an experienced user would probably like to easily go to a specific site or sensor.

4.2.1.2. ✓ Solution

The app could include an advanced search bar that proposes actions to the user on inputs. Here is an example: the user start to type the name of a site → The search bar propose to:

- Sites
 - Open the site ... → the user is redirected to the sites endpoint with the right site opened
 - Open the site ...
- Map
 - Navigate to ... → the user is redirected to the map on the right site location
 - Navigate to ...

A good UI example is the algolia search bar:



Note: this can only be done when every page has a proper router endpoint.

5. Error Management

5.1. Error Protection

5.1.1. Disable draggable gateway in “edit Gateway”

5.1.1.1. Problem

In edit gateway popup, the gateway can be moved on the map without asking the user so which is really confusing and error prone.

5.1.1.2. Solution

There should be an edit button to trigger the edition mode and ask the user if he agreed to move the gateway to the new location.

5.1.2. Better input validation

5.1.2.1. Problem

As mentioned in the paper, “*It is preferable to detect errors before validation rather than after*”.

5.1.2.2. Solution

Each input in the interface should display what the error is and prevent those. Here is a list of what should be taken into account:

- No empty spaces
- No characters in number inputs
- The decimal value should be the expected one (comma vs period)

5.2. Quality of Error Messages

5.2.1. Improve error messages

5.2.1.1. Problem

Most error messages are not descriptive enough.

5.2.1.2. Solution

Improve error messages by using task-oriented wording.

5.2.2. Add report a bug option when “Unknown error” occurs in request

5.2.2.1. Problem

Some “Unknown errors” can happen when the frontend communicates with the backend (error 500 for example). In most cases the users can’t solve the issue by themselves and there is currently no help center.

5.2.2.2. Solution

When such an error occurs, there should be a “report bug” button in the toast message that opens an email with a pre-filled title, contact and body.

The title should include:

- The site id
- What was the action

The body of the email should include the user message that explain how to reproduce the error but also the following data in a hidden section (by using html5 <details> tag):

- The request with the http verb used
- The request body
- The response's http status code
- The error message in the response

5.3. Error Correction

6. Consistency

6.1. Consistency

6.1.1. Keep fire icons consistent

6.1.1.1. Problem

When detecting a fire, the fire alert button is presented with the sensor icon where on the map another is used:



Icon of a sensor used on the fire alert animation



Icon used in a pin to indicate the location of a fire on the map

6.1.1.2. Solution

The visual representation of an entity must be unique as much as possible to facilitate its memorization and thus allow a faster recognition later on. That's why it would be recommended to use the flame icon on each fire related UI as the sensor icon is already used to represent a sensor.

6.1.2. Keep action buttons format consistent

6.1.2.1. Problem

The main action buttons like **EDIT** or **DELETE** have different layout/design in pages:

- /experiments in tab **training**
- /experiments in tab **experiment**

6.1.2.2. Solution

Respect on those pages the original design: Big Icons in a button with no background-color or border. Actions should always be displayed (not just on mouse hover).



6.1.3. Use common design for entity amount

6.1.3.1. ✗ Problem

The interface often presents an entity with its amount. However the design of the amount is not always the same and that doesn't help the user to easily identify it. An example is the difference between the amount of the devices and the fire alarm. One is presented as a badge and the other just display the amount next to it:



6.1.3.2. ✓ Solution

A decision should be made to determine which format to use when presenting an amount. The badge seems to be a good candidate for that.

7. Significance of Codes

7.1. Significance of Codes

8. Compatibility

8.1. Compatibility

8.1.1. Update lat/long together

8.1.1.1. Problem

It's common for a user to work with latitude and longitude together as. However in some inputs the latitude and longitude can be edited separately.

8.1.1.2. Solution

A user will be able to modify a single latitude or longitude, but the form should always present both as it's more natural.

Kolophon

Dieses Dokument wurde mit der L^AT_EX-Vorlage für Abschlussarbeiten an der htw saar im Bereich Informatik/Mechatronik-Sensortechnik erstellt (Version 1, Juni 2022). Die Vorlage wurde von Yves Hary und André Miede entwickelt (mit freundlicher Unterstützung von Thomas Kretschmer, Helmut G. Folz und Martina Lehser). Daten: (F)10.95 – (B)426.79135pt – (H)688.5567pt