- Willkommen zu unserer Präsentation für Audit 2

Inhalt

1. Projektablaufplan
2. Benutzer- / Stakeholder-Analyse
3. Domänenmodelle
4. Use Cases
5. Tech Stack
6. Datenbank-Modelle
7. Erste Projektrisiken
8. Artefakte Audit 3

- Wir gehen auf einige der Inhalte ein, die wir im vorherigen Audit nicht ausführlich genug behandelt haben (Projektablaufplan, Benutzer- / Stakeholder-Analyse, Domänenmodelle, Use Cases)
- Für diesen Audit relevante Themen (Tech Stack, ERM-, eERM- und Relationenmodell, erste Projektrisiken)

Projektablaufplan

Aufgabe	Deadline	Status
Stakeholder-Analyse	20.11.23	✅
Domänenmodelle	27.11.23	✅
ER-Modell	20.11.23	✅
Tech-Stack (Begründung)	20.11.23	✅
Erste Projektrisiken	20.11.23	✅
Projektablaufplan	27.11.23	✅
Use Cases	30.11.23	✅

Projektablaufplan

eER-Modell	04.12.23	✅
Relationenmodell	04.12.23	✅
Präsentationen	7.12.23	☑
Kommunikation / Interaktion	7.12.23	✅
Design	01.01.24	☑
Proof of Concept	15.01.24	☑
Poster 1. Version	20.01.24	
Rapid Prototyp	22.01.24	

- Hier sehen wir den Projektablaufplan
- Sowohl eine Deadline als auch ob diese Aufgabe erledigt, wurde
- Die Grauen Hacken sind hierbei für Aufgaben, die angefangen wurden und nicht fertig sind
- Die Aufteilung passt sich im Verlauf des Projektes an, da wir die Aufgaben häufig gemeinsam erledigen, bzw. sich die Zuteilung je nach Verfügbarkeit der Zeit der Teammitglieder verändert

Benutzeranalyse

Stakeholdergruppe	Definition	Relevante Themen	Zielsetzungen
Entwickler	An dem Projekt beteiligte Programmierer.	Änderungen im Verhalten und im Design der Website.	Transparente Kommunikation, übersichtlicher Prozess
Designer	Diejenigen, die für die Gestaltung von UI verantwortlich sind.	Änderungen des Verhaltens der UI und des Kontextes.	Transparente Kommunikation, übersichtlicher Prozess
Marketing-Manager	Manager, die das Projekt nach außen präsentieren.	Neuerungen in Design und Verhalten o. Arbeitsschritte sowie Fortschritt.	Maximale Aufmerksamkeit, Maximum-Prinzip, Informationen über Änderungen

Benutzeranalyse

	Programmierer.	Design der Website.	übersichtlicher Prozess
Designer	Diejenigen, die für die Gestaltung von UI verantwortlich sind.	Änderungen des Verhaltens der UI und des Kontextes.	Transparente Kommunikation, übersichtlicher Prozess
Marketing-Manager	Manager, die das Projekt nach außen präsentieren.	Neuerungen in Design und Verhalten o. Arbeitsschritte sowie Fortschritt.	Maximale Aufmerksamkeit, Maximum-Prinzip, Informationen über Änderungen
Projekt-Manager	Manager, die das Projekt verwalten.	Fortschritt und Intention.	Übersicht des Projektstandes, Informationen über

- Die Benutzeranalyse umfasste Benutzergruppen, die eine Wirkung auf das System haben
- Dabei sind die Gruppen durch eine Definition spezifiziert und mit einem Ziel ausgestattet
- **Z.B.:** Designer sind für die Gestaltung einer Anwendung verantwortlich und haben an dem System ein Interesse, um Benachrichtigungen zu (Verhaltens-) Änderungen durch dieses zu bekommen.

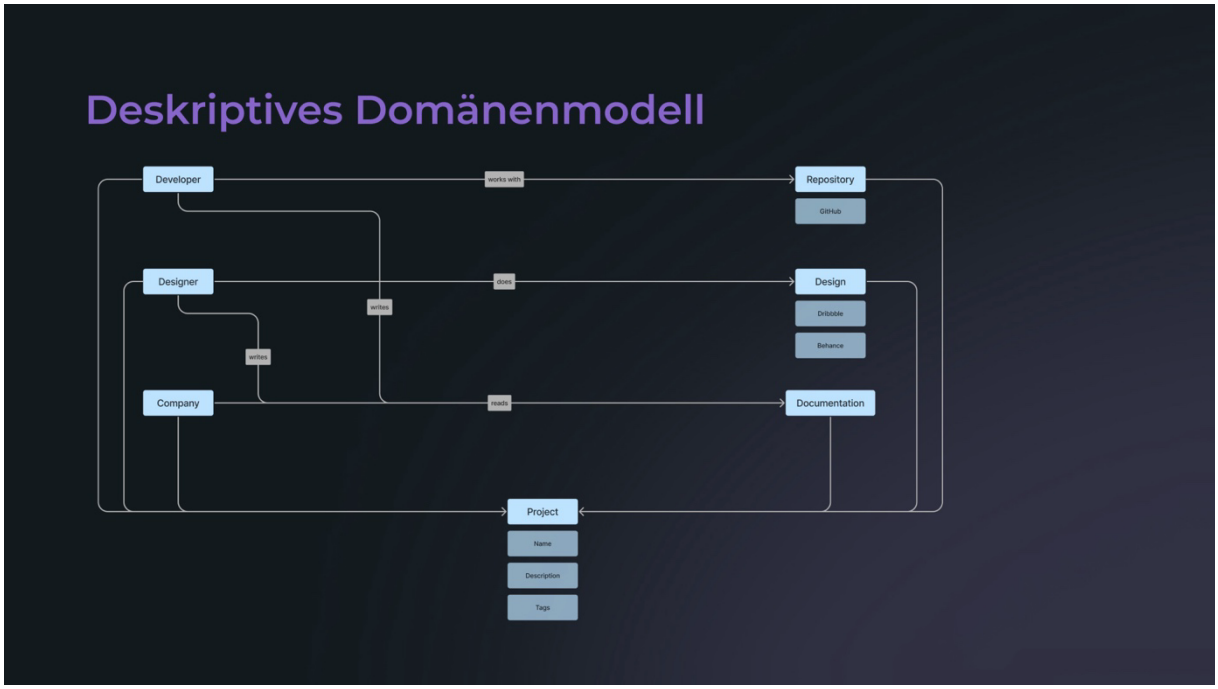
Stakeholder-Analyse

Bezeichnung	Relation	Objektbereich	Erfordernisse / Erwartungen
Entwickler	Anteil	Kontributionen / Feedback	Transparenz der Kontributionen, Mehr aktives Feedback
Designer	Anteil	Kontributionen / Feedback	Transparenz der Kontributionen, Mehr aktives Feedback
Marketing-Manager	Interesse	Guides / Verlauf	Benachrichtigungen über Änderungen
Projekt-Manager	Interesse / Anrecht /	Verlauf / Feedback /	Benachrichtigungen

Stakeholder-Analyse

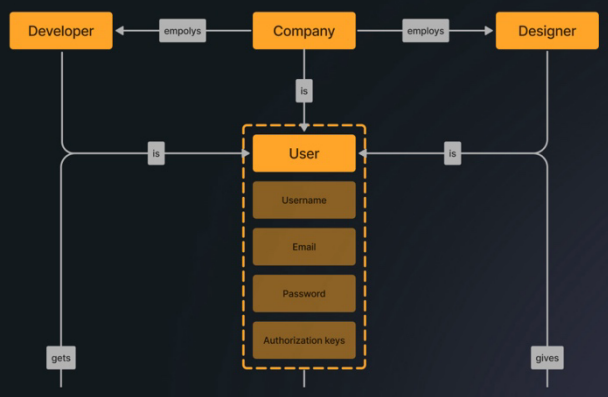
		Feedback	Kontributionen, Mehr aktives Feedback
Marketing-Manager	Interesse	Guides / Verlauf	Benachrichtigungen über Änderungen
Projekt-Manager	Interesse / Anrecht / Anspruch	Verlauf / Feedback / Updates	Benachrichtigungen über Änderungen, Transparenz des Standes
Product-Owner	Anrecht / Anspruch	Stand / Updates	Benachrichtigungen über Änderungen, Transparenz des Fortschritts

- Die Stakeholder-Analyse befasst sich mit Benutzergruppen und deren Relation zum System
- **Unsere Stakeholder sind:**
  - Entwickler
  - Designer
  - Marketing-Manager
  - Projekt-Manager
  - Product-Owner
- Diese haben unterschiedliche Relationen zum System und Anforderungen an das System
- **Z.B.:** Entwickler haben ein Anteil am System durch Kontributionen und erwarten Transparenz bei Kontributionen, unteranderem durch Feedback

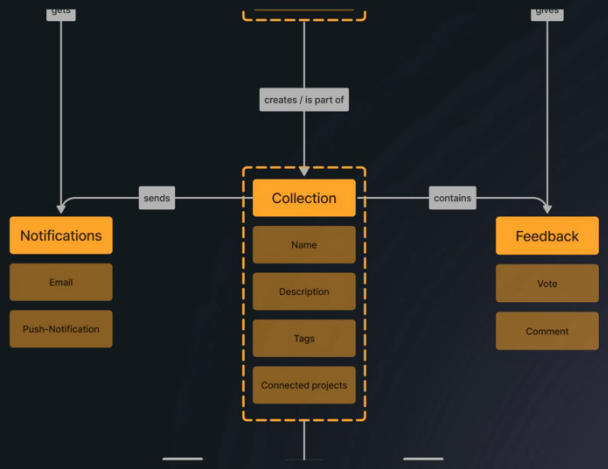


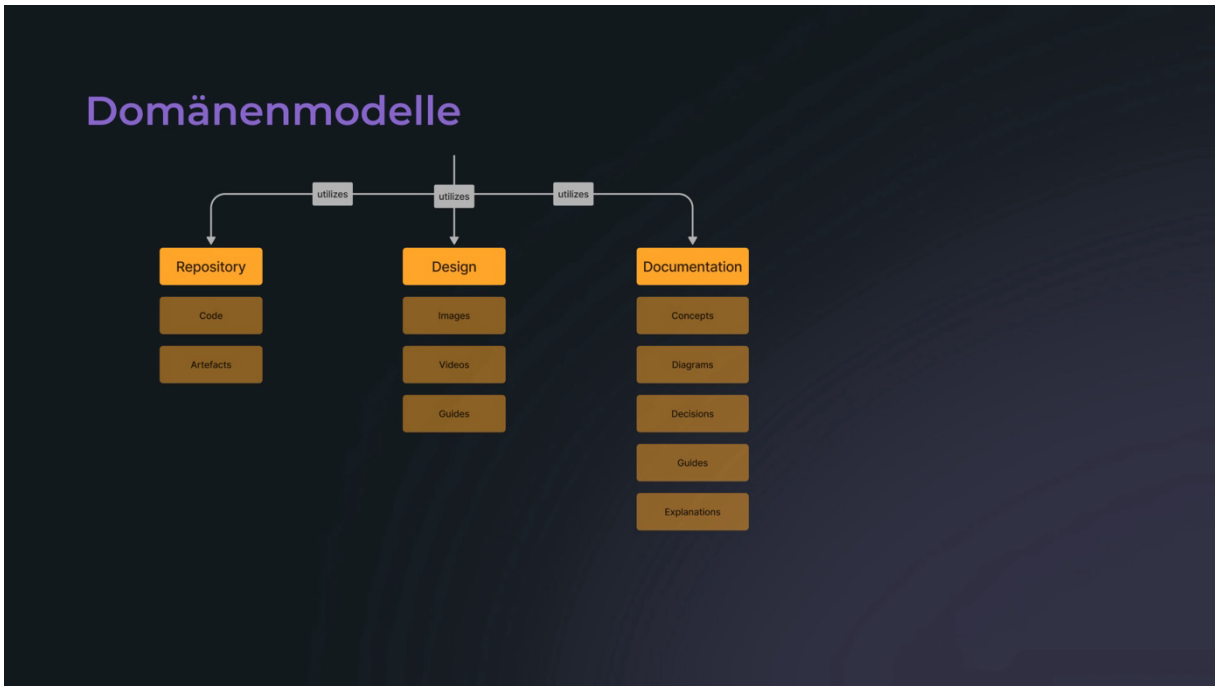
- **Arbeitsbereiche**, wie **Design**, **Entwicklung** und **Dokumentation** sind immer voneinander getrennt, sodass verschiedene Plattformen genutzt werden müssen
- Dadurch können die Beteiligten nicht immer Einblick auf die Arbeiten der anderen Beteiligten zugreifen
- Anhand des deskriptiven Domänenmodells kann man diese intransparenten Bereiche gut erkennen

Präskriptives Domänenmodell

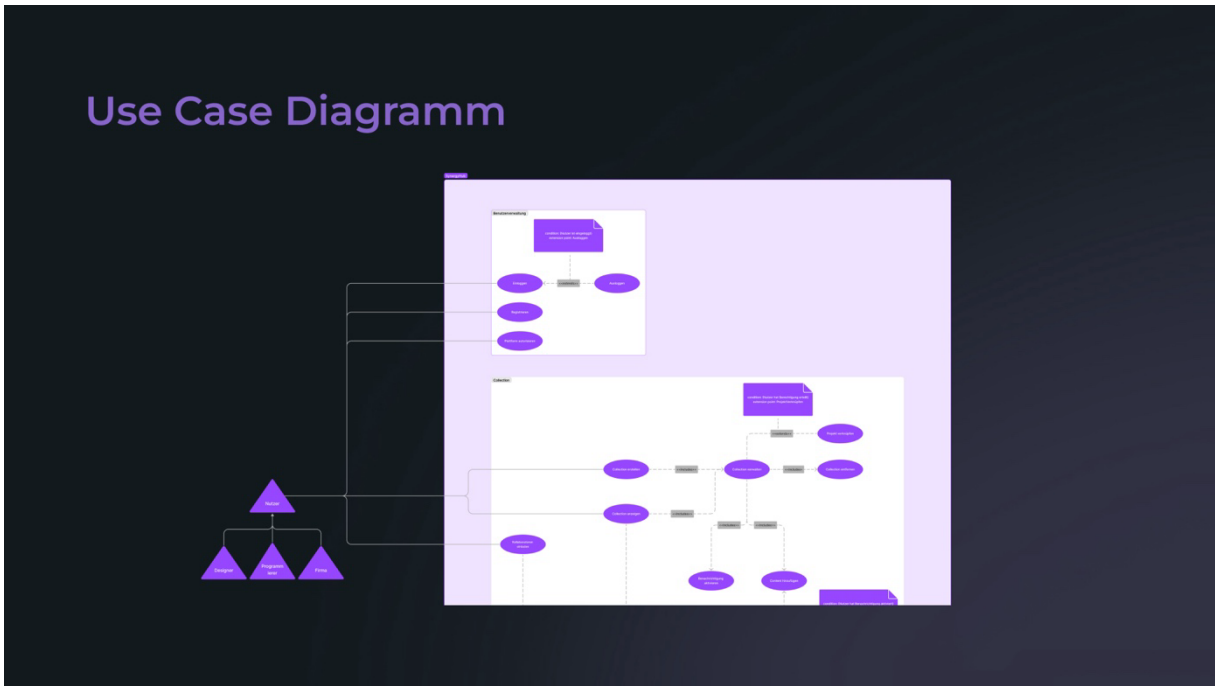


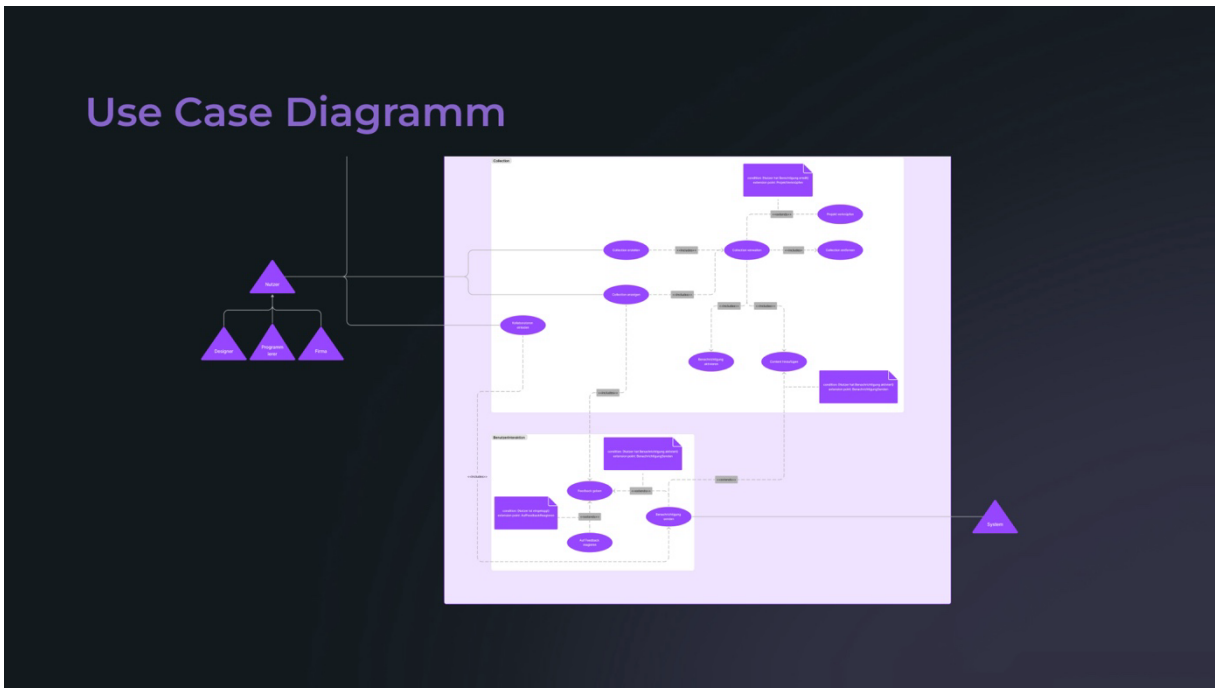
Domänenmodelle





- Über Collections sollen diese Bereiche untereinander transparenter gemacht werden
- Außerdem soll es einfache Möglichkeiten geben, Feedback zwischen den einzelnen Benutzern auszutauschen
- Um ein Mittel zu bieten, immer auf dem aktuellen Stand zu bleiben, sollen außerdem Benachrichtigungen eingeführt werden, welche plattformübergreifend funktionieren sollen





- **<u>System lässt sich in drei Anwendungsbereiche unterteilen:</u>**
  - Benutzerverwaltung
  - Collection
  - Benutzerinteraktion
- Die einzelnen Use Cases sind dem Anwendungsfalldiagramm zu entnehmen

Textuelle Use Cases

Registrieren

use case Registrieren

actors Nutzer

precondition Der Nutzer hat keinen Account und möchte sich registrieren

main flow

1. Der Nutzer geht zur Seite für die Registrierung.
2. Der Nutzer gibt seine Informationen ein, wie Vorname, Nachname, E-Mail-Adresse und ein Passwort seiner Wahl.
3. Das System prüft, ob die Eingegebenen Daten okay sind, z.B. ob die E-Mail-Adresse korrekt ist und das Passwort sicher genug ist.
4. Wenn die Daten in Ordnung sind, erstellt das System einen neuen Account für den Nutzer.
5. Der Nutzer bekommt eine E-Mail zur Bestätigung, um sicherzustellen, dass die Registrierung echt ist.
6. Nach erfolgreicher Bestätigung wird der Nutzer zur Anmeldeseite weitergeleitet.

postcondition Account erstellt

Ein neuer Nutzer-Account wurde erstellt.

end Registrieren

Einloggen

use case Einloggen

actors Nutzer

precondition Nutzer hat einen Account und ist nicht angemeldet

main flow

1. Der Nutzer gibt seine E-Mail und sein Passwort ein.
2. Das System prüft die eingegebenen Daten auf Korrektheit.
3. Der Nutzer ist angemeldet und wird auf die Hauptseite geleitet.

postcondition

Erfolgreiche Anmeldung des Nutzers

exceptional flow Fehlerhafte Daten

2a. Das System stellt eine Inkonsistenz der Daten fest.

3. Der Nutzer wird auf den Fehler hingewiesen.

postcondition Fehlerhafte Anmeldedaten

Der Nutzer wurde nicht angemeldet und wurde darüber benachrichtigt.

exceptional flow Kein Nutzer mit diesem Konto

2a. Das System stellt eine Inkonsistenz der Daten fest.

3. Der Nutzer wird auf den Fehler hingewiesen.

end Einloggen

Textuelle Use Cases

Erstellung von Collection

use case Erstellung von Collection

actors Nutzer

precondition Der Benutzer hat ein Konto und ist angemeldet

main flow

1. Der Benutzer navigiert zur Seite für die Erstellung einer Sammlung.
2. Der Benutzer gibt einen Namen und optional eine Beschreibung für die Sammlung ein.
3. Das System überprüft die eingegebenen Informationen auf Gültigkeit.
4. Bei gültigen Informationen erstellt das System eine neue Sammlung für den Benutzer.
5. Der Benutzer wird zur Seite der erstellten Sammlung weitergeleitet.

postcondition Collection erstellt

Eine neue Collection wurde erstellt.

end Erstellung von Collection

Verwaltung von Collection

use case Verwaltung von Collection

actors Administrator

precondition Der Administrator ist angemeldet und hat Zugriff auf die Sammlungsverwaltung

main flow

1. Der Administrator wählt die Option "Sammlung verwalten" aus.
2. Das System zeigt eine Liste aller vorhandenen Elemente in der Sammlung an.
3. Der Administrator kann folgende Aktionen ausführen:
   - Element hinzufügen
   - Element entfernen
   - Element bearbeiten
   - Sammlung durchsuchen
4. Das System aktualisiert die Sammlung basierend auf den durchgeführten Aktionen.

postcondition Collection aktualisiert

Die Collection wurde aktualisiert.

end Verwaltung von Collection

Textuelle Use Cases

Autorisierung von anderer Plattformen

use case Autorisierung auf anderer Plattform

actors Benutzer, externe Plattform

precondition Der Benutzer ist angemeldet und möchte die Anwendung mit einer externen Plattform autorisieren

main flow

1. Der Benutzer wählt die Option "Externe Plattform autorisieren" in den Anwendungseinstellungen.
2. Das System leitet den Benutzer zur Anmeldeseite der externen Plattform weiter.
3. Der Benutzer gibt seine Anmeldeinformationen für die externe Plattform ein.
4. Die externe Plattform überprüft die Anmeldeinformationen.
5. Wenn die Anmeldung erfolgreich ist, fordert die externe Plattform den Benutzer auf, die Autorisierung für die Anwendung zu erteilen.
6. Der Benutzer stimmt der Autorisierung zu.
7. Die externe Plattform generiert einen Autorisierungscode und sendet diesen an die Anwendung.
8. Das System verwendet den Autorisierungscode, um einen Zugriffstoken von der externen Plattform zu erhalten.
9. Das System speichert den erhaltenen Zugriffstoken für zukünftige Interaktionen mit der externen Plattform.

postcondition Benutzer hat andere Plattformen Autorisiert

Der Benutzer hat die Berechtigung für die Nutzung einer anderen Plattform erteilt.

end Autorisierung von anderen Plattformen

Feedback geben

use case Feedback geben

actors Benutzer, System

precondition Der Benutzer hat die Möglichkeit, Feedback zu geben, und das System unterstützt die Erfassung von Rückmeldungen

main flow

1. Der Benutzer navigiert zu der Feedback-Seite oder -Funktion in der Anwendung.
2. Das System präsentiert dem Benutzer ein Formular oder eine Schnittstelle zur Eingabe von Feedback.
3. Der Benutzer gibt sein Feedback ein, das Informationen zu positiven Erfahrungen, Verbesserungsvorschlägen oder Problemen enthalten kann.
4. Das System überprüft und speichert das eingegebene Feedback.

postcondition Benutzer hat ein Feedback gegeben

Der Benutzer hat neues Feedback hinterlassen.

end Feedback geben

Textuelle Use Cases

Benachrichtigung für Collection aktivieren

use case Benachrichtigung für Collection aktivieren

actors Nutzer,System

precondition Der Benutzer ist angemeldet und hat Zugriff auf die Einstellungen oder Verwaltungsfunktionen der Sammlung

main flow

1. Der Benutzer navigiert zu den Einstellungen oder der Verwaltungsseite der Sammlung.
2. Das System zeigt die Option "Benachrichtigungen aktivieren" für die ausgewählte Sammlung an.
3. Der Benutzer wählt die Option "Benachrichtigungen aktivieren" aus.
4. Das System fragt nach den Präferenzen des Benutzers für Benachrichtigungen (z. B. E-Mail-Benachrichtigungen, Push-Benachrichtigungen).
5. Der Benutzer wählt seine bevorzugten Benachrichtigungseinstellungen aus.
6. Das System speichert die ausgewählten Einstellungen für die Benachrichtigung der Sammlung.

postcondition Der Nutzer hat die Benachrichtigung für eine Collection aktiviert

Die Einstellungen des Nutzers wurden erfolgreich aktualisiert

end Benachrichtigung für Collection aktivieren

Ausloggen

use case Ausloggen

actors Nutzer

precondition Nutzer ist aktuell angemeldet

main flow

1. Der Nutzer wählt die Option zum Ausloggen aus.
2. Das System führt eine Abmeldung durch, beendet die aktuelle Sitzung und leitet den Nutzer auf die Anmeldeseite.

postcondition Erfolgreich ausgeloggt

Der Nutzer befindet sich auf der Anmeldeseite oder einer Bestätigungsseite.

end Ausloggen

Textuelle Use Cases

Projekt verknüpfen

use case Projekt verknüpfen

actors Nutzer, System

precondition Nutzer ist angemeldet und auf der Projektverwaltungsseite

mainflow

1. Der Nutzer wählt die Option, ein neues Projekt zu verknüpfen oder ein bestehendes Projekt auszuwählen.
2. Das System zeigt dem Nutzer eine Liste der verfügbaren Projekte oder ermöglicht ihm die Erstellung eines neuen Projekts.
3. Der Nutzer wählt das gewünschte Projekt aus oder erstellt ein neues Projekt und bestätigt die Auswahl.
4. Das System speichert die Verknüpfung zum ausgewählten Projekt und aktualisiert die Projektverwaltungsseite.

precondition Erfolgreiche Verknüpfung des Projekts

Das ausgewählte Projekt ist nun mit dem aktuellen Arbeitskontext verknüpft.

end Projekt verknüpfen

Collection entfernen

use case Collection entfernen

actors Nutzer, System

precondition Nutzer ist angemeldet und auf der Seite zur Verwaltung von Sammlungen (Collections)

main flow

1. Der Nutzer wählt die Option, eine bestimmte Collection zu entfernen.
2. Das System zeigt dem Nutzer eine Bestätigungsabfrage oder ein Dialogfeld, um sicherzustellen, dass er die ausgewählte Collection wirklich entfernen möchte.
3. Der Nutzer bestätigt die Aktion durch Klicken auf "Entfernen" oder eine ähnliche Bestätigungsoption.
4. Das System entfernt die ausgewählte Collection und aktualisiert die Ansicht der verbleibenden Collections.

postcondition Nutzer hat eine Collection entfernt

Die Collection wurde erfolgreich entfernt.

end Collection entfernen

Textuelle Use Cases

Collection anzeigen

use case Collection anzeigen

actors Nutzer, System

precondition Nutzer ist angemeldet und auf der Hauptseite der Anwendung oder auf der Seite zur Verwaltung von Sammlungen

main flow

1. Der Nutzer wählt die Option, eine bestimmte Collection anzuzeigen.
2. Das System zeigt dem Nutzer die ausgewählte Collection mit ihren Inhalten und relevanten Details an.

postcondition dem Nutzer wird eine Collection angezeigt

Das System präsentiert die Inhalte der Collection.

end Collection anzeigen

Kollaborateuren einladen

use case Kollaborateuren einladen

actors Nutzer, Kollaborateuren, System

precondition Nutzer ist angemeldet und befindet sich auf der Seite des Projekts oder der Zusammenarbeit, für die Kollaborateuren eingeladen werden sollen

main flow

1. Der Nutzer wählt die Option, Kollaboratoren einzuladen.
2. Das System zeigt dem Nutzer ein Formular oder ein Dialogfeld an, in dem er die E-Mail-Adressen der Kollaboratoren eingeben kann.
3. Der Nutzer gibt die E-Mail-Adressen der Kollaboratoren ein und wählt die entsprechenden Berechtigungen oder Rollen für jeden Kollaborator aus.
4. Der Nutzer bestätigt die Einladung, und das System sendet Einladungs-E-Mails an die angegebenen E-Mail-Adressen.
5. Die eingeladenen Kollaborateuren erhalten die Einladungs-E-Mails mit einem Link zur Annahme der Einladung.

postcondition Der Benutzer hat eine Einladung verschickt

Ein neuer Kollaborateur wurde hinzugefügt

end Kollaborateuren einladen

Textuelle Use Cases

Content hinzufügen

use case Content hinzufügen

actors Nutzer, System

precondition Nutzer ist angemeldet und befindet sich auf der Seite des Projekts oder Bereichs, wo neuer Content hinzugefügt werden soll

main flow

1. Der Nutzer wählt die Option, neuen Content hinzuzufügen.
2. Das System zeigt dem Nutzer ein Formular oder eine Eingabemaske, in der er relevante Informationen für den neuen Content eingeben kann (z. B. Titel, Beschreibung, Dateianhänge).
3. Der Nutzer gibt die erforderlichen Informationen ein und lädt bei Bedarf Dateien hoch.
4. Der Nutzer bestätigt die Eingabe, und das System speichert den neuen Content.
5. Das System zeigt eine Bestätigungsmeldung an und aktualisiert die Ansicht, um den neuen Content anzuzeigen.

postcondition Content hinzugefügt

Es wurde neuer Content zu einer Collection hinzugefügt

end Content hinzufügen

- Um klar kommunizieren zu können, welche Funktionalitäten für einen bestimmten Use Case erfüllt sein sollen, wurden die Use Cases zudem noch textuell spezifiziert
- Genaue Spezifikationen sind den Abbildungen zu entnehmen

Tech-Stack

Frontend

Kategorie	HTML	XML	JSX
Bekanntheit	Standard und weit verbreitet.	Eher speziell, nicht so verbreitet.	Spezifisch für React.
Leichtgewichtigkeit	Einfache Syntax und Struktur.	Schwere Syntax im Vergleich zu HTML.	Spezifisch für React.
Aufwand	-	Leichte Lernkurve.	Lernkurve.

- Statische visuelle Komponenten werden mit HTML, CSS und JavaScript umgesetzt, da diese Technologien standardisiert sind und eine hohe Geschwindigkeit des Frontends gewährleisten
- Außerdem eine flache Lernkurve und daher zeitlich sehr effizient

Tech-Stack

Frontend

Kategorie	CSS	SCSS / Less
Leichtgewichtigkeit	Einfache Syntax und Struktur.	Bietet erweiterte Funktionen, aber Lernkurve
Aufwand	-	Lernkurve für Entwickler, die nicht mit Präprozessoren vertraut sind.

Tech-Stack

Frontend

Kategorie	JS	TypeScript	ES6
Bekanntheit	Grundlegende Sprache für Webentwicklung.	Typisierte Übermenge von JavaScript	Moderne JavaScript-Funktionen.
Aufwand	-	Etwas steilere Lernkurve	Einige Funktionen möglicherweise nicht in älteren Browsern unterstützt

Tech-Stack

Frontend

Kategorie	Vue.js	React	Angular
Leichtgewichtigkeit	Einfache Integration und Verwendung.	Große Community, mehr Ressourcen.	Große Community, mehr Ressourcen.
Dokumentation	Gut Dokumentiert.	Gut Dokumentiert.	Gut Dokumentiert.
Aufwand	Für umfangreichere Projekte könnte React/ Angular von größeren Communities profitieren.	-	-

- Dynamisch generierte Inhalte sollen durch die Integration von Vue.js realisiert werden, was durch einen einfachen Import via JavaScript ermöglicht wird
- Da Vue.js außerdem in vergangenen Veranstaltungen behandelt wurde, ist auch hier die Lernkurve flach, was zeitlich gesehen, sehr effizient ist

Tech-Stack

Backend

Kategorie	Node.js / Express	Koa.js/Hapi.js
Leichtgewichtigkeit	Sehr leichtgewichtig und schnell.	Auch leichtgewichtig, aber möglicherweise spezifischere Use Cases.
Skalierbarkeit	Bekannt für hohe Geschwindigkeit und Skalierbarkeit.	Gute Skalierbarkeit, aber möglicherweise weniger schnell als Node.js.
Community-Support	Große und aktive Community.	Aktive Community, aber möglicherweise kleiner als die

- Der Server basiert auf Node.js und Express.js
- Node.js wurde aufgrund seiner hervorragenden Geschwindigkeit und Skalierbarkeit ausgewählt
- Außerdem gibt es viele Funktionen, die durch die Community bereits als Packages zur Verfügung gestellt werden
- Express.js zeichnet sich durch seine schlanke Struktur aus, die zu einer besonders schnellen Performance beiträgt

Tech-Stack

Backend

		Use Cases.
Skalierbarkeit	Bekannt für hohe Geschwindigkeit und Skalierbarkeit.	Gute Skalierbarkeit, aber möglicherweise weniger schnell als Node.js.
Community-Support	Große und aktive Community.	Aktive Community, aber möglicherweise kleiner als die von Node.js/Express.js
Dokumentation	Gut dokumentiert.	Gut dokumentiert, aber möglicherweise kleiner als die von Node.js/Express.js.

Tech-Stack

Backend

Kategorie	PostgreSQL mit Neon	MySQL, MongoDB, SQLite
Datenbanktyp	Relationale Datenbank.	Unterschiedliche Typen: MySQL (relational), MongoDB (NoSQL), SQLite (eingebettet).
ACID-Transaktionen	Unterstützt ACID-Transaktionen für Datenintegrität.	Je nach Typ können ACID-Transaktionen unterschiedlich implementiert sein.
Performance	Sehr schnell, besonders in Verbindung mit Neon.	Performance variiert je nach Typ und Anwendungsfall.

- Datenbank wird relational sein, um eine strukturierte Verwaltung der Daten zu ermöglichen
- Der bevorzugte Dialekt für die Datenbank ist PostgreSQL, basierend auf persönlichen Entscheidungen
- Die Datenbank wird auf Neon gehostet, da Neon für seine Schnelligkeit bekannt ist und im Vergleich zur Konkurrenz durchgehend abrufbar ist, ohne, dass der Server erst neu starten muss

Tech-Stack

Backend

		SQLite (eingebettet).
ACID-Transaktionen	Unterstützt ACID-Transaktionen für Datenintegrität.	Je nach Typ können ACID-Transaktionen unterschiedlich implementiert sein.
Performance	Sehr schnell, besonders in Verbindung mit Neon.	Performance variiert je nach Typ und Anwendungsfall.
Persönliche Präferenz	Persönliche Entscheidung.	Abhängig von den spezifischen Anforderungen und Präferenzen des Teams.

Tech-Stack

Backend

Kategorie	SocketIO	WebSocket-Implementierungen in anderen Sprachen
Echtzeitkommunikation	Sehr leichtgewichtig, unterstützt Real-Time Data.	Direktere Kontrolle über WebSocket-Implementierungen, könnten komplexer zu handhaben sein.
Integration mit Node.js	Sehr gut unterstützt in Node.js durch eigenes Package.	Implementierung in anderen Sprachen möglicherweise spezifischer.

- Echtzeitdaten werden über SocketIO ausgetauscht
- Socket.io wird in Node.js durch ein eigenes Package optimal unterstützt und bietet trotz seiner umfangreichen Funktionalität eine einfache Implementierung
- Außerdem wurde auch SocketIO in vergangenen Modulen bereits behandelt, wodurch auch hier die Lernkurve flach ist und die Nutzung dieser Technologie somit zeitlich effizient ist

Tech-Stack

Backend

Integration mit Node.js	Sehr gut unterstützt in Node.js durch eigenes Package.	Implementierung in anderen Sprachen möglicherweise spezifischer.
Funktionsumfang	Hoher Funktionsumfang trotz leichtgewichtiger Natur.	Abhängig von der Implementierung, könnte weniger Funktionalität bieten als SocketIO.
Kompatibilität	Weit verbreitet und gut kompatibel mit verschiedenen Plattformen.	Kompatibilität kann je nach Implementierung variieren.

Tech-Stack

Authentifizierung

Kategorie	OAuth 2.0	JWT (JSON Web Tokens)	OpenID Connect
Sicherheit	Hohe Sicherheit durch etablierten Industriestandard.	Sicher, aber erfordert sichere Handhabung von Secret Keys.	Basierend auf OAuth 2.0, bietet zusätzliche Identitätsmanagement funktionen.
Verbreitung	Breit unterstützter Standard in der Industrie.	Weit verbreitet, besonders für tokenbasierte Authentifizierung.	Verbreitet, bietet einen Standard für Identitätsmanagement.

- Für die Authentifizierung wird OAuth 2.0 verwendet
- Diese Technologie wird sowohl von den externen Diensten, die in diesem Projekt verwendet werden, gefordert, als auch aufgrund ihrer hohen Sicherheitsstandards ausgewählt

Tech-Stack

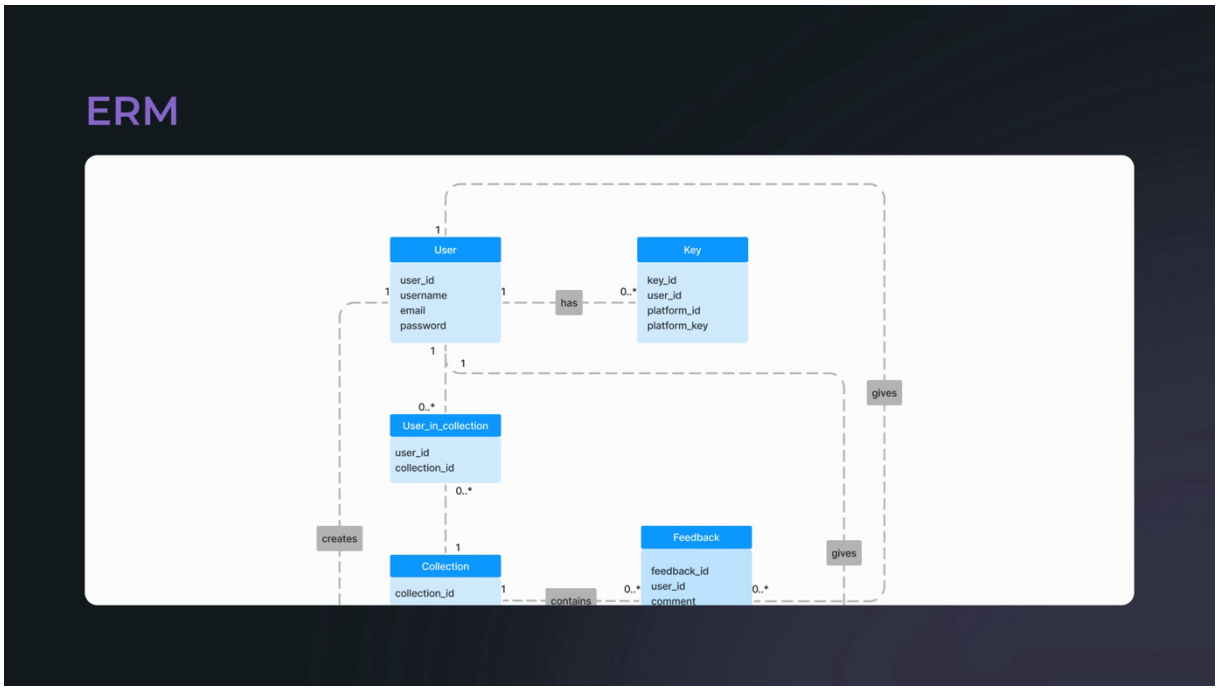
Authentifizierung

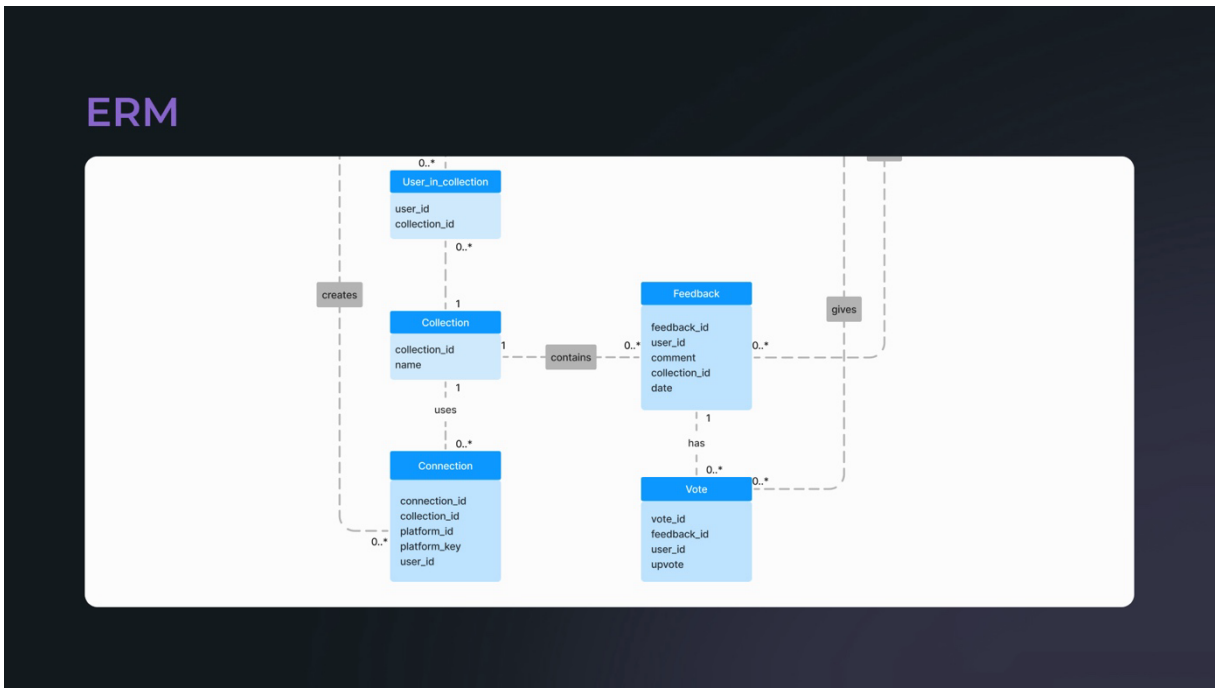
Komplexität	Komplexer, da mehrere Schritte und Rollen beteiligt sind.	Einfach zu implementieren und zu verstehen.	Zusätzliche Funktionen können die Implementierung komplexer machen.
Flexibilität	Relativ flexibel, ermöglicht verschiedene Authentifizierungstyp en.	Begrenzte Flexibilität, spezialisiert auf tokenbasierte Authentifizierung.	Flexibel, bietet zusätzliche Funktionen für Identitätsmanagement.
Token-Ausstellung	Token durch	Token werden vom	Token-Ausstellung

Tech-Stack

Authentifizierung

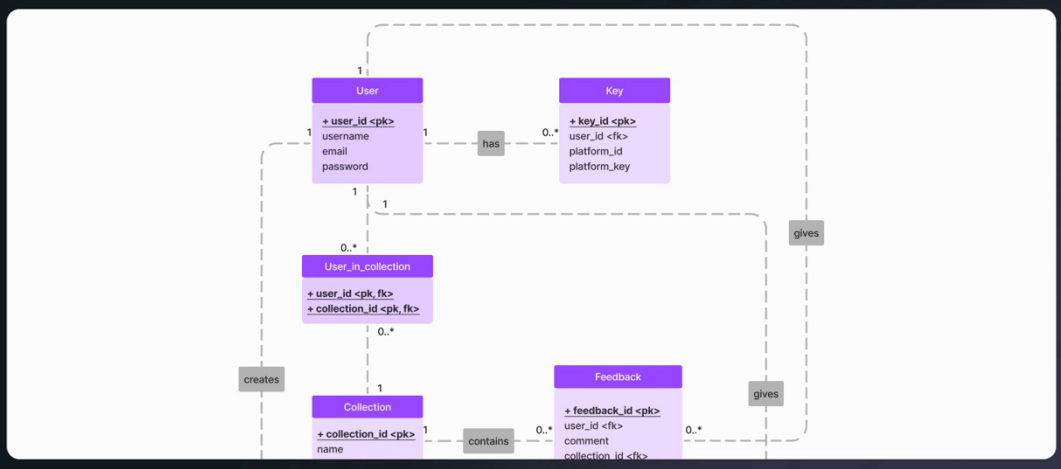
	verschiedene Authentifizierungstypen.	tokenbasierte Authentifizierung.	für Identitätsmanagement.
Token-Ausstellung	Token durch Authorisierungsserver ausgestellt.	Token werden vom Server signiert und können dezentral überprüft werden.	Token-Ausstellung durch Identitätsprovider.
Nutzungsszenarien	Geeignet für Anwendungen mit externen Diensten, die OAuth 2.0 unterstützen.	Gut für webbasierte Anwendungen, APIs und Single Sign-On.	Spezialisiert auf Identitätsmanagement, gut für Single Sign-On.



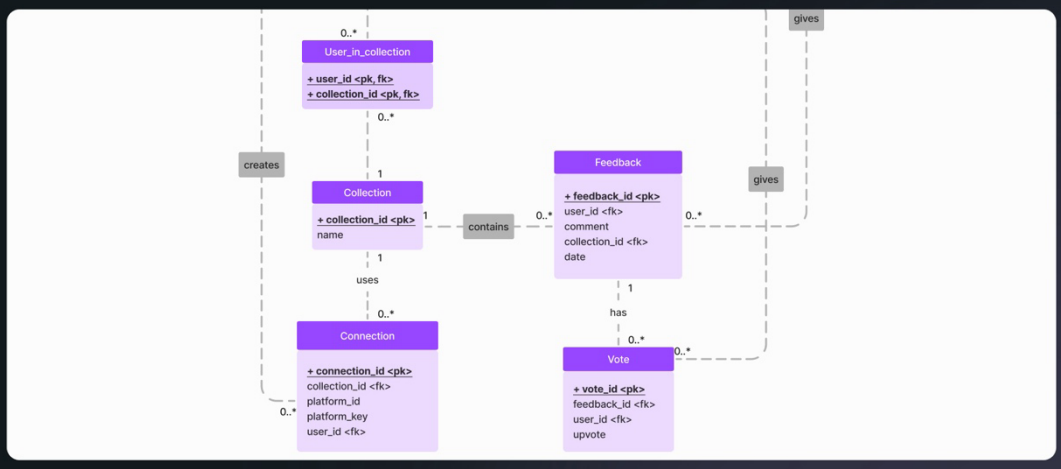


- Es werden sieben Tabellen benötigt
- Bei der Modellierung wurde außerdem schon darauf geachtet, dass die ersten drei Normalformen eingehalten werden

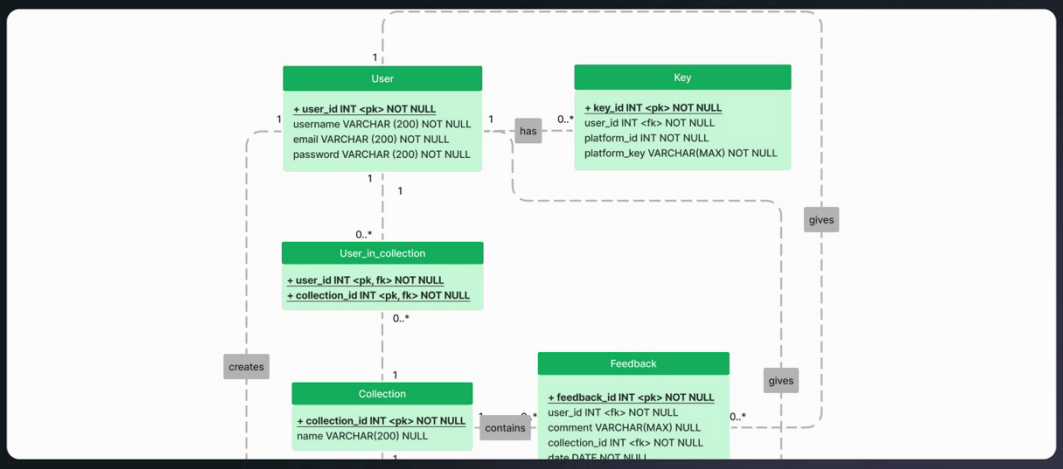
eERM


User
+ user_id <pk>
username
email
password
Key
+ key_id <pk>
user_id <fk>
platform_id
platform_key
has
gives
User_in_collection
+ user_id <pk, fk>
+ collection_id <pk, fk>
creates
Collection
+ collection_id <pk>
name
contains
Feedback
+ feedback_id <pk>
user_id <fk>
comment
collection_id <fk>
gives

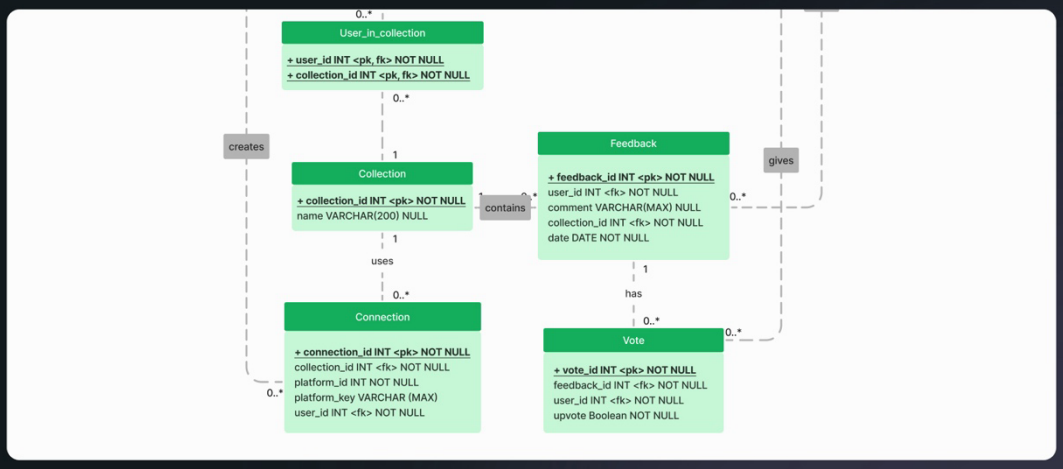
eERM


gives
User_in_collection
+ user_id <pk, fk>
+ collection_id <pk, fk>
creates
Collection
+ collection_id <pk>
name
contains
Feedback
+ feedback_id <pk>
user_id <fk>
comment
collection_id <fk>
date
gives
uses
has
Connection
+ connection_id <pk>
collection_id <fk>
platform_id
platform_key
user_id <fk>
Vote
+ vote_id <pk>
feedback_id <fk>
user_id <fk>
upvote

Relationenmodell



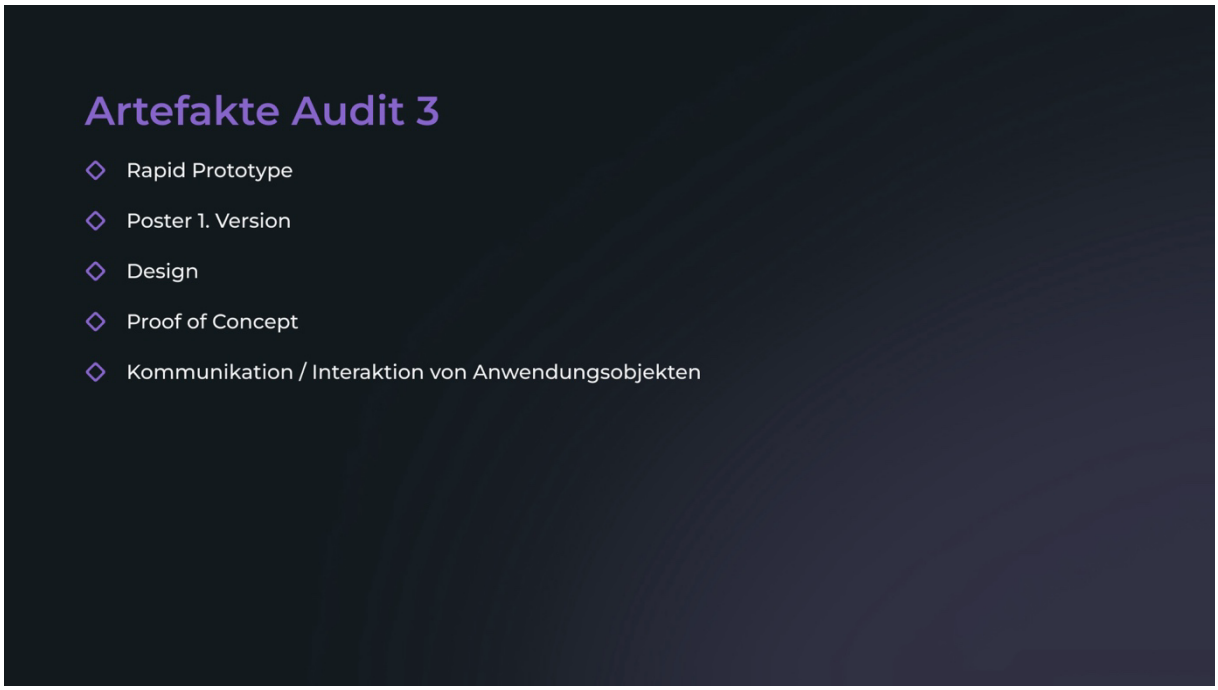
Relationenmodell



Erste Projektrisiken

Risiko	Gegenmaßnahmen
Datenleck	Verschlüsselungen, Regelmäßige Wartung, Regelmäßige Updates.
Zu hohe Kosten	Evaluation des Budget, Gute Planung.
Geringe Einnahmen	Marketing, Werbung, Überwachung des Cashflows, Änderung der Rechtsform
Keine Möglichkeit für Erweiterungen	Festlegung von Standards / Standardisierung

- Größtes Risiko ist Datenleck
- Lösen kann man das über Verschlüsselung, regelmäßige Wartung und regelmäßige Updates
- Außerdem zu hohe Kosten, was durch eine gute Planung und Evaluation umgangen werden kann
- Geringe Einnahmen können v.a. durch gutes Marketing verhindert werden
- Außerdem sollte System so geschrieben sein, dass Erweiterungen einfach eingefügt werden können



- Unsere Artefakte für Audit 3:
  - Rapid Prototype
  - Poster 1. Version
  - Design
  - Proof of Concept
  - Kommunikation / Interaktion von Anwendungsobjekten