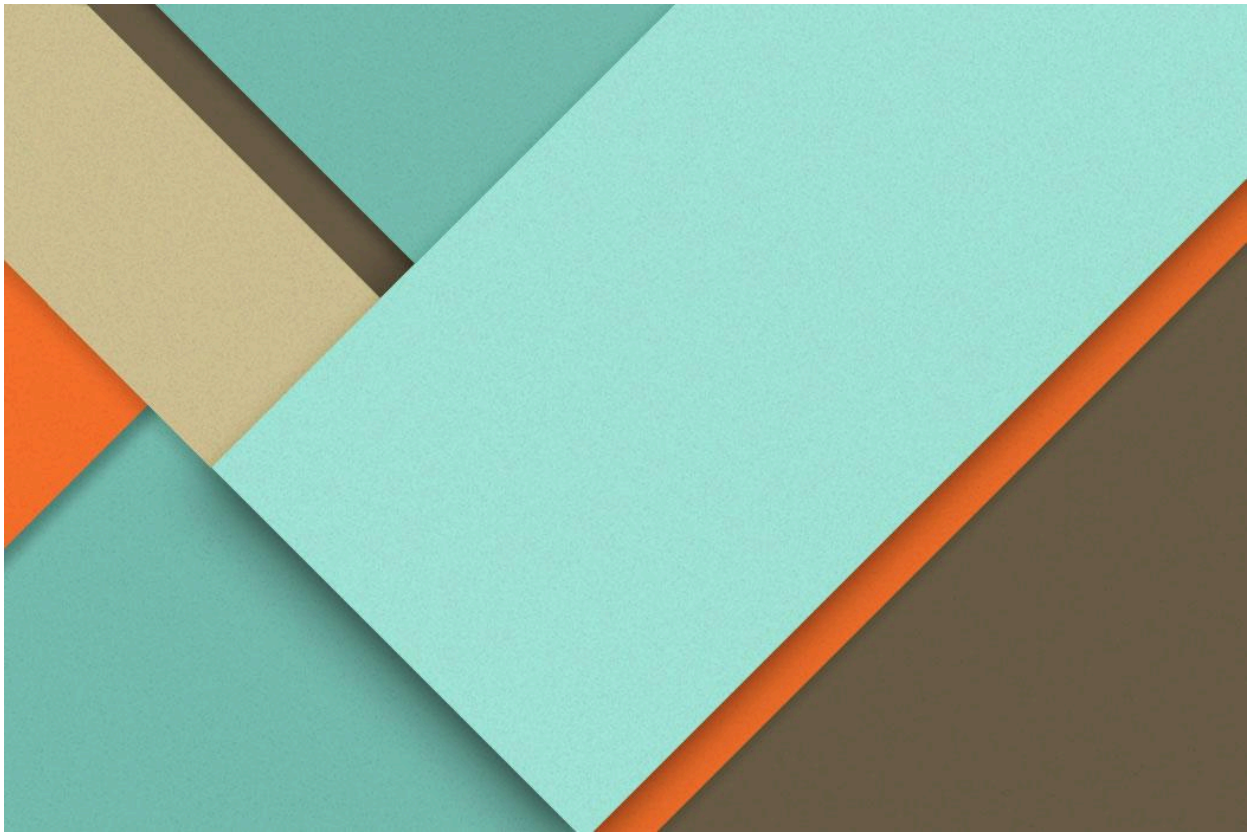


Habito Systemdokumentation

01.02.2025

—

Gruppenmitglieder
Samir Wadi, Claudius Naß, Emre Gül, Muhammed Aslan, Jasmin Kreho, Mahdi Amouri

Projektbetreuung
Angelika Hönemann

Durchführungszeitraum
01.10.2024 - 02.02.2025

Inhaltsverzeichnis

1. **Einleitung**
2. **Projektbeschreibung**
   - 2.1. Ziel der Webanwendung
   - 2.2. Nutzen der Webanwendung
   - 2.3. Besonderheiten der Webanwendung
3. **Projektplanung**
   - 3.1. Teamvorstellung mit Zuständigkeiten
   - 3.2. Planungstools und Entwicklungsmethode
4. **Link zur Deployten app**
5. **Use-Cases**
   - 5.1. User Use-Cases
     - 5.1.1. Registrierung eines neuen Benutzers
     - 5.1.2. Anmeldung eines Benutzers
     - 5.1.3. Erstellung eines neuen Habits
     - 5.1.4. Abschluss eines Habits
     - 5.1.5. Bearbeitung eines Habit
     - 5.1.6. Löschen eines Habits
     - 5.1.7. Verwalten von Freundschaften
     - 5.1.8. Verwalten von Missionen
     - 5.1.9. Einkauf von Gegenständen im Shop
     - 5.1.10. Bearbeitung des Profils
     - 5.1.11. Verschicken von Nachrichten
   - 5.2. System Use-Cases
     - 5.2.1. Tägliches zurücksetzen der Habits
     - 5.2.2. Berechnung der XP und Level-UP
     - 5.2.3. Fortsschrittsprüfung für Missionen
     - 5.2.4. Kauf von Gegenständen im Shop
     - 5.2.5. Verwaltung von Freundschaftsanfragen

6. Mockups

6.1. Registrierung

6.2. Login

6.3. Dashboard

6.4. Missionen

6.5. Habit erstellen

6.6. Habit abschließen

6.7. Shop

6.8. Tutorial

6.9. Freunde

6.10. Profil

7. Code-/System-Architektur/Struktur

7.1. Frontend, Backend, Datenbank

7.2. Interne und externe APIs

7.3. Komponenten

7.4. Abhängigkeiten

7.5. Verwendete Tools

7.6. Testumgebung

7.7. Diagramme

7.7.1. Klassendiagramm

8. **Arbeitsprozess**

8.1. Organisation

8.2. Herausforderungen

8.2.1. Lernerfolg

8.2.2. Probleme

8.2.3. Geplante vs. umgesetzte Anforderungen

9. **Quellen**

Inhaltsverzeichnis **1**

1. Einleitung **4**

2. Projektbeschreibung **5**

2. 1 Ziel der Webanwendung 5

2. 2 Nutzen der Anwendung 5

2. 3 Besonderheiten der Webanwendung 6
3. Projektplanung 7
3.1 Teamvorstellung mit Zuständigkeiten 7
4. Link Zur Deployten App 7
5. Use-Cases 8
5.1. User Use-Cases 8
5.2. System Use-Cases 9
6. Mockups 10
6.1 Login / Registration 10
6.2 Dashboard: 11
6.3 Tutorial: 12
6.4 Shop: 13
6.5 Missionen: 14
6.6 Profil: 15
6.7 Farbschema 15
6.8 Companions 16
7. Code-/System-Architektur/Struktur 17
7.1. Frontend, Backend, Datenbank 17
7.2. Interne Routes und externe APIs 18
/register: 19
/google: 19
/verify: 19
/user-missions: 19
/friends 19
/friend-requests 19
Externe API-Routen 19
Google OAuth 19
Cloudinary API (Bild-Upload) 20
7.3. Komponenten 20
7.4. Abhängigkeiten 25
7.5. Verwendete Tools 25
7.6. Testumgebung 26
7.6.1 Jest 26
7.6.2 Cypress 26
Cypress als Testumgebung (Frontend) 26
7.7. Diagramme 26
Systemarchitektur Diagramm: 27
8. Arbeitsprozess 27
8.1. Organisation 27

8.2. Herausforderungen 28
8.2.1. Lernerfolg(e) 28
8.2.2. Probleme 28
8.2.3. Geplante und umgesetzte Anforderungen 29

1. Einleitung

Habito ist eine Webanwendung, die Nutzern dabei hilft, verschiedene Gewohnheiten (Habits) zu erstellen und zu verfolgen, um ihre persönlichen Ziele zu erreichen. Nutzer können Habits aus verschiedenen Lebensbereichen wie Ernährung, Sport, Bildung oder Entspannung auswählen und sie nach Kategorien gruppieren. Durch das Durchhalten und Verfolgen von Habits sammeln Nutzer Erfahrungspunkte (XP), die ihre Fortschritte anzeigen.

2. Projektbeschreibung

2. 1 Ziel der Webanwendung

Ein zentrales Ziel von **Habito** ist es, Nutzer durch eine gamifizierte Habit-Tracking-Erfahrung zu motivieren, kontinuierlich positive Gewohnheiten in ihren Alltag zu integrieren. Dabei soll die Anwendung helfen, langfristige Verhaltensänderungen zu etablieren, indem sie Fortschritt sichtbar macht, Belohnungen in Form von XP bietet und soziale Interaktionen ermöglicht.

2. 2 Nutzen der Anwendung

Der Nutzen von Habito liegt darin, Nutzer durch eine strukturierte und motivierende Umgebung dabei zu unterstützen, positive Gewohnheiten nachhaltig in ihren Alltag zu integrieren. Die Anwendung bietet:

- Gamification: Erfahrungspunkte (XP), Belohnungspunkte, Streaks und Belohnungen steigern die Motivation, Habits regelmäßig auszuführen.
- Struktur & Übersicht: Nutzer können ihre Gewohnheiten nach Kategorien ordnen und ihren Fortschritt visuell verfolgen.
- Flexibilität: Habits lassen sich individuell anpassen und mit der Zeit weiterentwickeln.
- Soziale Interaktion: Gemeinsame Habits mit Freunden erhöhen die Verbindlichkeit und fördern gegenseitige Motivation.
- Datenbasierte Analyse: Nutzer erhalten Einblicke in ihr Verhalten, um ihre Routinen zu optimieren und eine ausgewogene Lebensweise zu fördern.
- Dadurch hilft Habito, persönliche Ziele effizienter zu erreichen und langfristig gesunde, produktive Routinen aufzubauen.

2. 3 Besonderheiten der Webanwendung

Gamification & Belohnungssystem

- XP & Level-Ups: Nutzer sammeln Erfahrungspunkte für abgeschlossene Habits und können sich weiterentwickeln.
- Missionen & Herausforderungen: Spezielle Aufgaben motivieren zur langfristigen Nutzung.
- Skins & Avatare: Kategorienpezifische Skins (z. B. ein Faultier für Meditation, ein Gorilla für Sport) machen Fortschritte visuell erlebbar.
- Boost-Items: Tokens für doppelte XP oder zum Einfrieren von Streaks erhöhen die Flexibilität.

Dynamische Habit-Entwicklung

- Steigende Herausforderung: Nach 14 Tagen kann ein Habit aufgewertet werden, um größere Fortschritte zu erzielen.
- Individuelle Anpassung: Nutzer bestimmen Dauer, Aufwand und Häufigkeit ihrer Habits selbst.
- Vorlagen von anderen Nutzern: Inspiration durch Community-geteilte Habits.

Soziale Funktionen & Motivation

- Gemeinsame Habits: Nutzer können mit Freunden zusammen an Habits arbeiten und sich gegenseitig motivieren.
- Vergleichbare Fortschritte: Ein Dashboard zeigt Statistiken über verschiedene Habit-Kategorien.

Intelligente & Nutzerfreundliche Features

- KI-gestützte XP-Anpassung: Die App berechnet, ob XP-Belohnungen fair sind.
- Usability-Fokus: Simpler Modus ohne Gamification für Nutzer, die nur ihre Fortschritte tracken wollen.
- Tutorial mit Maskottchen: Nutzer werden schrittweise eingeführt und erleben die App erst, bevor sie sich registrieren müssen.

3. Projektplanung

3.1 Teamvorstellung mit Zuständigkeiten

Muhammed:

- erstellen von Unit und Integrationstests aller Backend Komponenten
- erstellung und überarbeitung des Backends

- Erstellung des Frontends
- UI Design im Frontend sowie das hinzufügen von Animationen
- Verbindung vom Backend zum Frontend

Jasmin:

- Backend Entwicklung
- Zuständig für die erstellung einer CI / CD Pipeline

Mahdi:

- Backend Entwicklung
- hosting der Datenbank

Samir:

- erstellen des UI / UX Design
- Implementierung des UI im Frontend

Emre:

- Testen des React Frontends mit einem geeigneten Framework (Cypress)

Claudius:

- Frontend Entwicklung

4. Link Zur Deployten App

https://habitox-app.onrender.com

5. Use-Cases

5.1. User Use-Cases

Registrierung eines neuen Benutzers:

- Der Benutzer gibt seine Daten wie Benutzername, E-Mail und Passwort ein, um ein neues Konto zu erstellen, daraufhin wird an seine E-Mail eine Bestätigungsmail für die Authentifizierung gesendet, um die Registrierung erfolgreich abzuschließen. Nach erfolgreicher

Registrierung kann er sich anmelden und auf die Funktionen der Webanwendung zugreifen.

Anmeldung eines Benutzers:

- Der Benutzer gibt seine Zugangsdaten ein, um sich anzumelden. Nach erfolgreicher Authentifizierung wird er auf das Dashboard weitergeleitet.

Erstellung eines neuen Habits:

- Der Benutzer kann einen neuen Habit erstellen, indem er einen Namen, eine Kategorie und die gewünschte Häufigkeit angibt. Dieses Habit wird dann in seine persönliche Habit-Liste aufgenommen.

Abschluss eines Habits:

- Sobald ein Habit abgeschlossen ist, kann der Benutzer den Fortschritt markieren, wodurch XP-Punkte gutgeschrieben werden, je öfter und konsistenter er dieses Habit abschließt, desto höher steigt seine Streak und die XP-Erhaltung multipliziert sich.

Bearbeitung eines Habits:

- Der Benutzer kann bestehende Habits ändern, z. B. Name, Kategorie oder Häufigkeit, um sie seinen aktuellen Zielen anzupassen.

Löschen eines Habits:

- Ein Habit kann aus der Liste entfernt werden, wenn er nicht mehr benötigt wird.

Verwalten von Freundschaften:

- Der Benutzer kann Freundschaftsanfragen senden, annehmen oder ablehnen. Freunde können gegenseitig miteinander chatten.

Verwalten von Missionen:

- Der Benutzer kann aktive Missionen auswählen und an deren Fortschritt arbeiten, um Punkte für den Shop zu erhalten.

Einkauf von Gegenständen im Shop:

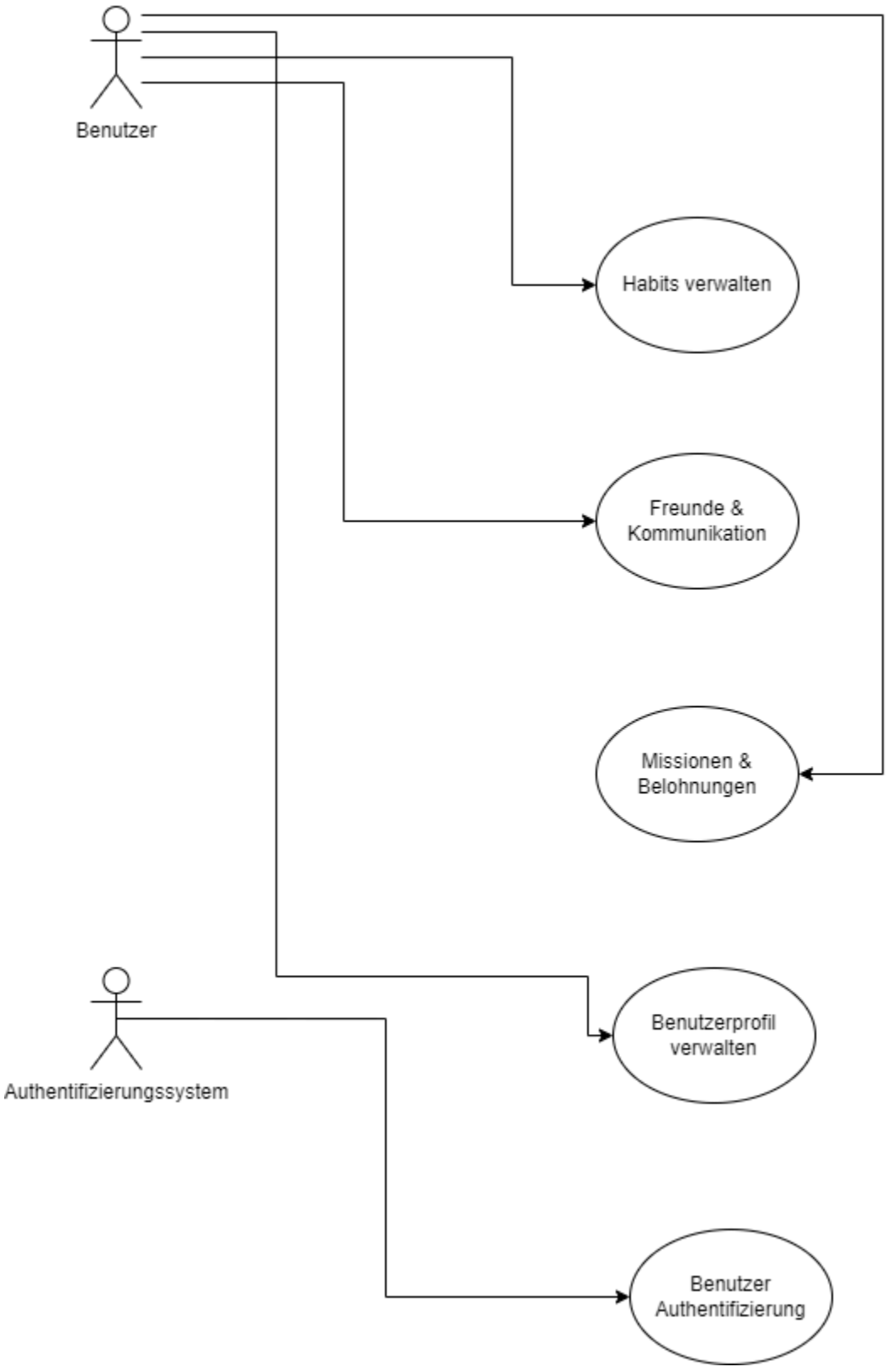
- Der Benutzer kann mit gesammelten Punkten Companions kaufen, die in seinem Inventar gespeichert werden, mit diesen kann er dann den jetzigen Companion austauschen.

Bearbeitung des Profils:

- Der Benutzer kann sein Profilbild ändern, persönliche Informationen anpassen und Statistiken über seinen Fortschritt einsehen.

Verschicken von Nachrichten:

- Nachrichten können zwischen Benutzern gesendet werden, um sich gegenseitig zu motivieren oder Pläne zu teilen.


Benutzer
Habits verwalten
Freunde & Kommunikation
Missionen & Belohnungen
Authentifizierungssystem
Benutzerprofil verwalten
Benutzer Authentifizierung

5.2. System Use-Cases

Tägliches Zurücksetzen der Habits:

- Das System setzt jeden Tag den Status der Habits zurück, sodass der Benutzer erneut daran arbeiten kann.

Berechnung der XP und Level-Up:

- XP wird basierend auf abgeschlossenen Habits und Missionen berechnet. Nach einer bestimmten Anzahl von XP erreicht der Benutzer ein neues Level.

Fortschrittsprüfung für Missionen:

- Das System überprüft den Fortschritt von Missionen und belohnt den Benutzer bei deren Abschluss Punkten.

Kauf von Gegenständen im Shop:

- Das System überprüft die Punkte des Benutzers, reduziert diese beim Kauf eines Gegenstands und fügt den Gegenstand zum Inventar hinzu.

Verwaltung der Companions:

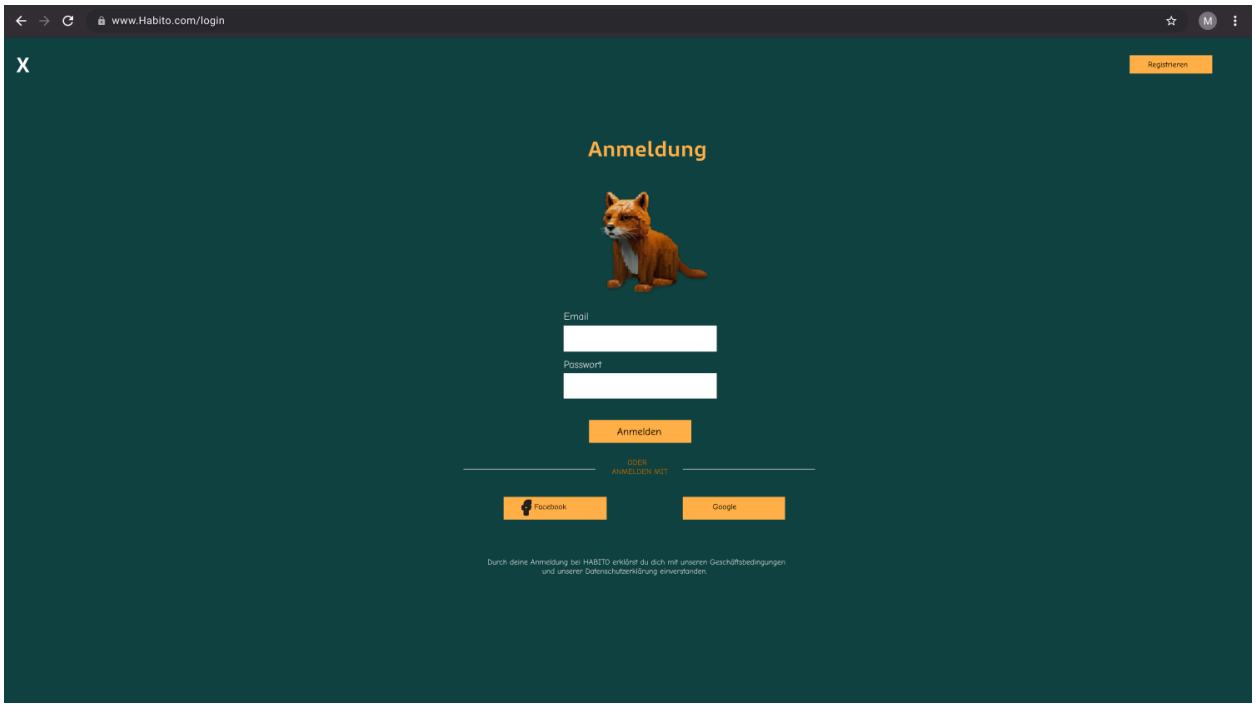
- Durch die Companions im Inventar kann der ausgerüstete Companion ausgetauscht werden, so dass auf jeder Seite der neue Companion angezeigt wird.

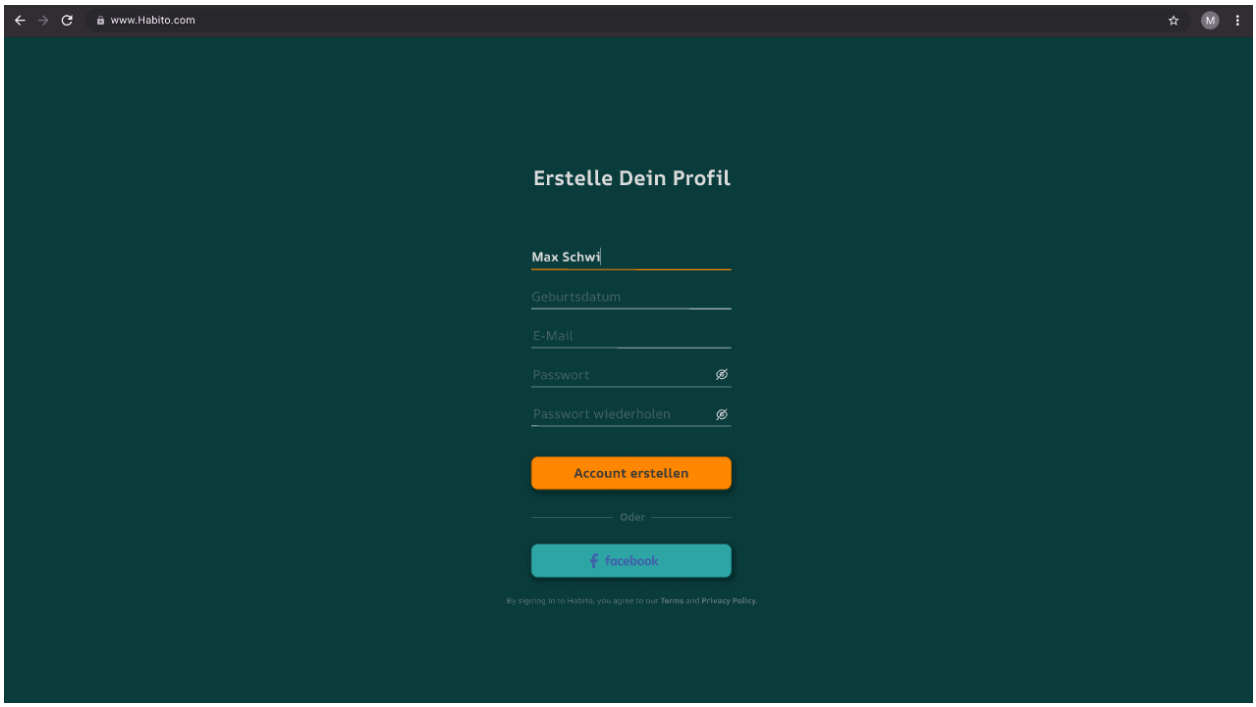
Verwaltung von Freundschaftsanfragen:

- Das System verarbeitet gesendete Freundschaftsanfragen, zeigt deren Status an und synchronisiert Änderungen zwischen Benutzern.

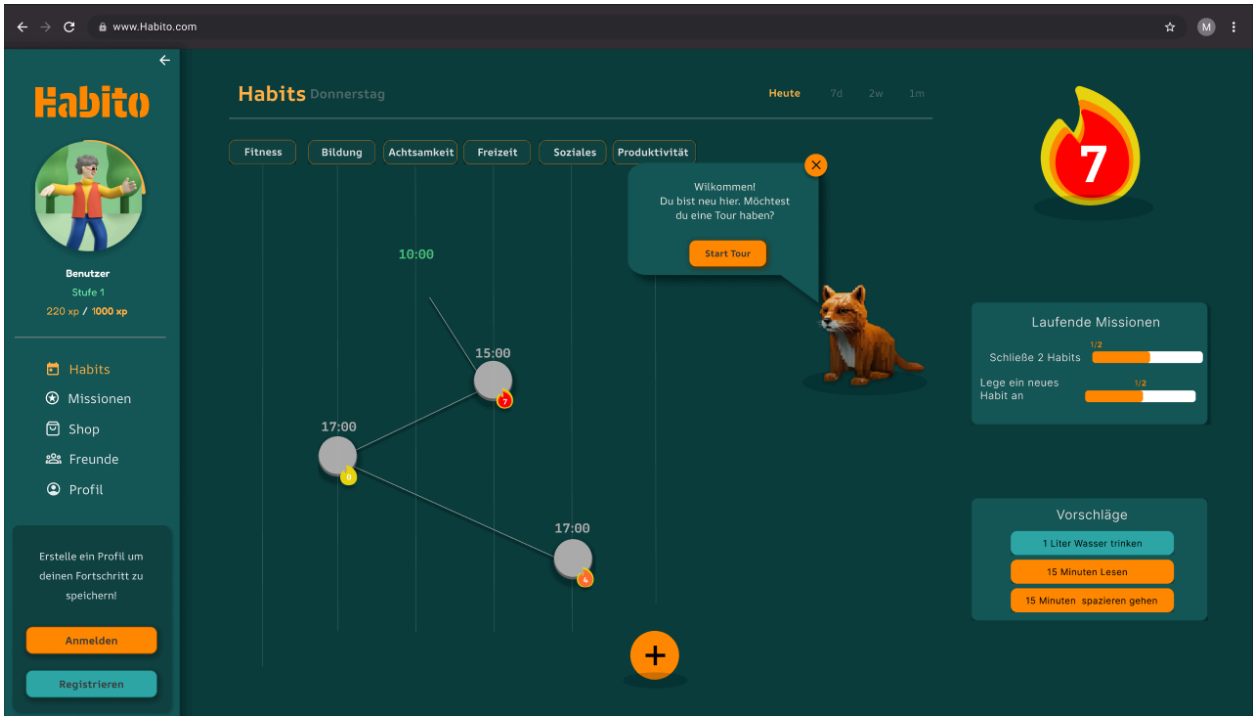
6. Mockups

6.1 Login / Registration

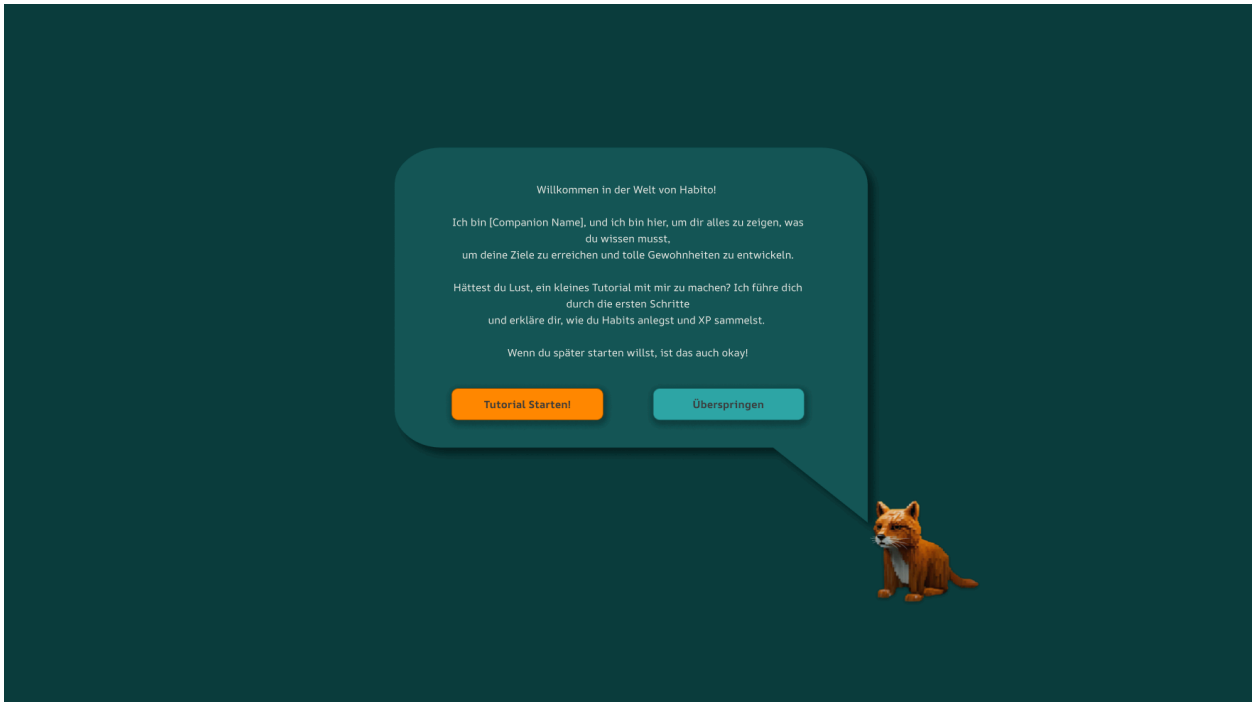


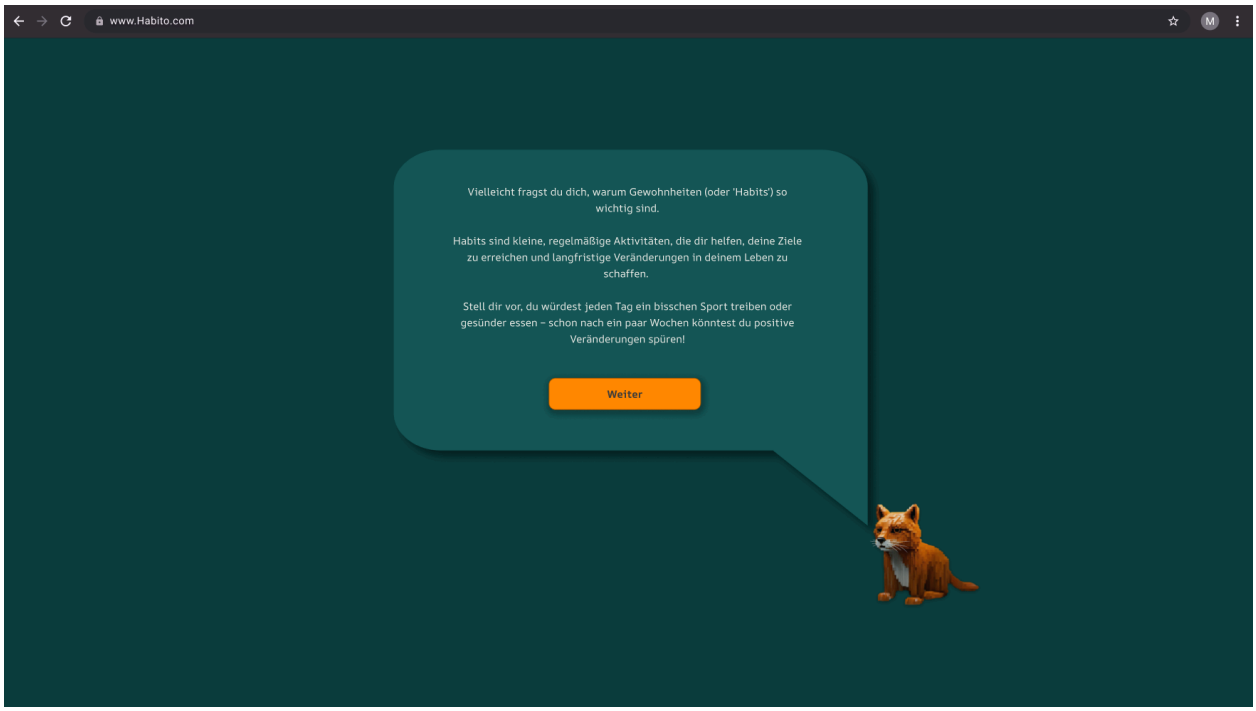


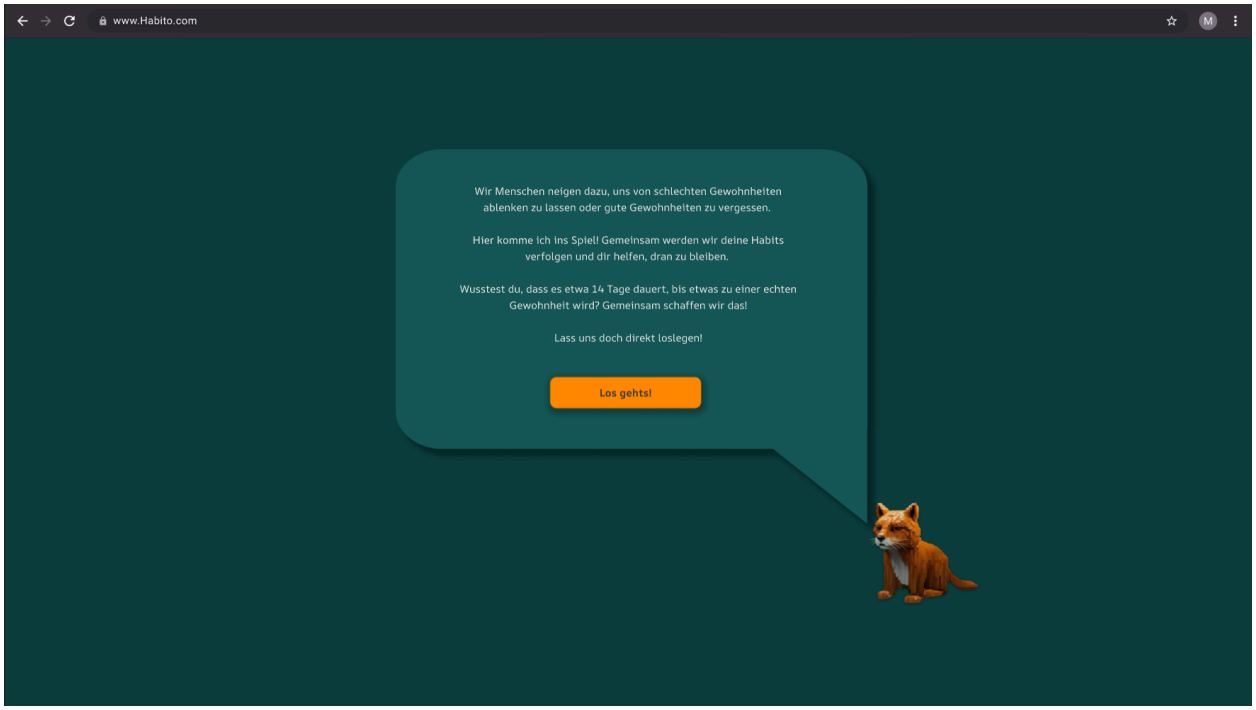
6.2 Dashboard:



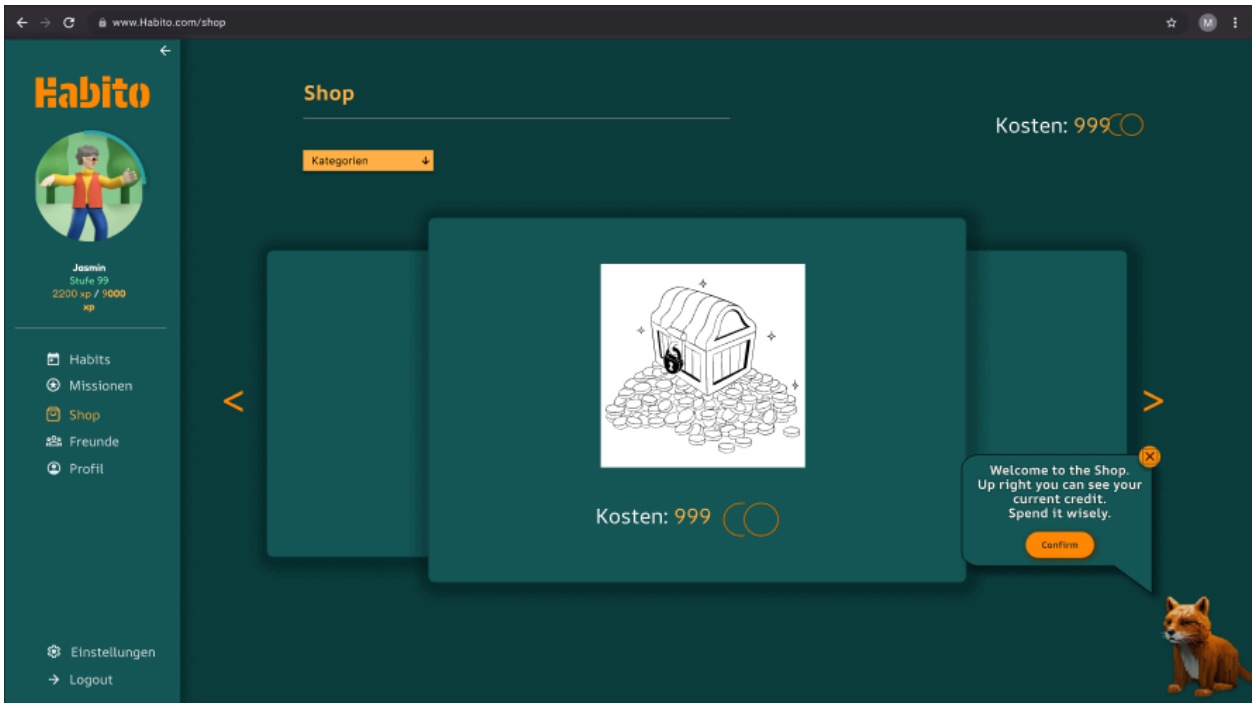
6.3 Tutorial:



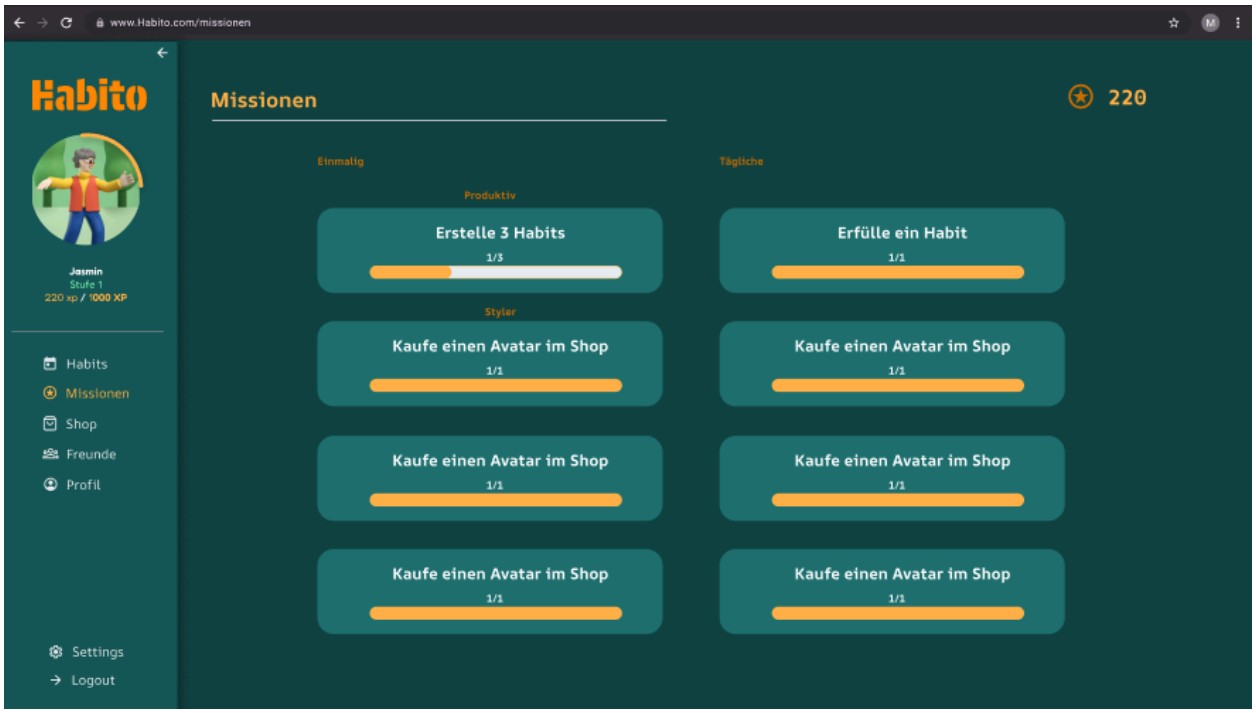

www.Habito.com
Vielleicht fragst du dich, warum Gewohnheiten (oder 'Habits') so wichtig sind.
Habits sind kleine, regelmäßige Aktivitäten, die dir helfen, deine Ziele zu erreichen und langfristige Veränderungen in deinem Leben zu schaffen.
Stell dir vor, du würdest jeden Tag ein bisschen Sport treiben oder gesünder essen – schon nach ein paar Wochen könntest du positive Veränderungen spüren!
Weiter


www.Habito.com
Wir Menschen neigen dazu, uns von schlechten Gewohnheiten ablenken zu lassen oder gute Gewohnheiten zu vergessen.
Hier komme ich ins Spiel! Gemeinsam werden wir deine Habits verfolgen und dir helfen, dran zu bleiben.
Wusstest du, dass es etwa 14 Tage dauert, bis etwas zu einer echten Gewohnheit wird? Gemeinsam schaffen wir das!
Lass uns doch direkt loslegen!
Los gehts!

6.4 Shop:



6.5 Missionen:



6.6 Profil:

6.7 Farbschema

Farben

Grüntöne

Dunkles Hintergrundgrün:
dunkles Grün, das gut als Haupt-Hintergrundfarbe passt
#0B3D3D

Mittleres Grün:
für Buttons, Kartenhintergründe oder aktive Elemente
#145555

Highlight-Grün:
zum Hervorheben von aktiven oder ausgewählten Elementen
#1C7C7C

Helles Akzentgrün:
zur Nutzung als sekundäre Akzentfarbe
#2FA9A9

Orangetöne

Primäres Orange:
für Buttons, wichtige Hinweise oder Call-to-Action-Elemente
#FF8800

Dunkles Orange:
zur Unterscheidung von inaktiven Orange-Elementen
#CC7000

Helles Orange:
für Mouseover- oder Hover-Effekte)
#FFB347

Neutraltöne

Dunkles Grau:
für Lesbarkeit bei Texten
#E0E0E0

Helles Grau:
für Linien und Rahmen
#A0A0A0

Schrift

Heading 1 (H1) – 32px

6.8 Companions






7. Code-/System-Architektur/Struktur

7.1. Frontend, Backend, Datenbank

Frontend:

- **Technologie:** Das Frontend basiert auf **React.js mit Typescript** und nutzt **Redux** zur zentralen Zustandsverwaltung.
- **Design und Styling:** Hauptsächlich wurden **React Styled components** benutzt, um das UI zu verbessern und mehr Quality of Life zu implementieren.

Backend:

- **Technologie:** Implementiert mit **Node.js** und **Express.js**. Das Backend wurde mit drei Schichten Architektur implementiert.
- **Middleware:** JWT-basierte Authentifizierung und Middleware für Fehlerbehandlung und Routen-Schutz garantieren die Sicherheit und Stabilität des Systems.

Datenbank:

- Die Anwendung verwendet **MongoDB** als NoSQL-Datenbank, die sich durch ihre Flexibilität und Skalierbarkeit auszeichnet. Durch die dokumentenbasierte Struktur können Daten leicht modelliert und bei Bedarf erweitert werden.

7.2. Interne Routes und externe APIs

/users:

- Ermöglicht die Benutzerregistrierung, Anmeldung und Bearbeitung von Profilinformationen..
- Abrufen und Aktualisieren von Punkten und Inventar.
- Hochladen eines Profilbildes.

/habits:

- Das Hinzufügen von XP nach Abschließen der Habits/Gewohnheiten.
- Bereitstellung von Endpunkten für das Erstellen, Bearbeiten, Löschen und Abschließen von Habits.
- Verwaltung von Streaks

/missions:

- Erstellung, Aktualisierung und Löschung von Missionen.
- Prüfung und Aktualisierung des Missionsfortschritts.

/shop:

- Ermöglicht den Abruf und Kauf von Shop-Items.
- Verwaltung des Inventars (Shop-Items werden nach dem Kauf ins Inventar befördert) und Abzug von Punkten nach dem Kauf.
- Das Erstellen der Shop-Items ist möglich.

/inventory:

- Ermöglicht den Abruf von Shop-Items.
- Hinzufügen von Items ermöglicht.

/messages:

- Senden und Empfangen von Nachrichten.
- Verwaltung von Benachrichtigungen und Nachrichtenstatus.

/register:

- Registrierung neuer Benutzer und Verifizierung per Code.

/google:

- Authentifizierung und Anmeldung via Google OAuth.

/verify:

- Verifizierung Zugriffsrechte des eingeloggten Users anhand eines JWT

/user-missions:

- Erstellung und Verwaltung benutzerbezogener Missionen.
- Fortschrittsaktualisierung

/friends

- Abrufen der Freundesliste.

/friend-requests

- Senden, Annehmen und Ablehnen von Freundschaftsanfragen.

Externe API-Routen

Google OAuth

- `/auth/google` – Anmeldung über Google.
- `/google/callback` – Callback nach erfolgreicher Authentifizierung.

Cloudinary API (Bild-Upload)

- Wird für das Hochladen von Benutzerprofilbildern und Companions verwendet.

7.3. Komponenten

Komponente	Beschreibung	Props
AddHabitButton.tsx	Eine Schaltfläche, die es dem Benutzer ermöglicht, ein neues Habit hinzuzufügen. Beim Klicken auf die Schaltfläche wird eine Funktion aufgerufen, die ein Modal zur Erstellung eines neuen Habits öffnet.	OnClick
StarsAnimation	Eine Komponente, die eine Animation von Sternen anzeigt. Diese Animation kann verwendet werden, um besondere Ereignisse wie das Erhalten von XP hervorzuheben.	show
Button.tsx	Eine generische Button-Komponente mit Styling. Diese Komponente kann in der gesamten Anwendung verwendet werden, um konsistente Schaltflächen zu erstellen	children, ...props
Companion.tsx	Die Companion-Komponente zeigt den Begleiter des	-

Komponente	Beschreibung	Props
	Benutzers an. Der Begleiter ist ein Bild, das den Fortschritt des Benutzers in der Anwendung darstellt und durch die Anwendung leitet.	
SocialContainer.tsx	Die SocialContainer-Komponente dient als Container für soziale Interaktionen, wie das Verwalten und Hinzufügen von Freunden. Sie enthält die ManageFriends- und AddFriends-Komponenten.	children (React.ReactNode): Die Inhalte, die innerhalb des Containers angezeigt werden.
ManageFriends.tsx	Die ManageFriends-Komponente ermöglicht es dem Benutzer, seine Freundesliste zu verwalten. Der Benutzer kann Freunde entfernen und Nachrichten an Freunde senden.	keine
AddFriends.tsx	Die AddFriends-Komponente ermöglicht es dem Benutzer, neue Freunde hinzuzufügen, indem er den Benutzernamen des Freundes eingibt und eine Freundschaftsanfrage sendet.	keine
ProfileSocialContainer.tsx	ie ProfileSocialContainer-Komponente dient als Container für soziale Interaktionen im Profilbereich. Sie kann verschiedene soziale Komponenten enthalten.	children (React.ReactNode): Die Inhalte, die innerhalb des Containers angezeigt werden.

Komponente	Beschreibung	Props
InformationsContainer.tsx	Die InformationsContainer-Komponente zeigt Informationen über den Benutzer an, wie den Benutzernamen, das Beitrittsdatum und die aktuelle Streak.	Keine
FriendRequest.tsx	Die FriendRequests-Komponente zeigt die Freundschaftsanfragen des Benutzers an und ermöglicht es, diese anzunehmen oder abzulehnen.	keine
DashboardPage.tsx	Die DashboardPage-Komponente ist die Haupt-Dashboard-Seite, auf der alle Habits des Benutzers angezeigt werden. Sie enthält auch die Navigation, den Begleiter und die Missionsliste.	keine
CreateHabitModal.tsx	Die CreateHabitModal-Komponente ist ein Modal zur Erstellung eines neuen Habits. Sie ermöglicht es dem Benutzer, die Details des neuen Habits einzugeben und zu speichern.	isOpen, onClose, onSave, existingHabits
Inventory.tsx	Die Inventory-Komponente zeigt das Inventar des Benutzers an, einschließlich der gekauften Gegenstände. Der Benutzer kann den Begleiter (Companion) wechseln, indem er auf einen Gegenstand klickt.	items, _id, name, imageUrl, onClose

Komponente	Beschreibung	Props
Input.tsx	Die Input-Komponente ist eine generische Eingabekomponente mit Styling. Sie unterstützt alle Standard-HTML-Attribute für Eingabefelder.	type, placeholder, ...props
HabitInfoPopup.tsx	Die HabitInfoPopup-Komponente zeigt zusätzliche Informationen über ein Habit an, wie den Namen, die Beschreibung, die Zeit und die XP. Das Popup kann ein- und ausgeblendet werden.	name, description, time, xp, isVisible, className
HabitCircles.tsx	Die HabitCircles-Komponente zeigt einen Kreis mit einem Icon, das die Kategorie des Habits darstellt. Der Kreis kann je nach Erfüllungsstatus des Habits unterschiedlich gestylt sein.	category, getCategoryIcon, isFullfilled
Habit.tsx	Die Habit-Komponente zeigt ein einzelnes Habit in der Liste an. Sie enthält Informationen wie den Namen, die Beschreibung, die Zeit, die Kategorie und den Erfüllungsstatus des Habits. Beim Klicken auf das Habit wird ein Modal mit weiteren Details geöffnet.	id, name, description, time, fullfilled, category, frequency, level, xp, duration, durationType, friends, runtime, user, onDeletehabit, isNextHabit, fetchHabits, currentDay, streak
Navbar.tsx	Die Navbar-Komponente ist die Navigationsleiste der Anwendung. Sie enthält Links zu verschiedenen Seiten der Anwendung, zeigt Benutzerinformationen und XP-Leiste an und ermöglicht das Abmelden.	keine

Komponente	Beschreibung	Props
Streak	Die Streak-Komponente zeigt die aktuelle Streak des Benutzers an. Eine Streak repräsentiert die Anzahl der aufeinanderfolgenden Tage, an denen der Benutzer seine Habits erfüllt hat.	keine
TestModal.tsx	Die TestModal-Komponente zeigt ein Modal mit den Details eines Habits an. Der Benutzer kann das Habit bearbeiten, löschen oder als abgeschlossen markieren.	habit, isExpanded, onClose, onDelet, showDeleteoptions, onDeleteSingle, onDeleteSeries, fetchHabits, onComplete
TimeTabele.tsx	Die Timetable-Komponente zeigt den Zeitplan der Habits an. Sie kann entweder die Habits für die nächsten 7 Tage oder die Habits für den aktuellen Tag anzeigen.	habits, onDeleteHabit, filter, nextHabitId, nextHabitDay, fetchHabits
FriendRequests.tsx	Die FriendRequests-Komponente zeigt die Freundschaftsanfragen des Benutzers an und ermöglicht es, diese anzunehmen oder abzulehnen.	keine
ExperienceBoard.tsx	Die ExperienceBoard-Komponente zeigt das Erfahrungsboard des Benutzers an, einschließlich des Profilbilds, des Benutzernamens und der gesammelten XP.	keine

7.4. Abhängigkeiten

Frontend-Abhängigkeiten:

- `express`: Basis-Framework für Routing und Middleware.
- `jsonwebtoken`: Token-Authentifizierung für Benutzer.
- `mongoose`: Integration und Modellierung der MongoDB-Datenbank.
- `cors`: Unterstützung für Cross-Origin-Anfragen.

Frontend-Abhängigkeiten:

- `react`: Hauptbibliothek für die Benutzeroberfläche.
- `redux`: State-Management-Bibliothek.
- `axios`: HTTPS-Client für die Kommunikation mit dem Backend.
- `react-bootstrap/CSS`: UI-Komponenten für konsistentes Styling.

7.5. Verwendete Tools

Entwicklung:

- **Visual Studio Code** als IDE.

Testing:

Backend:

- Unit-Tests mit **Jest.**
- **Postman** zum Testen der Routen.

Versionskontrolle:

- Nutzung von **GitLab** für die Repository-Verwaltung, CI/CD und Teamzusammenarbeit.

Deployment:

- Die Anwendung wird auf einer Cloud-Plattform **Heroku** deployed.

Datenbank:

- **MongoDB Compass** zur Datenbankverwaltung.

Kommunikation:

- **Discord**
- **WhatsApp**

Design:

- **Figma** für die Wireframes und Mockups
- **Adobe Firefly** für die Erstellung der Companions

7.6. Testumgebung

7.6.1 Jest

Jest als Testumgebung (Backend & Frontend)

Für das **Unit-Testing** von **Backend- und Frontend-Komponenten** wird **Jest** als Testframework verwendet.

- **Backend-Testing**: Jest wird genutzt, um die API-Endpunkte, Services und Datenbank-Interaktionen in **Node.js/Express** zu testen. Durch **Supertest** lassen sich HTTP-Anfragen simulieren und das Verhalten der API validieren.
- **Mocking & Isolation**: Abhängigkeiten wie **MongoDB** werden mit **MongoMemoryServer** oder Mock-Funktionen ersetzt, um unabhängige Tests zu gewährleisten.
- **Frontend-Testing**: React-Komponenten werden mit Jest und `react-testing-library` getestet, einschließlich Snapshot-Tests zur UI-Konsistenz.

7.6.2 Cypress

Cypress als Testumgebung (Frontend)

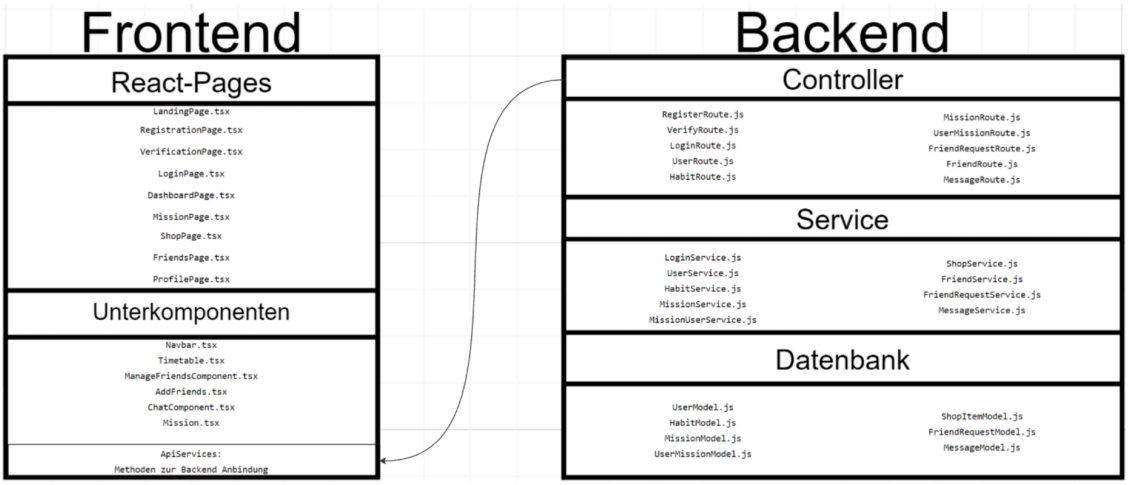
Für das End-to-End-Testing (E2E) wird **Cypress** genutzt, um den gesamten Nutzerfluss in **Habito** zu validieren. Cypress ermöglicht es, die Anwendung in einer echten Browserumgebung zu testen und Nutzerinteraktionen zu simulieren.

- **UI-Tests**: Prüfen, ob Buttons, Eingabefelder und Navigation wie erwartet funktionieren.
- **Automatisierte Testabläufe**: Nutzeraktionen wie das Erstellen und Abschließen von Habits werden simuliert und überprüft.

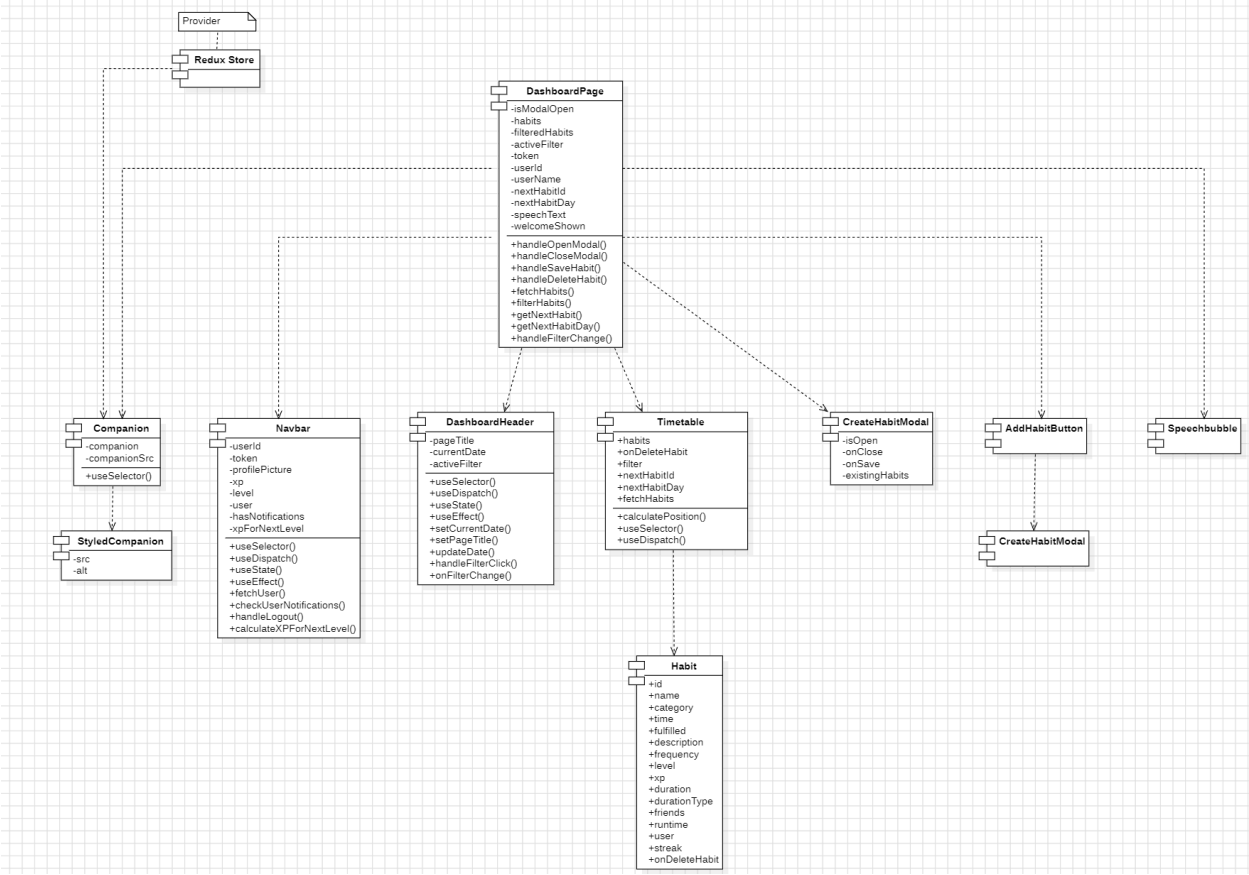
- **API-Tests**: Sicherstellung, dass die Kommunikation zwischen Frontend und Backend korrekt funktioniert.

7.7. Diagramme

Systemarchitektur Diagramm:



Komponenten Diagramm Dashboard:



8. Arbeitsprozess

8.1. Organisation

Kommunikation:

- Nutzung von Discord für Teamabsprachen und WhatsApp für spontane Kommunikation
- Erstellung eines eigenen Discord-Servers für Teamabsprachen und Austausch

Projektmanagement:

- Einsatz von GitLab für Code-Management und Aufgabenverteilung
- Klare Rollenverteilung (Backend, Frontend, Design, Testing)

Meetings:

- Regelmäßige Meetings zur Abstimmung, unterstützt durch spontane Treffen bei Bedarf (online)
- Meetings in der Hochschule Donnerstags

Teamstruktur:

- Sechs Teammitglieder mit klar definierten Aufgabenbereichen
- Zusammenarbeit fokussiert auf gute Abstimmung und schnelle Kommunikation

Feedback & Reviews:

- Geplant: Regelmäßige Code-Reviews und Feedback-Runden, um die Kommunikation klar zu halten

8.2. Herausforderungen

8.2.1. Lernerfolg(e)

- Routinierte Implementierung durch Erfahrung im Entwicklungsprozess
- Bessere vorausschauende Planung, um Probleme zukünftig frühzeitig zu erkennen und zu vermeiden
- Vertiefung von Best Practices
- Optimierte Projektstruktur und Dateiorganisation für bessere Übersichtlichkeit und Skalierbarkeit
- Komponentenbasierte Arbeitsweise, um Wiederverwendbarkeit zu erhöhen
- Ganzheitliches Verständnis der Applikation

8.2.2. Probleme

- Effiziente Zusammenarbeit war schwierig, da es das erste gemeinsame Projekt mit sechs Teammitgliedern war
- Schwierigkeit, mehrere Termine pro Woche für den Austausch zu finden
- Koordination und Abstimmung innerhalb des Teams war kompliziert (Sprintplanung & Tickets)
- Schwierigkeiten beim Deployment des Projekts

8.2.3. Geplante und umgesetzte Anforderungen

- Benutzerverwaltung: Registrierung und Login
- Soziale Funktionen: Freundesliste und Interaktionsmöglichkeiten
- Dashboard: Übersicht über eigene Habits und Missionsfortschritte
- Shop-System: Erwerb von Cosmetics mit verdienter In-Game-Währung
- Missionen & Gewohnheiten: Erstellung eigener Missionen und Habit-Tracking
- XP- & Geldsystem: Fortschritt und Belohnungen durch Aktivitäten
- Chat-Funktion: Direkte Kommunikation zwischen Nutzern
- Gamification-Elemente: Verschiedene Companions und spielerische Anreize