# Project Bookkeeper

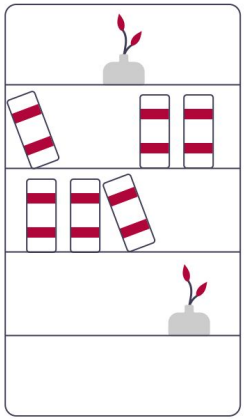
## Interim Presentation - Team Rot

SSE 2022

Hasso-Plattner-Institut

# Agenda

1. Demo
2. Aktueller Stand auf Roadmap
3. Änderungen seit Kickoff
4. Team & Projekt-Organisation
5. Technology Deep Dive: Notification Inbox
6. Team-Statistiken
7. Learnings



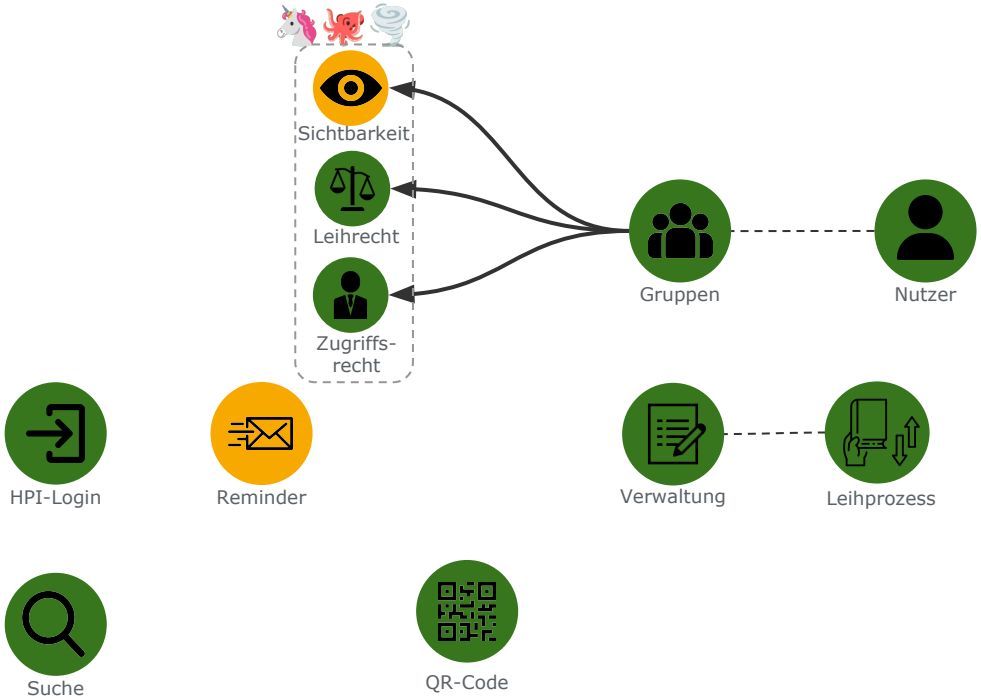
# Project Objective

*„Create a compelling system to manage and share private and organization assets.”*

# Live Demo

[Demo]

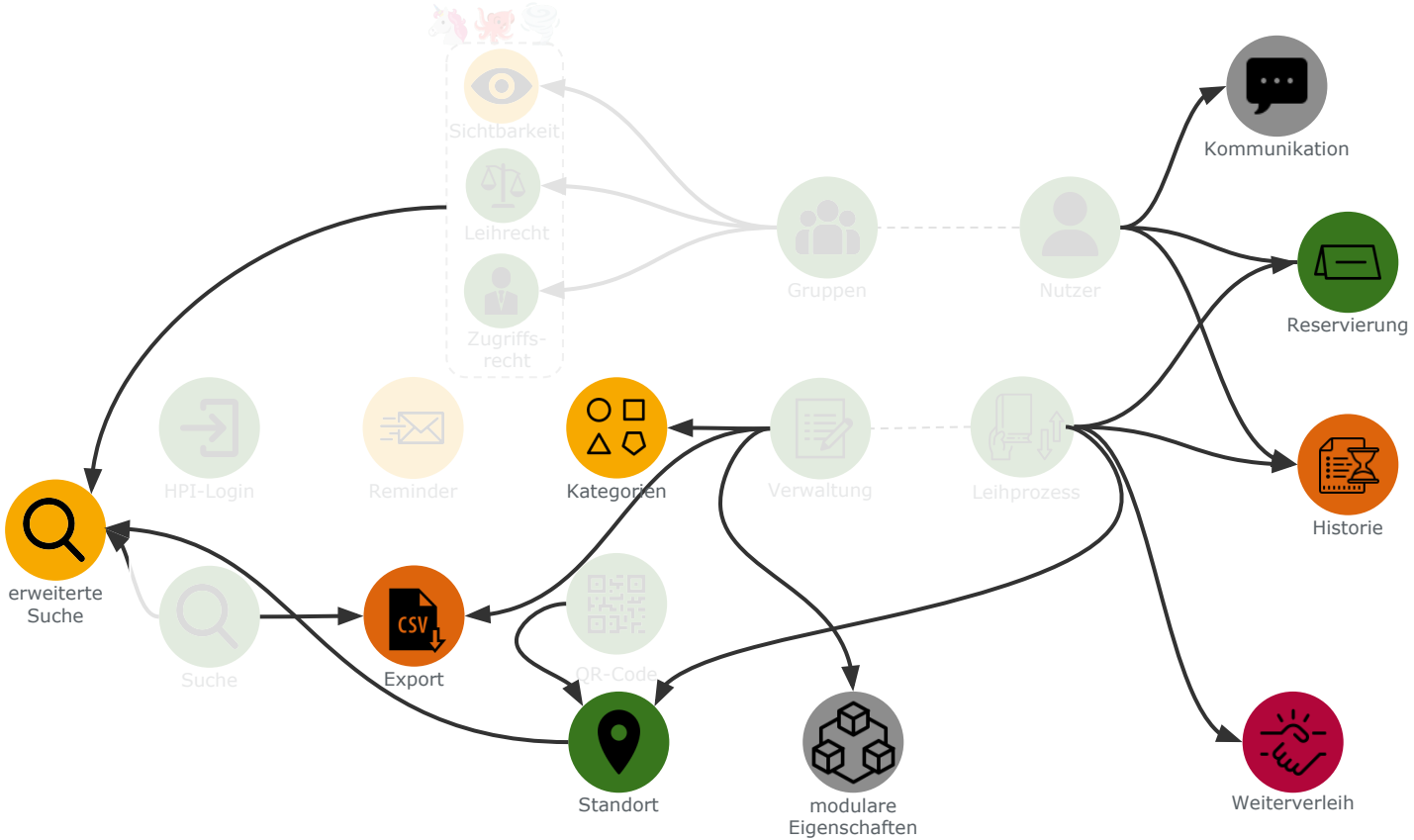
# Roadmap: Chapter I




Team CL
Team GdM
Team HP
Team TR
Team BA
Team FN

Done
In Progress
Product Backlog
Draft
Rejected

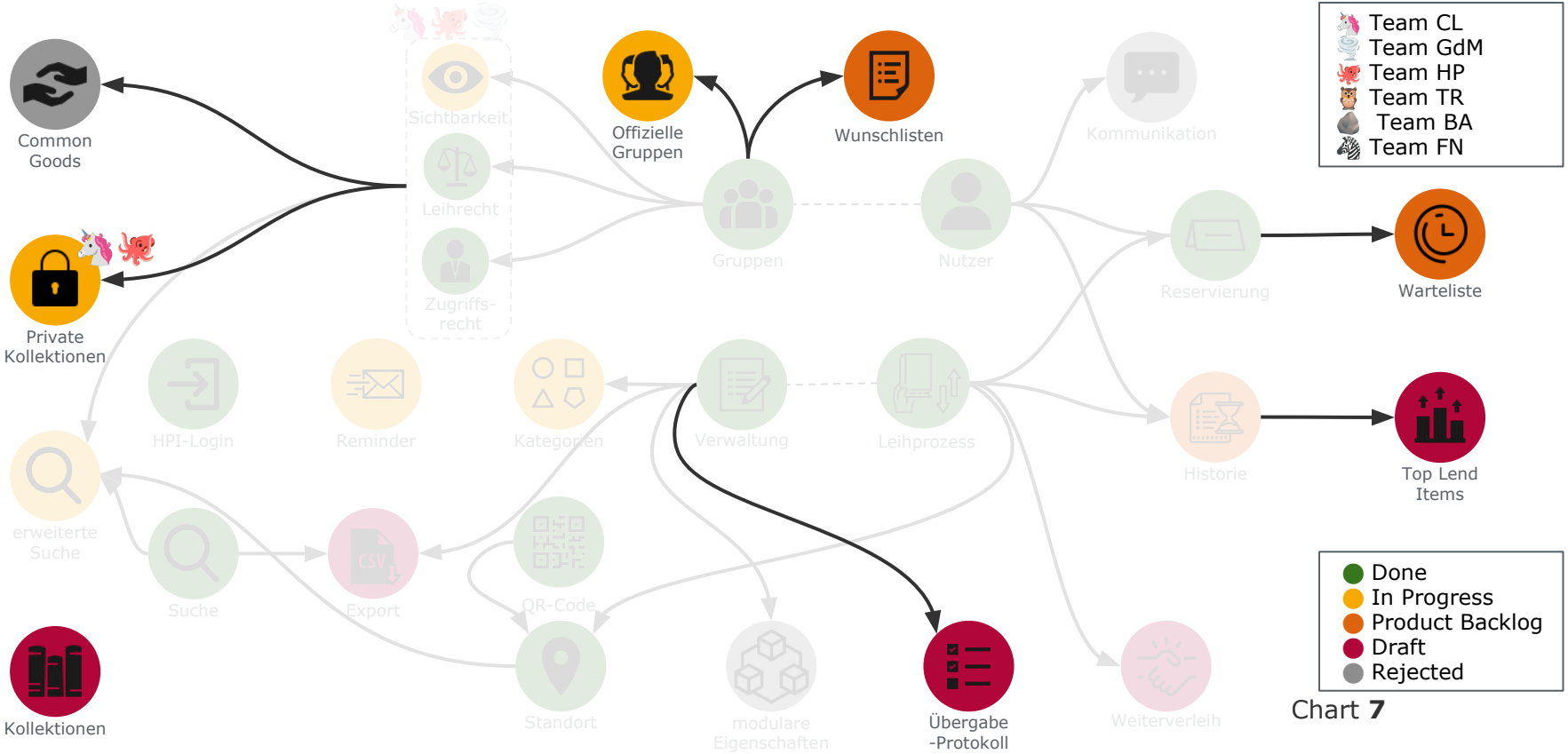
# Roadmap: Chapter II




Team CL
Team GdM
Team HP
Team TR
Team BA
Team FN

Done
In Progress
Product Backlog
Draft
Rejected

# Roadmap: Chapter III



# Roadmap: Timeline

HPI Hasso Plattner Institut

**Sprint 1**

**14.11. — Kickoff**

**14.11. - 27.11.**

→ Item CRUD-Operationen
→ Item-Ansicht, UI/UX für Homescreens
→ Login mit OIDC

**Sprint 2**

**28.11. - 11.12.**

→ QR-Code-Integration
→ Standorttracking
→ Ausleih- und Rückgabeprozess
→ Notification Inbox auf Homescreen

# Roadmap: Timeline


**12.12. - 08.01.**

→ Gruppenverwaltung und Managementrechte für Items

→ Reservierung von Items

→ Kategorienfilterung

**09.01. — Intermediate Presentation**

# Roadmap: Timeline


**09.01.-29.01.**
→ Wunschliste
→ Ausleihhistorie
→ Export
→ Warteliste

**30.01.-05.02. — Final Presentation Preparation**

**06.02. — Final Presentation & Project End**

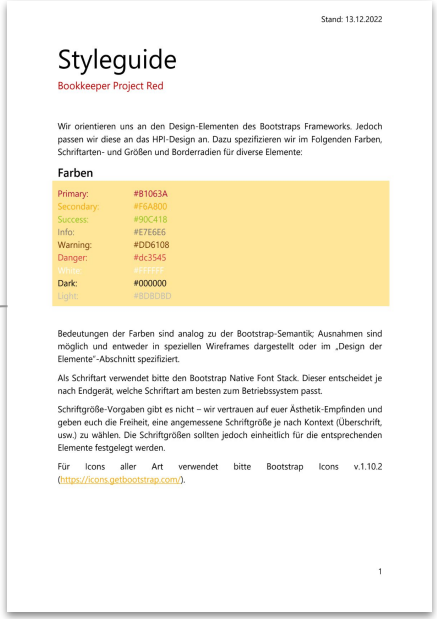
# Änderungen seit Kickoff

Neu hinzugekommen

## Der Styleguide

### Farben

| | |
|---|---|
| Primary: | #B1063A |
| Secondary: | #F6A800 |
| Success: | #90C418 |
| Info: | #E7E6E6 |
| Warning: | #DD6108 |
| Danger: | #dc3545 |
| White: | #FFFFFF |
| Dark: | #000000 |
| Light: | #BDBDBD |



Stand: 13.12.2022

Styleguide

Bookkeeper Project Red

Wir orientieren uns an den Design-Elementen des Bootstraps Frameworks. Jedoch passen wir diese an das HPI-Design an. Dazu spezifizieren wir im Folgenden Farben, Schriftarten- und Größen und Borderradien für diverse Elemente:

**Farben**

Primary: #B1063A
Secondary: #F6A800
Success: #90C418
Info: #E7E6E6
Warning: #DD6108
Danger: #dc3545
White: #FFFFFF
Dark: #000000
Light: #BDBDBD

Bedeutungen der Farben sind analog zu der Bootstrap-Semantik; Ausnahmen sind möglich und entweder in speziellen Wireframes dargestellt oder im „Design der Elemente"-Abschnitt spezifiziert.

Als Schriftart verwendet bitte den Bootstrap Native Font Stack. Dieser entscheidet je nach Endgerät, welche Schriftart am besten zum Betriebssystem passt.

Schriftgröße-Vorgaben gibt es nicht – wir vertrauen auf euer Ästhetik-Empfinden und geben euch die Freiheit, eine angemessene Schriftgröße je nach Kontext (Überschrift, usw.) zu wählen. Die Schriftgrößen sollten jedoch einheitlich für die entsprechenden Elemente festgelegt werden.

Für Icons aller Art verwendet bitte Bootstrap Icons v.1.10.2 (https://icons.getbootstrap.com/).

1

Stand: 13.12.2022

**Design der Elemente**

Folgende Eigenschaften sollen von Bootstrap-Elementen eingehalten werden:

| | |
|---|---|
| Buttons und Buttongroups | Border-Radius: 5px<br>Fontstyle: semibold |
| Alerts | Border-Radius: 5px<br>Fontstyle Beschreibung: regular |
| Media Objects | Border-Radius: 5px |
| Links | Farbe: Secondary |
| Modal | Border-Radius: 8px |
| Überschriften | Fontstyle: semibold |

Der Styleguide wird ggf. laufend weiterentwickelt und dadurch neue User Stories generieren.

**Tipp:** ab Bootstrap v5.2.0 gibt es einige neue globale CSS Variablen, mit denen ihr die Bootstrap Elemente global anpassen könnt. Daher könnte sich ein Upgrade auf v5.2.0 lohnen.

Zudem ist er absichtlich nicht vollständig – wir möchten den Entwicklern die Freiheit geben, anhand der gegebenen Spezifikationen weitere sinnvolle Designentscheidungen zu treffen und diesen Styleguide nur als Orientierung zu verstehen. Sollte es zu Fragen oder Unklarheiten kommen, dürft ihr gerne Kontakt mit uns aufnehmen:

Sebastian Wilke

Discord: Sebastian | Encotric#7794
Telegram: https://t.me/encotric

Maximilian Speer

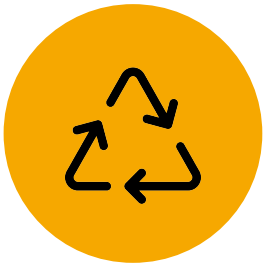
Discord: Max Speer#5520
Telegram: https://t.me/Kenneth587

2

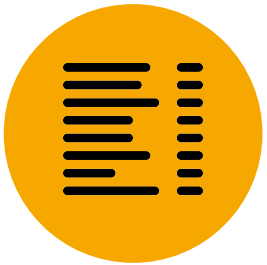
# Änderungen seit Kickoff

Der **Styleguide** bietet einige **Vorteile** für unsere Scrum-Teams:

**International verständlich**
(deutsche + englische Version)



**Wiederverwendung**
von vorhandenen Bootstrap-Elementen



**Einheitliche, genaue Vorgaben**
für konsistente Designs

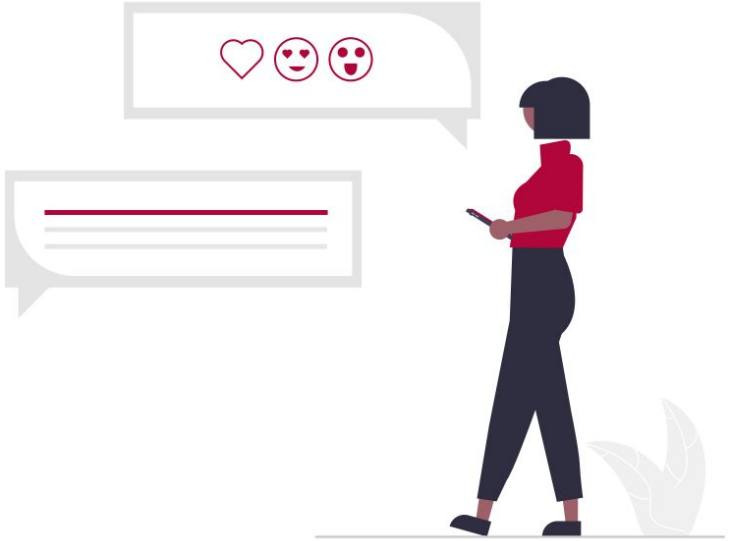
**Schnelles Erfassen**
der Vorgaben durch minimalistisches Dokument

# Änderungen seit Kickoff

Bugs können ab sofort im Discord-Channel **#bugs** unkompliziert gemeldet und diskutiert werden.

**Schreibt bitte immer dazu:**

- was das **aktuelle Verhalten** ist
- wie das **gewünschte Verhalten** aussehen sollte
- ggf. **Schritte**, um den Bug zu **reproduzieren**



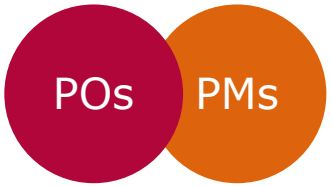
# Team und Projekt-Organisation

# Team und Projekt-Organisation


- wöchentliche Meetings
- Synchronisation: Entwerfen und Aufteilen neuer Features
- zweiwöchige Kundentreffen basierend daraus
- Erstellung der User Stories erfolgt individuell


- zweiwöchige Meetings
- Synchronisation: Besprechung teamübergreifender Entscheidungen, Probleme, Fragen
- Vorbereitung von Präsentationen, Reports

# Team und Projekt-Organisation




- zweiwöchige Meetings
- **Teamübergreifende Probleme/Fragen** bzgl. Produkt, z.B.:
  - Deployment
  - Github Board Organisation
  - Verknüpfung der verschiedenen Komponenten
  - teamübergreifende Abhängigkeiten
- **Planung & Aufteilung** der Präsentationen
- Organisation der generellen **Kommunikationen**

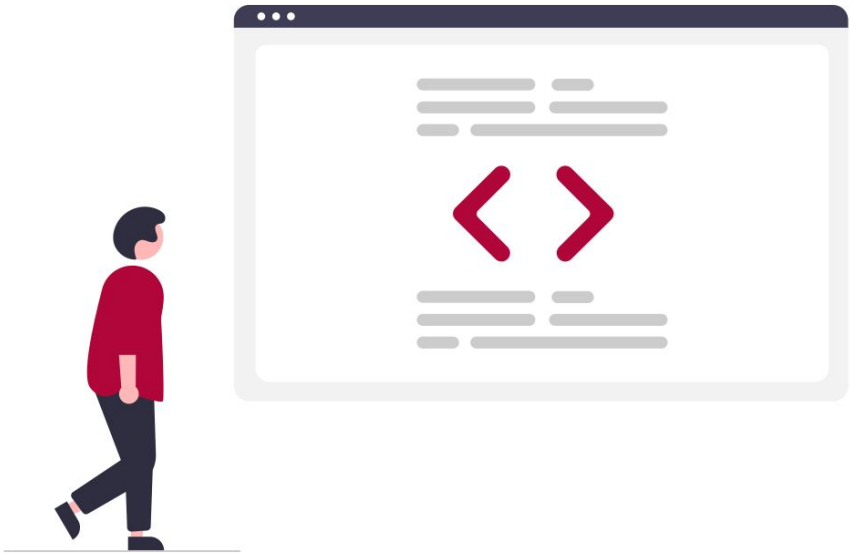
# Team und Projekt-Organisation

**Was lief gut?**

+ strukturierte Meetings
+ etablierte Kommunikationskanäle
+ Dokumentenzugriff per Drive
+ Notion Organisation

**Was wollen wir verbessern?**

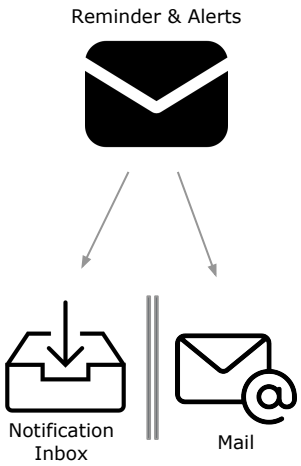
- Verantwortlichkeiten zwischen Teams noch besser koordinieren
- noch mehr teamübergreifende Kommunikation bei Dev Fragen
- Abhängigkeiten zwischen Teams probieren zu minimieren

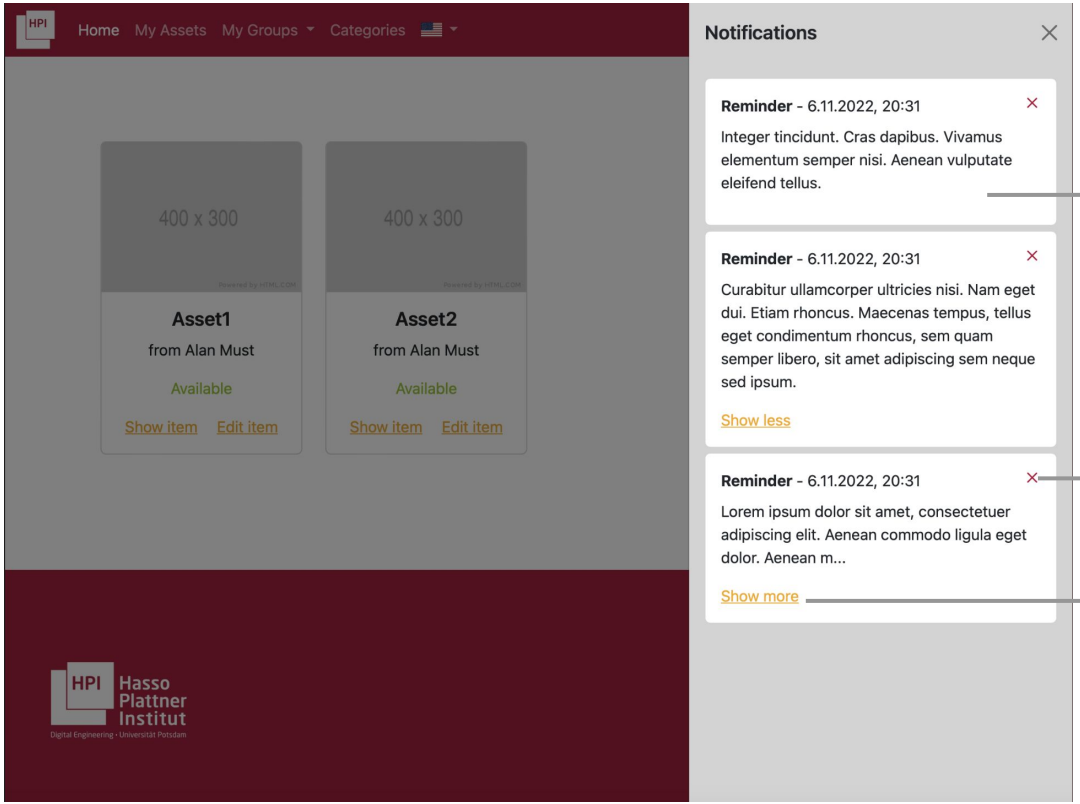


# Technology

## Notification Inbox

# Notification Inbox



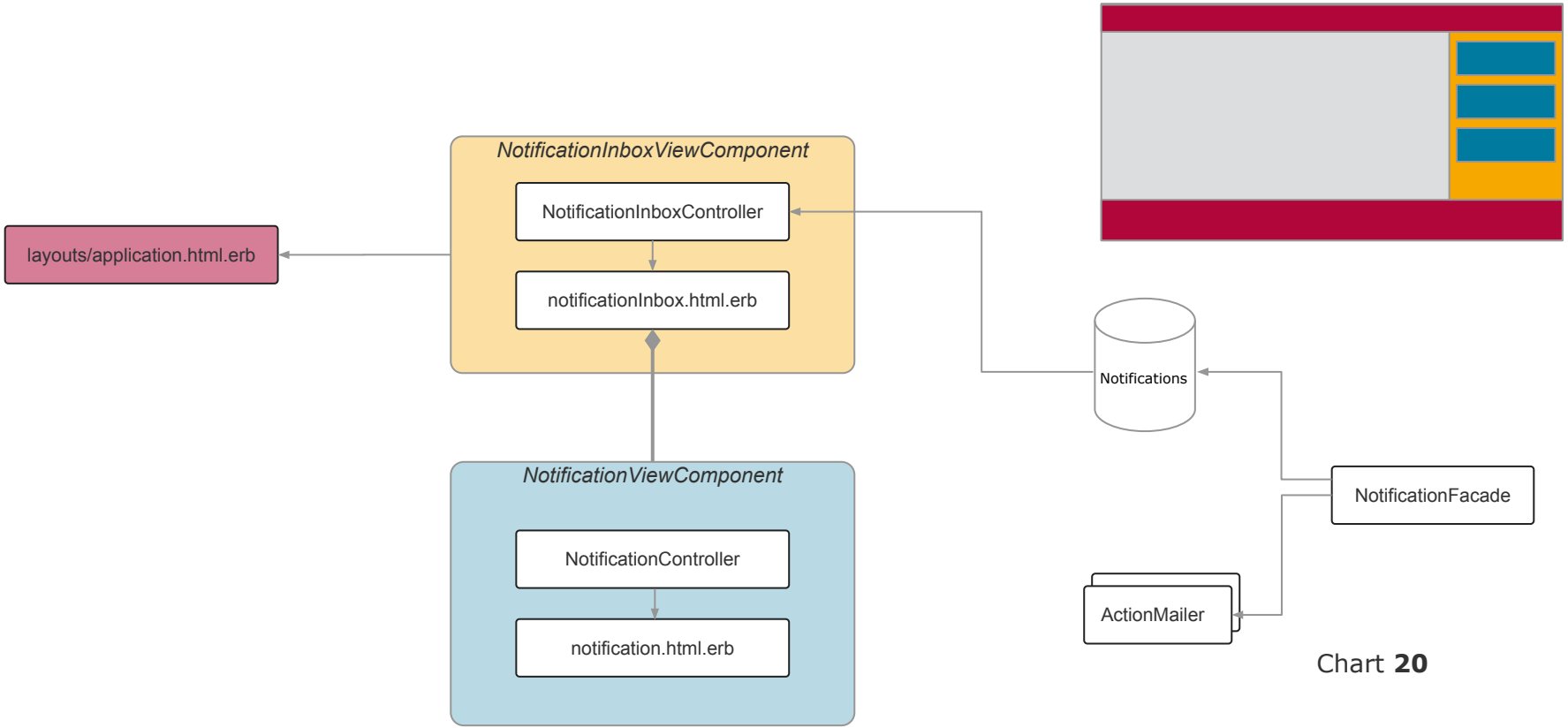




Auflistung aller versendeten Nachrichten

dauerhaftes Ausblenden / “Löschen” durch Nutzer

Ein-/Ausklappen von langen Nachrichten

# Implementierungsansatz



# NotificationInboxComponent

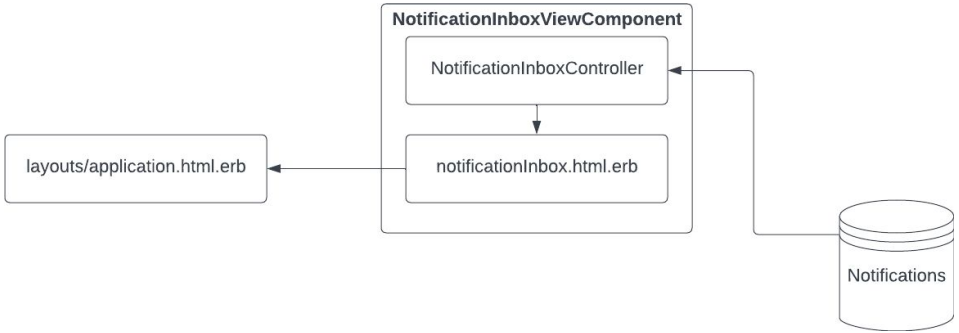
```erb
<div class="row notification-inbox">
  <div class="mx-auto">

    <% @notifications.each do |notification| %>
      <%= render(NotificationComponent.new(notification)) %>
    <% end %>

    <% if @notifications.empty? %>
      <div class="notification-inbox__empty">
        <p class="notification-inbox__empty-text">You don't have any notifications yet. 🤷</p>
      </div>
    <% end %>
  </div>
</div>
```

Card-Rendering je Notification in Sub-Component

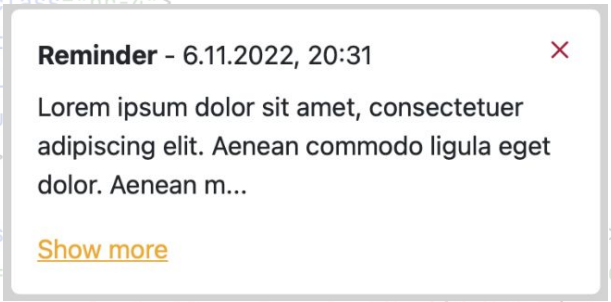
Ausnahmebehandlung

Einbinden in Base-Template:

```erb
<div class="offcanvas-body">
  <%= render(NotificationInboxComponent.new(current_user)) %>
</div>
```

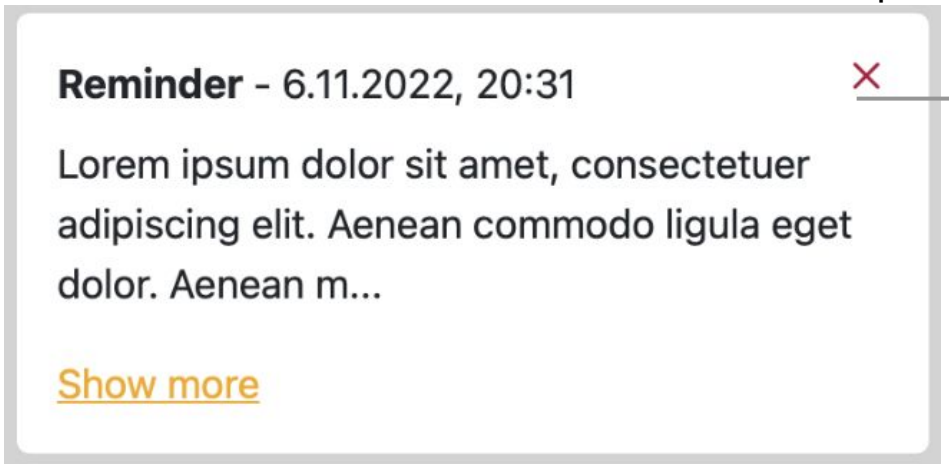


# NotificationComponent



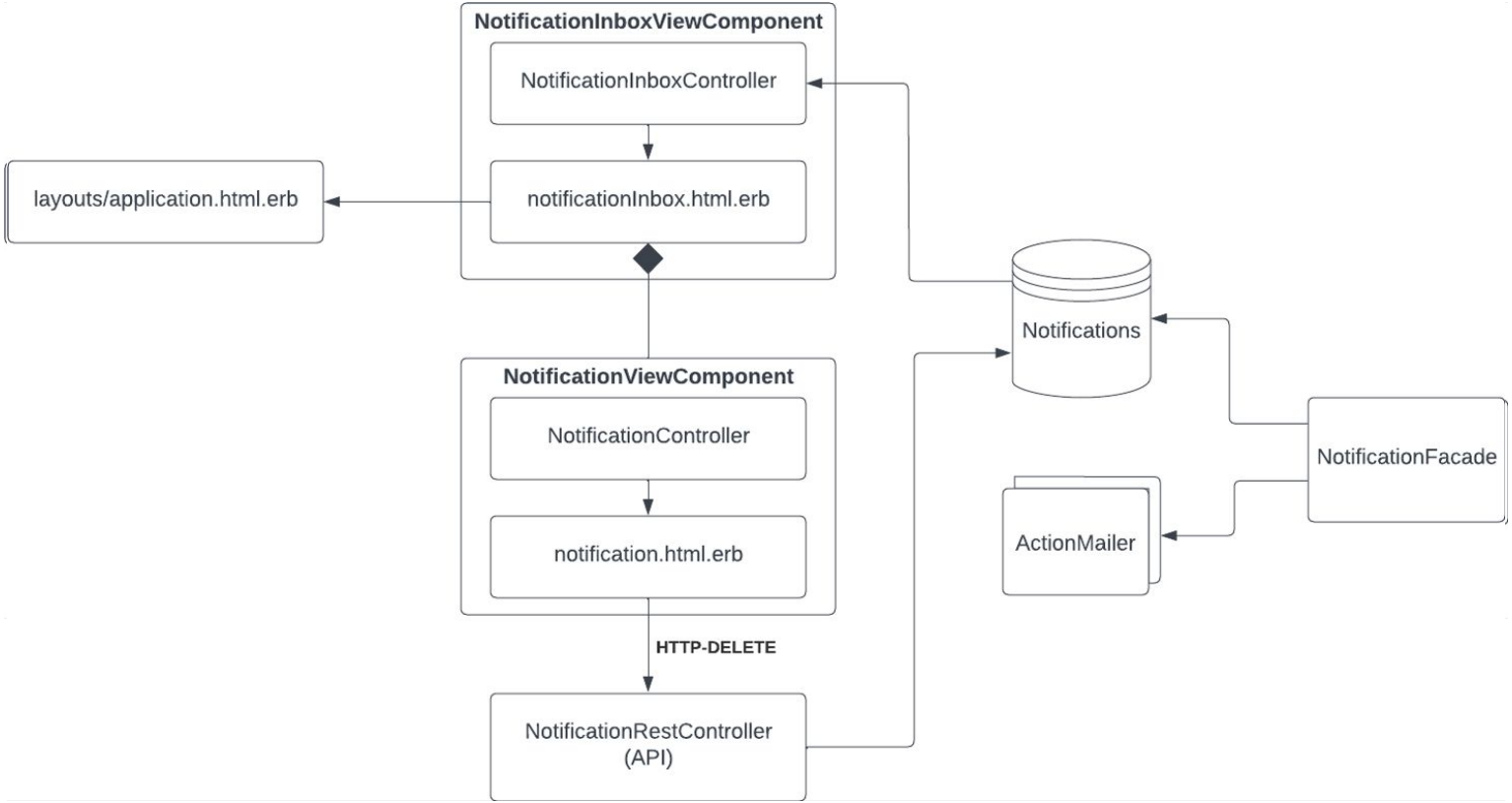
# One last thing…

permanentes Ausblenden
von Nachrichten




Bei Button-Clicks **HTTP-DELETE Request im Hintergrund** absenden und **NotificationComponent** (bei Status 2xx) **ausblenden** (per DOM-Zugriff)

# Erweiterte Architektur



# Nutzung der API und DOM-Manipulation (Client)

```
function deleteNotification(notificationId) {
    fetch(`/notifications/${notificationId}`, {method: 'DELETE', redirect: 'follow'})
        .then(result => {
            if (result.status >= 200 && result.status < 300) {
                let msgContainer = document.getElementById(`notification-${notificationId}`);
                msgContainer.remove()
            } else {
                alert("Failed to delete notification.")
            }
        })
        .catch(error => console.log('error', error));
}
```

API-Request

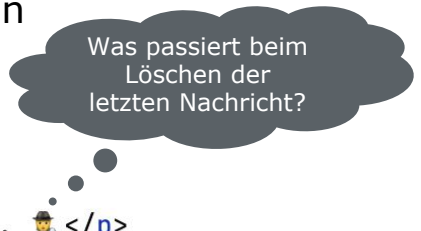
DOM-Removal

# Schwachstellen

- **Verteilte** und **implizite** Definition der UI-Dynamik, u.a.:
  - Rendering via *ViewComponent*
  - Collapse/Expand via *CSS*
  - Ausblenden von gelöschten Elementen via *JS*

- **Bug-Anfälligkeit**, wenn Server-Side-Rendering mit *DOM*-Manipulationen auf dem Client gepaart wird:

```
<% if @notifications.empty? %>
  <div class="notification-inbox__empty">
    <p class="notification-inbox__empty-text">You don't have any notifications yet. 🤷</p>
  </div>
<% end %>
```




- **Styling von Components** nur über *CSS* und *HTML* schwierig/unintuitiv, bspw. inhaltslängenabhängiges Anzeigen des “Show more”-Buttons

# Lösungsansatz: **Client-Side-Rendering** mittels Vue

Viele Funktionalitäten des Features betreffen **dynamisches Rendering auf dem Client** ohne neue Requests, bspw:

- Anzeige des ausgeklappten / eingeklappten Nachrichteninhalts
- Ausblenden von gelöschten Nachrichten
- Anzeige von “Show more” / “Show less”
- Anzeigen des “Empty Inbox”-Hinweises

Wir **verlagern** das **Rendering** für Components dieses Features **auf den Client** und kommunizieren mit dem Rails-Backend über **API-Endpunkte**

**Vue.js**

“An *approachable*, *performant* and *versatile* framework for building web user interfaces.”

# Warum Vue.js?

## Vue.js

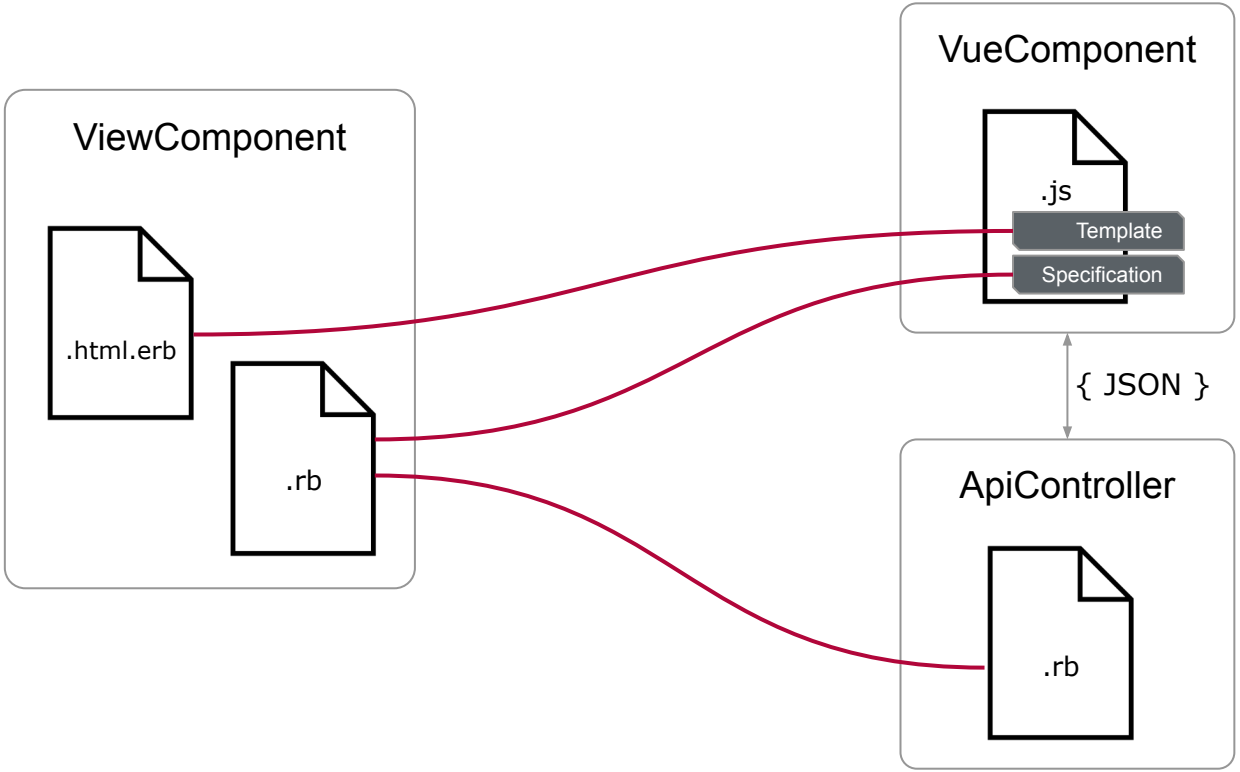
"An ***approachable***, *performant* and ***versatile*** framework for building web user interfaces."

***approachable*** → Builds on top of standard HTML, CSS and JavaScript with intuitive API and world-class documentation.

***versatile*** → A rich, incrementally adoptable ecosystem that scales between a library and a full-featured framework.

https://vuejs.org/

# Architekturvergleich: SSR vs. CSR



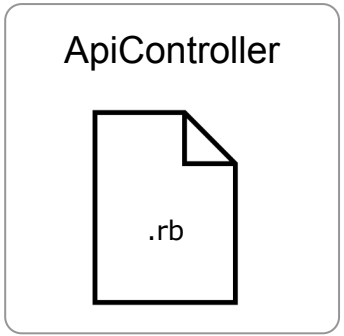
# Backend: Notifications-API

**GET**

```ruby
class NotificationsController < ActionController::API
  include ActionController::MimeResponds

  def all
    visible_notifications = Notification.where(
      display: true,
      user_id: current_user.id
    ).order(:sent).last( limit 30)

    respond_to do |format|
      format.json { render json: visible_notifications }
    end
  end
```

**DELETE**

```ruby
  def destroy
    notification = Notification.find(params["id"])

    # Allow modifications only on notifications a user owns
    return unless notification.user_id == current_user.id

    # ...
    notification.display = false
    notification.save
  end
end
```



# Frontend: NotificationComponent

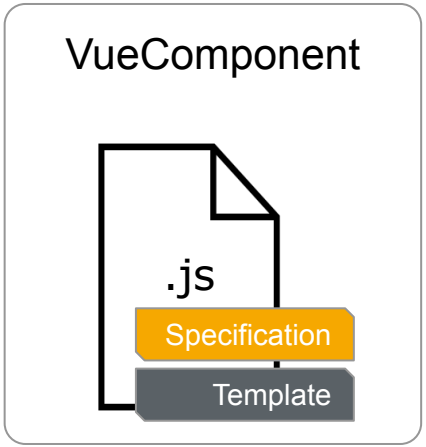
Parametrisierung

Zustand

Methoden

Computed Properties

```
export const Notification = {
    props: {
        notification: {id: Number...}
    },
    data: () => {
        return {
            collapsed: true
        }
    },
    methods: {
        remove() {...}
    },
    computed: {
        message() {...},
        shortMessage() {...},
        longMessage() {...},
        expandable() {...},
    }
}
```

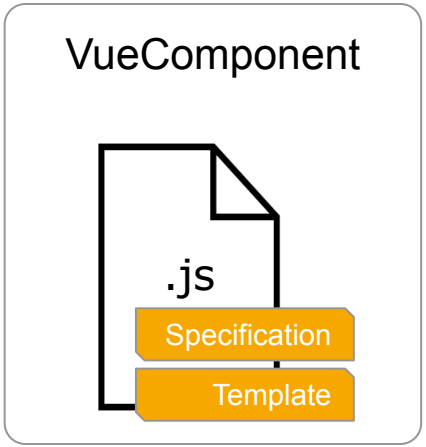


# Frontend: NotificationComponent

*Computed Properties*

```
<p>{{ message }}</p>
<a class="card-link" @click="collapsed = !collapsed" v-if="expandable">
    {{ collapsed ? "Show more" : "Show less" }}
</a>
```

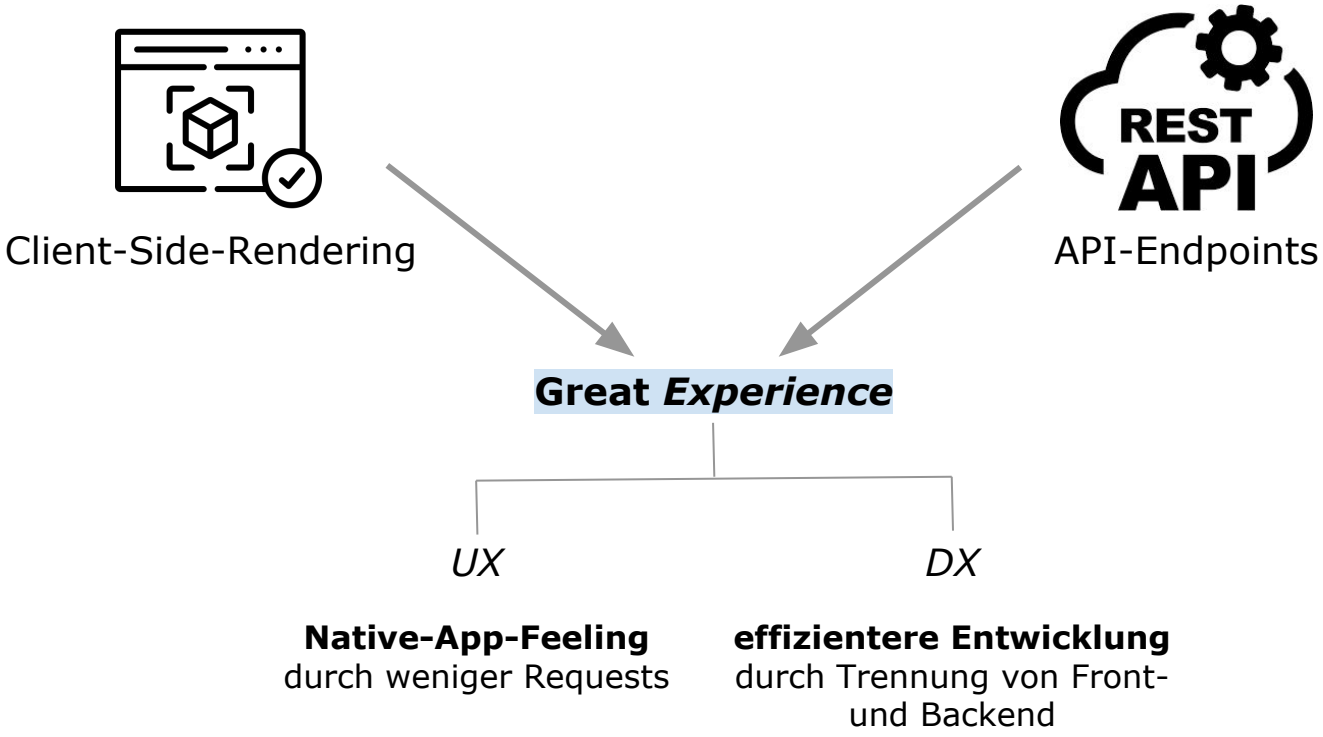
```
message() {
    if (this.collapsed) {
        return this.shortMessage + (this.expandable ? "..." : "");
    }
    return this.notification.message;
},
shortMessage() {
    return this.notification.message.slice(0, MSG_CUTOFF);
},
expandable() {
    return this.shortMessage !== this.notification.message;
}
```



# Frontend: Notification-Inbox

```
<div class="row notification-inbox">
  <div class="mx-auto">
    <div id="notifications">
      <Notification
            v-for="notification in notifications"
            @removed="removeNotification(notification.id)"
            :key="notification.id"
            :notification="notification"
      />
      <div class="notification-inbox__empty" v-if="notifications.length === 0">
        <p class="notification-inbox__empty-text">You don't have any notifications yet. 🤷</p>
      </div>
    </div>
  </div>
</div>
```

# Fazit

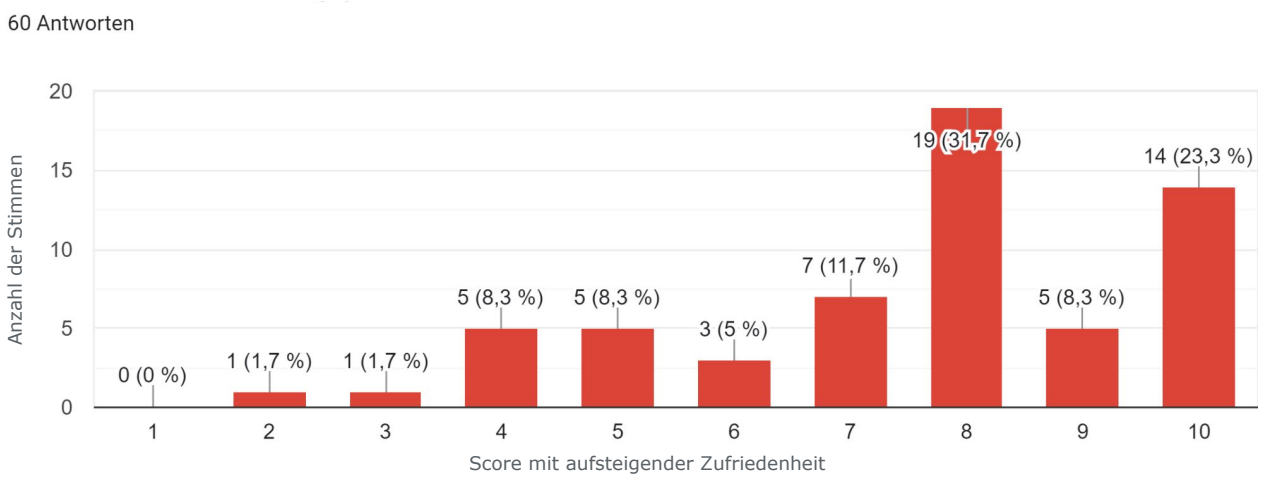

Client-Side-Rendering
REST API
API-Endpoints
Great *Experience*
*UX*
*DX*
**Native-App-Feeling**
durch weniger Requests
**effizientere Entwicklung**
durch Trennung von Front- und Backend

# Statistiken & Feedback

# Daten pro Developer

Wie zufrieden sind die Devs mit ihrem **eigenen Zeitmanagement** gewesen?

bezüglich humaner Uhrzeiten und Arbeitslänge

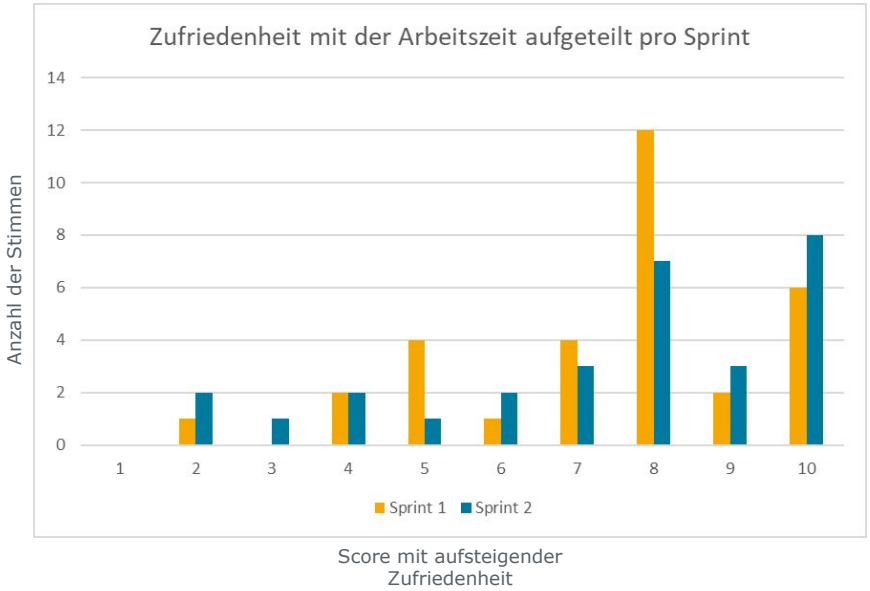



\+ Mehr als **50%** sind **ziemlich zufrieden** mit ihren Arbeitszeiten

\- es gibt wenige, aber sehr starke **Ausreißer**

# Daten pro Developer

Wurde diese **Zufriedenheit im 2. Sprint** besser?




\+ Es gibt **mehr** Devs, die **9 oder 10** Punkte verteilt haben

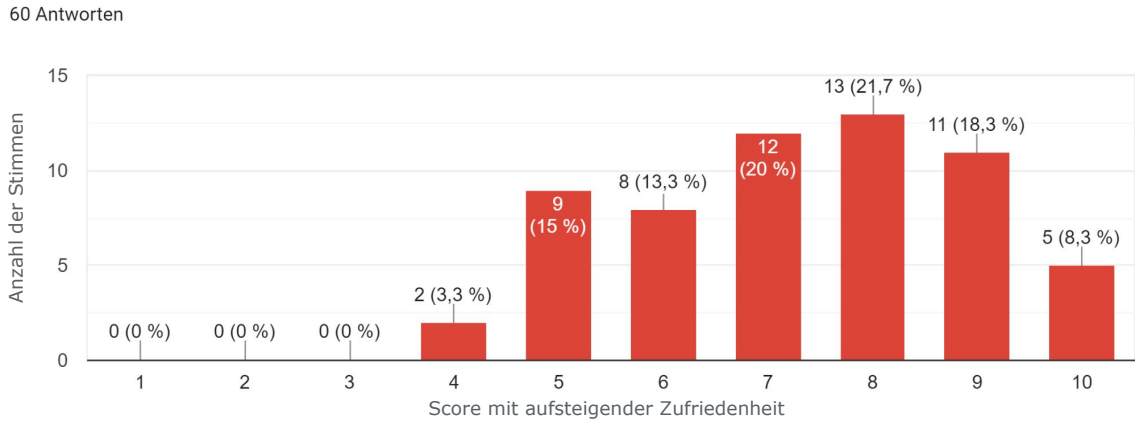
• Keine Sichtbare Verbesserung zwischen den Sprints

\- **mehr 2 & 3er** Wertungen in Sprint 2

# Daten pro Developer

Wie zufrieden sind die Devs mit ihrer **eigenen Code-Qualität**?

Testing, Code Smells, Modularität, …

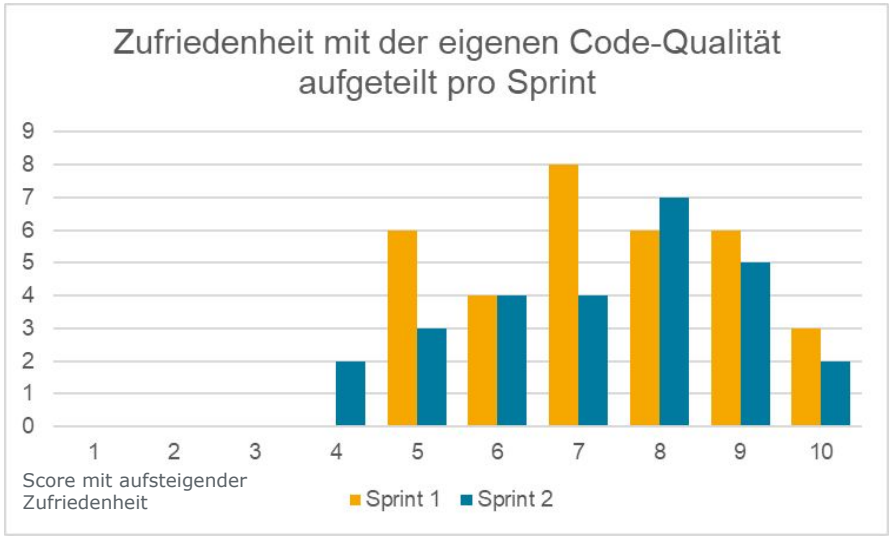



\+ ca. **48,3%** bewerten ihren eigenen Code mit **8 oder höher**

\- man sollte eigentlich davon ausgehen, dass **jeder** Developer **bestmöglichen Code** produziert

# Daten pro Developer

Wie hat sich die **Zufriedenheit über die Sprints** entwickelt?




Ø Sprint 1: 7,3
Ø Sprint 2: 7,26

- **keine Veränderung** in der allgemeinen Zufriedenheit
- grundsätzlich scheint jeder mit seinem eigenen Code **nicht vollends zufrieden** zu sein

# Daten pro Scrum Team

Wie zufrieden waren die Teams im Sprint mit der **Teamarbeit** in ihrem Team?

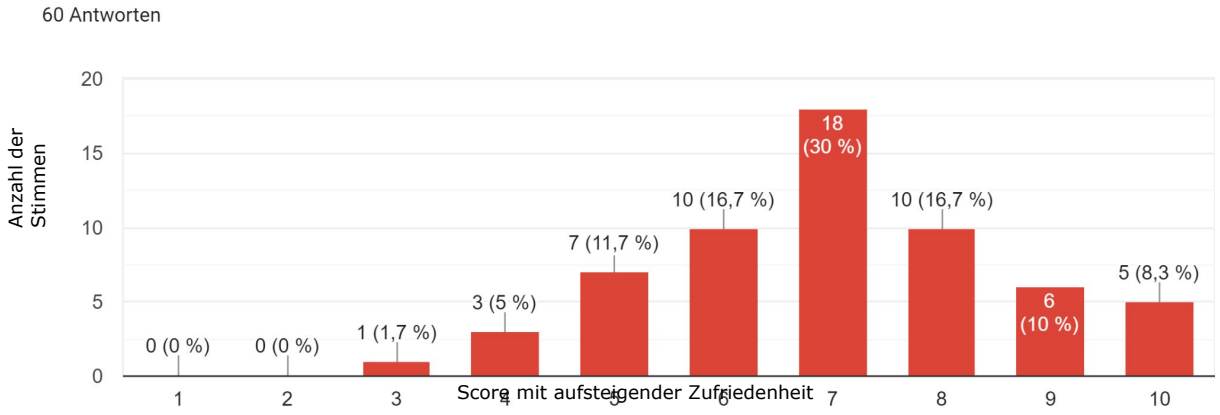

\+ Durchschnittlich eine **hohe Zufriedenheit**

\+ Bis auf ein Team hat die **Zufriedenheit** über die Sprints **zugenommen**

\- es gab bei einem Team eine Abnahme der Zufriedenheit, der Auslöser hierfür wurde identifiziert

# Daten für Team Rot

Wie ist die Zufriedenheit mit der **allgemeinen Organisation in Team Rot**?




\+ im Schnitt gibt es eine **Zufriedenheit von ca. 7**

\- es gibt jedoch **einige Ausreißer** nach unten

# Daten für Team Rot

Wie ist die Zufriedenheit mit der **allgemeinen Organisation in Team Rot**?

**Was lief gut?**

+ **Gute Übersicht** durch wöchentliche PM & PO Treffen
+ übersichtliches, **gepflegtes GitHub**
+ Teamaufteilung
+ sichtbarer **funktionaler Fortschritt**
+ spürbare **Motivation**

**Was lief nicht so gut?**

- **Verantwortung diffundiert**
- Teams als **Bubbles**
- **asynchrone Sprints**
- Verteilung der User Stories teils unklar
- **Doppelte Bearbeitung** von User Stories

**Verbesserungen**

- POs behalten Überblick, erstellen **Tickets** & weisen Devs darauf hin
- **Teamübergreifende Reviews?**
- **Bessere Absprache** der POs

# Feedback von Stakeholdern


Michael Perscheid

+ **kontinuierlicher Fortschritt**

- **spätes Deployment**
  → Upper Management hatte lange **keine Demo**


Christoph Matthies

+ PRs sinnvoll
+ **Commit Guidelines** & OIDC Setup **in Readme** dokumentiert
+ **Projects Board** übersichtlich
+ Tickets dokumentieren **Abhängigkeiten** & **Mockups**

- **PRs** mehr **zur Doku** nutzen
- Branches aufräumen
- **Deployment** umstellen
- **Kontrollfluss-Mockups in Tickets**


Ralf Teusner

+ **Kernprozess verstanden** & grundlegend **ausgeführt**
+ **weitere Ansätze** von Funktionalitäten **existieren**

- **mehr eigene Ideen** & Vorschläge einbringen
- **keine Demo**

# Lessons Learned

**Organisation während der Sprints** ist essenziell

- **Trennung** zwischen Development- und Orga-Aufgaben erleichtert die individuelle Arbeit

**Kommunikations-Overhead** muss berücksichtigt werden

**Dokumentation & Issue-Tracking** helfen bei einem großen Projekt den Überblick zu behalten

- **Draft-PRs** machen aktuelle Projektbaustellen schnell erkennbar

**Flache Hierarchien** erschweren die Aufgabenverteilung und das Treffen von Entscheidungen

# Lessons Learned (POs)

**Umfangreicher & gemeinsamer Prototype** hilft allen:

- Einheitliche Grundlage für alle Teams
- Referenz für Entwicklung und Gespräch mit Kunden

Fehlendes technisches Verständnis der Product Owner erfordert mehr **Kommunikation mit Devs**. Feedback ist sehr wertvoll.

**Zeit** mit dem Kunden ist kostbar

**Verantwortlichkeiten** müssen geklärt werden

- Aufgaben ohne Verantwortlichkeit müssen berücksichtigt werden

# Lessons Learned (SMs)

## Kommunikation

- **Team aus Teams** Gefühl herstellen ist schwer
- **Wichtige Entscheidungen teamübergreifend** früh treffen & kommunizieren
- Teamübergreifende **Meetings** durchführen
- Online Tools unterstützen

## Zeit

- Klare Anforderungen & Freiräume
- **Timeboxen** & Priorisieren
- Zeit **sichtbar & hörbar** machen
- extra Timekeeper Rolle
- Zu technische & bilaterale **Gespräche stoppen**

## Verantwortung

- **Verantwortungsdiffusion**
- Verantwortung klar verteilen
- Protokoll hält Verantwortungen fest

# Lessons Learned: Hilfreiche Tools (SMs)

**Notion:** Scrum Master Hub anlegen

**Planning Poker Online**:

https://planningpokeronline.com/

**Gute, abwechslungsreiche Retros** mit Anleitungen zusammenstellen:

https://retromat.org

**Sichtbares Timekeeping** im Meeting mit projiziertem, digitalem TimeTimer:

https://timer.designthinkingcoach.de

**Rede-Tokens**

- einige betteln nach mehr Tokens
- gut für Retro
- schlecht für Sprint Planning, durch viele verschiedene Issues

**Fragenspeicher**

- Diskussionen auslagern und auf später verschieben

**Diskussionsrunde** in Meetings einplanen

**Unterbrechen**, begründen & zeitlich oder in Kleingruppen verschieben

# Thank you!

## Interim Presentation - Team Rot

SSE 2022

Hasso-Plattner-Institut

# Roadmap: Gantt Chart


- Done
- In Progress
- Product Backlog
- Draft
- Rejected