

# Abschlussprojekt Secure Software Engineering - Notes

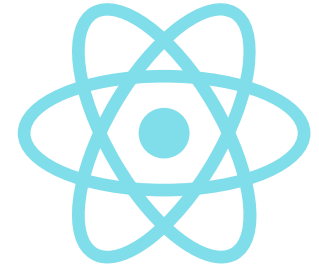
Von Clemens Horn, Till Kowoll und  
Nico Schneider

```
mirror_mod = modifier_ob.  
# Set mirror object to mirror  
mirror_mod.mirror_object =  
operation == "MIRROR_X":  
mirror_mod.use_x = True  
mirror_mod.use_y = False  
mirror_mod.use_z = False  
operation == "MIRROR_Y":  
mirror_mod.use_x = False  
mirror_mod.use_y = True  
mirror_mod.use_z = False  
operation == "MIRROR_Z":  
mirror_mod.use_x = False  
mirror_mod.use_y = False  
mirror_mod.use_z = True  
# Selection at the end -add  
mirror_ob.select= 1  
modifier_ob.select=1  
context.scene.objects.active  
("Selected" + str(modifier_ob.  
mirror_ob.select = 0  
= bpy.context.selected_object  
data.objects[one.name].select  
print("please select exactly  
-- OPERATOR CLASSES ----  
types.Operator):  
on X mirror to the selected  
object.mirror_mirror_x"  
mirror X"  
context):  
context.active_object is not
```

# Inhalt

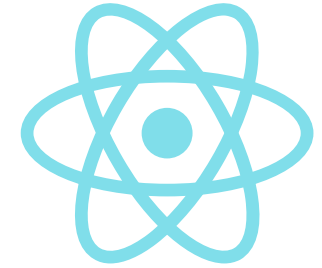
- Technologien
- Infrastruktur
- Funktionalitäten

# Technologien – React.js



- JavaScript Bibliothek für interaktive Benutzeroberflächen im Web
- Vorteile im Hinblick auf Sicherheit
  - Virtuelles DOM
    - React.js verwaltet Änderungen am DOM im Hintergrund, was dabei hilft riskante Manipulationen zu verhindern
    - Standardmäßiger Schutz vor XSS-Angriffen
    - Bei Verwendung von JSX werden Benutzereingaben korrekt behandelt und als Daten interpretiert
  - Komponentenbasierte Entwicklung
    - UI-Elemente werden in isolierte, wiederverwendbare Komponenten aufgeteilt
    - Fördert bewährte Sicherheitspraktiken -> Trennung von Zustand und Darstellung, Testen und Validieren einzelner Komponenten auf Sicherheitslücken

# Technologien – React.js



- Vorteile im Hinblick auf Sicherheit
  - Kontrollierter Datenfluss
    - In der Regel einwegiger Datenfluss
    - Bessere Kontrolle über den Zustand der Anwendung
    - Erschwert unerwünschte Manipulation von Daten oder nicht autorisierte Zugriffe
  - Unterstützung durch die Community
    - Große und aktive Entwickler-Community
    - Regelmäßige Sicherheitsprüfungen und Bugfixes
    - Sicherheitslücken können werden behoben bevor ernsthafter Schaden entsteht

# Technologien – Node.js



- Serverseitige JavaScript Laufzeitumgebung
- Ermöglicht JavaScript außerhalb des Browsers zu verwenden
- Vorteile im Hinblick auf Sicherheit
  - JavaScript-Sandbox:
    - Code wird in isolierter Umgebung ausgeführt
    - Hat keinen direkten Zugriff auf Betriebssystem oder andere Ressourcen
    - Potentiell schädlicher Code wird dadurch eingeschränkt
  - Aktive Community
    - Ständige Verbesserungen der Sicherheit durch regelmäßige Updates
  - Skalierbarkeit
    - Effiziente Verarbeitung von vielen gleichzeitigen Anfragen
    - Trägt dazu bei Angriffen, die auf Überlastung des Systems aus sind zu verhindern

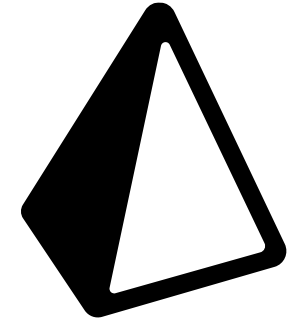
# Technologien – Nest.js

- Serverseitiges JavaScript Framework
- Basiert auf node.js
- Für serverseitige Webanwendungen, APIs und Microservices
- Vorteile im Hinblick auf Sicherheit
  - Dependency Injection
    - Abhängigkeiten können in Komponenten eingefügt werden
    - Authentifizierung und Autorisierung leichter zu implementieren
  - Middleware
    - Zwischen eingehendem Request und die ausgehende Response
    - Ermöglicht Implementierung von Sicherheitsmaßnahmen auf Anwendungsebene (bspw. Überprüfen von Parametern)
    - Setz

# Technologien – Nest.js

- Vorteile im Hinblick auf Sicherheit
  - Guarding
    - Guards ermöglichen Zugriff auf Routen und Endpunkte
    - Sicherheitsrelevante Aufgaben wie Überprüfen von Berechtigung oder Authentifizierung bevor Zugriff gewährt wird
  - Validierung
    - Validierung von Eingabedaten mithilfe von Pipes
    - Erleichtert Überprüfung der Datenintegrität und hilft potentielle Sicherheitslücken, wie unsichere Eingabewerte zu vermeiden
  - Skalierbarkeit
    - Ermöglicht Aufbau modularer, wiederverwendbarer Komponenten
    - Trennung von Verantwortlichkeit

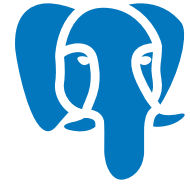
# Technologien - Prisma



- Datenbankorientierte ORM (Object-Relational Mapping) – Bibliothek
- Vereinfacht Entwicklung auf Datenbankanwendungen
- Erleichtert Zugriff auf Datenbanken
- Vorteile im Hinblick auf Sicherheit
  - Robuste Sicherheitsfunktionen
    - Integrierter Schutz vor den gängigsten Angriffen wie SQL-Injections, XSS, etc.
    - Datentyperstellung
    - Performanceoptimierung
    - Migrationen
  - Datenverschlüsselung
    - Sensible Informationen sind vor Unbefugten geschützt
  - Zugriffskontrolle
    - Man kann rollen und Berechtigungen definieren, um Zugriff zu beschränken

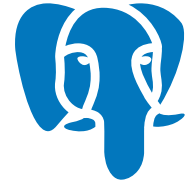


# Technologien - PostgreSQL



- Relationales Datenbankmanagementsystem
- Speicherung, Abfrage und Verwaltung von Daten
- Vorteile im Hinblick auf Sicherheit
  - Authentifizierung und Zugriffskontrolle
    - verschiedene Authentifizierungsmethoden, darunter Passwortauthentifizierung, SSL-Zertifikatsauthentifizierung und die Integration mit externen Authentifizierungssystemen
    - Benutzer können definierte Rollen und Berechtigungen verwenden, um den Zugriff auf Daten zu steuern und unbefugten Zugriff zu verhindern
  - Verschlüsselung
    - Auf mehreren Ebenen
    - kann die Übertragung von Daten über SSL (Secure Sockets Layer) verschlüsseln
    - Verschlüsselung von Daten auf Festplatten

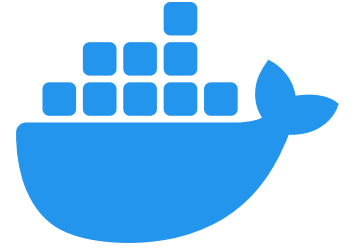
# Technologien - PostgreSQL



- Sicherheitsrichtlinien und Überwachung
  - ermöglicht es Administratoren, Sicherheitsrichtlinien zu implementieren
  - umfangreiche Überwachungsfunktionen, mit denen Administratoren Aktivitäten und Zugriffsversuche überwachen können
  - Protokolldateien erfassen Informationen wie fehlgeschlagene Anmeldeversuche und andere verdächtige Aktivitäten
- Erweiterte Sicherheitsfunktionen
  - Row-Level-Sicherheit, bei der feingranulare Zugriffsbeschränkungen auf Zeilenebene basierend auf bestimmten Kriterien definiert werden können
  - Dadurch können sensible Daten eingeschränkt und der Zugriff auf bestimmte Benutzer oder Rollen beschränkt werden
  - Sicherung und Wiederherstellung von Daten
- Aktive Community und regelmäßige Updates
  - große und aktive Community
  - Regelmäßige Updates und Patches werden veröffentlicht, um Sicherheitslücken zu beheben und potenzielle Bedrohungen anzugehen

# Technologien

- Docker
- Docker Compose





# Infrastruktur - Git

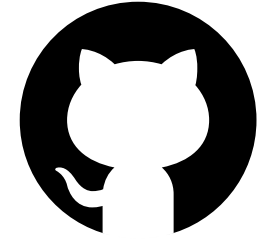
## Git-Flow

- Zwei Hauptbranches
- Main Branch
  - deployte und vollständig getestete und unterstützte Version der Anwendung
- Develop Branch
  - alle getesteten und neuen Features, die jedoch noch nicht releast wurden
- Feature Branches
  - neue Features werden isoliert und parallel von allen Teammitgliedern entwickelt und getestet werden
  - widmen sich jeweils einem GitHub-Issue
  - Nach Fertigstellung Pull Request auf Develop
  - Regelmäßige Integration des Develop in den Main

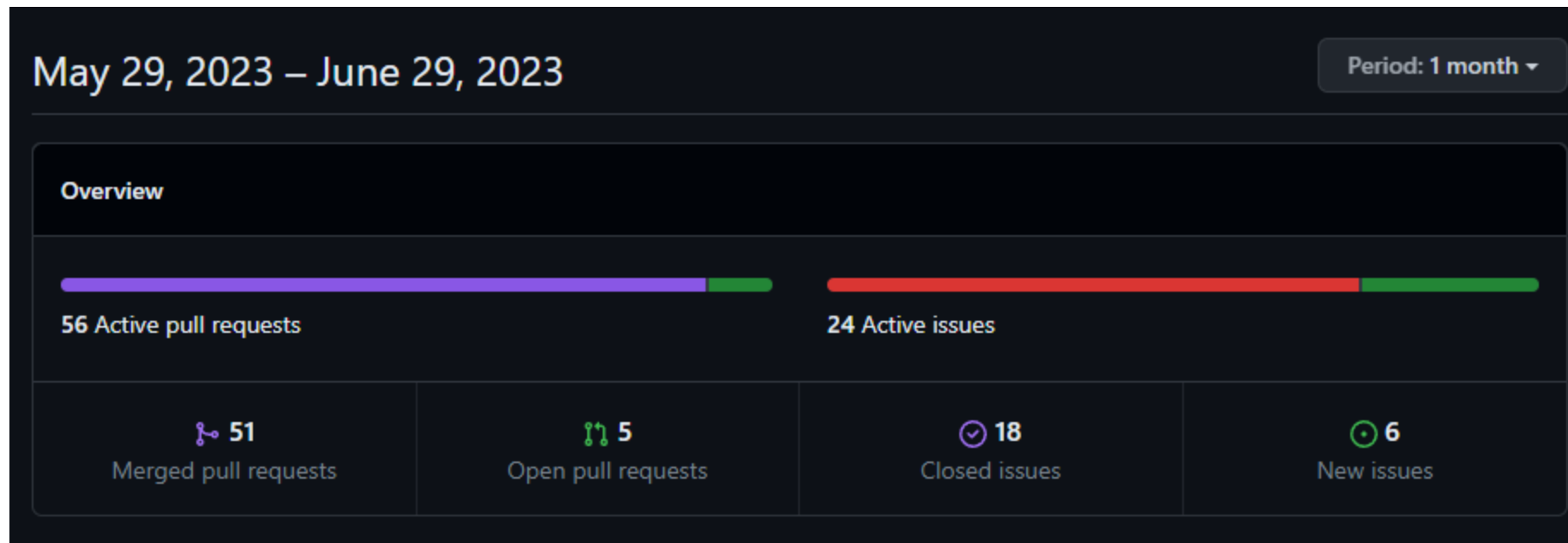
The screenshot displays the GitHub repository interface with a dark theme. It is organized into three main sections: 'Default branch', 'Your branches', and 'Active branches'. The 'Default branch' section shows the 'main' branch, which is the default and was updated 5 days ago by 'clemenscodes'. The 'Your branches' section lists two feature branches: 'feature/password-reset' (updated 9 hours ago) and 'feature/login' (updated 2 weeks ago), both by 'clemenscodes'. The 'Active branches' section shows the same two feature branches and adds the 'dependabot/npm\_and\_yarn/react-hook-form-7.45.1' branch, updated yesterday by 'dependabot[bot]'. Each branch entry includes a shield icon for protection status, the branch name in a blue pill, a fork icon, and the update information.

Section	Branch Name	Status	Updated	By
Default branch	main	Protected	Updated 5 days ago	clemenscodes
Your branches	feature/password-reset		Updated 9 hours ago	clemenscodes
	feature/login		Updated 2 weeks ago	clemenscodes
Active branches	feature/password-reset		Updated 9 hours ago	clemenscodes
	dependabot/npm_and_yarn/react-hook-form-7.45.1		Updated yesterday	dependabot[bot]

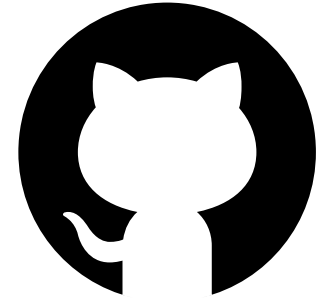
# Infrastruktur - Git



## Pull Requests und Issues



# Infrastruktur - Git



- Git CI/CD
  - GitHub-Actions
  - Pipeline bei Pull Request oder Push auf main
  - Code-Analyse -> Sicherheitsrisiken blockieren merge
  - Tests
  - Dependencies installiert und gecacht -> effizient und ressourcenschonend
  - Formatierung gemäß Prettier
  - alle internen Bibliotheken und Anwendungen mit Hilfe von Nx dezentral und parallel gelintet, gebaut und getestet
  - nur von Änderungen betroffene Projekte
  - Dependencies regelmäßig und automatisch auf Updates und Sicherheitslücken überprüft -> Dependabot
  - regelmäßig Pullrequests erstellt, die die Updates beinhalten

# Infrastruktur - Git

← Notes

✓ Feature/password reset #291

Summary

Jobs

- ✓ Set Agent Matrix
- ✓ Dependencies
- ✓ Notes App
- ✓ Agent 1
- ✓ Agent 2

Run details

- Usage
- Workflow file

Triggered via pull request 2 days ago

clemenscodes

 synchronize #136 `feature/password-reset`

Status

Success

Total duration

4m 16s

Artifacts

—

notes.yml

on: pull\_request

✓ Set Agent Matrix0s

✓ Dependencies45s

✓ Notes App3m 5s

Matrix: agents

- ✓ 2 jobs completed

Show all jobs

# Infrastruktur

- Betriebssysteme
  - Linux, Windows



- IDEs
  - VS Code und Webstorm
    - als Text-Editor verwendet und im Repository so konfiguriert, dass die verwendeten Erweiterungen und Einstellungen von allen Teammitgliedern automatisch synchronisiert werden können



- Bei Bedarf Discord für Treffen außerhalb der Uni





# Inhalt der Anwendung

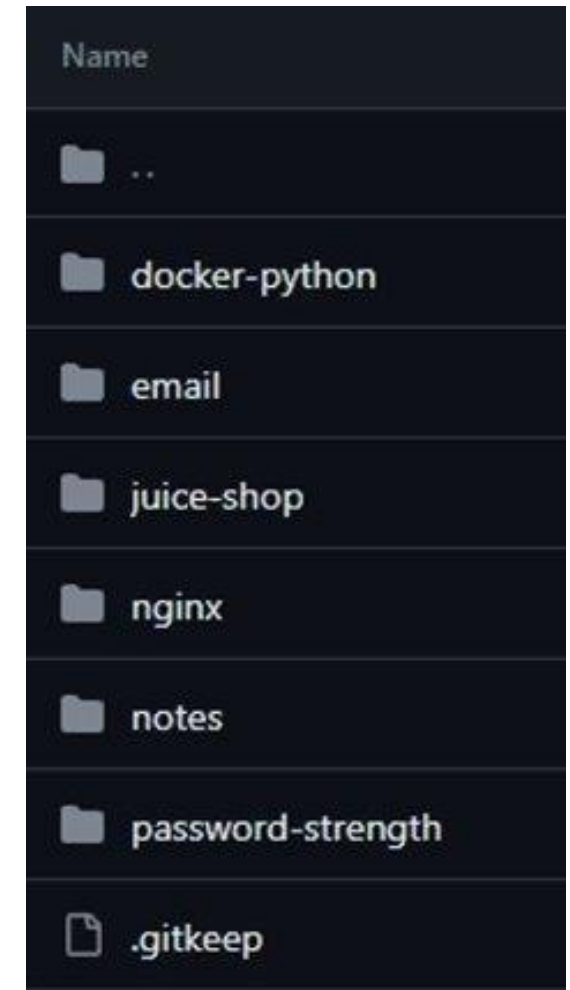
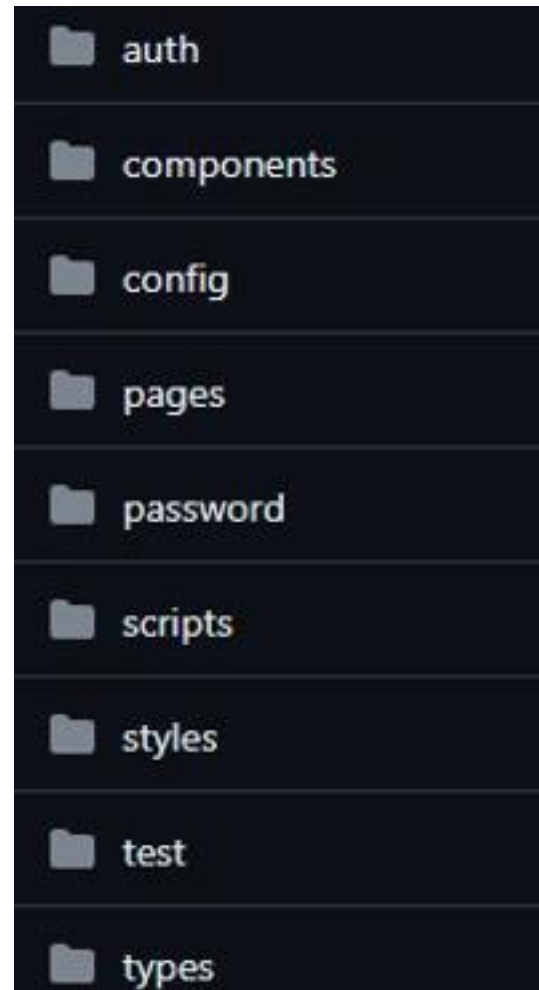
- Man kann sich registrieren und Notizen erstellen, die entweder öffentlich oder privat sind
- Die Notizen unterstützen die Verwendung von Markdown und HTML
- Den Notizen kann man ein YouTube-Video anhängen
- Man kann alle eigenen Notizen einsehen
- Öffentliche Notizen können über eine Suchfunktion gesucht werden
- Wenn man das Passwort vergessen haben sollte, kann man mit Angabe der E-Mail sein Passwort ändern

# Aufbau Projekt

- Nx Monorepo
- alle Aufgaben und Ressourcen des gesamten Moduls sind dort zu finden
- können über die Nx-CLI verwaltet werden
- Nx ermöglicht das einfache Importieren und Konsumieren von Bibliotheken
  - global registrierte Importpfade in der tsconfig.base.json
- Jedes Projekt wird durch Metadaten in der project.json beschrieben
  - teilen Nx mit, welcher Code zu der Bibliothek oder Anwendung gehört

# Aufbau Projekt

- Projekte befinden sich im Ordner apps
- Die internen Bibliotheken befinden sich im Ordner libs



# Funktionen - Registrierung

- Zuerst Validierung der Daten im Formular mittels zod
- Bibliothek um Datenstrukturen mittels TypeScript zu definieren und validieren
- Sicherstellung, dass Daten genau dem gewünschten Format entsprechen
- Wenn Daten invalide:
  - Formular in fehlerhaftem Zustand
  - Fehlermeldung im Client
  - Formular abschicken nicht möglich

Register


Enter all the relevant information.

Username

Email

Password

Confirm Password

 +

Already registered?

Login

# Registrierung - Datenvalidierung

- Eingabe von Benutzername, E-Mail und Passwort
- Diese Eingaben unterliegen folgenden Einschränkungen
- Benutzername:
  - zwischen 2 und 20 Zeichen
  - nur Groß,- und Kleinbuchstaben
  - nur Zahlen von 0 bis 9
  - Das einzig erlaubte Sonderzeichen ist ein Unterstrich
- E-Mail
  - Muss eine gültige E-Mail sein

# Registrierung - Datenvalidierung

- Passwort
  - mindestens 8 Zeichen
  - mindestens einen kleinen Buchstaben
  - mindestens einen großen Buchstaben
  - mindestens eine Zahl
  - mindestens ein Sonderzeichen
  - Muss die Überprüfung von zxcvbn-ts überstehen und den maximalen Sicherheitswert 4/4 erreichen
  - Muss bestätigt werden und übereinstimmen
- Serverseitige Validierung
  - Daten mit Hilfe von zod validiert
  - Regeln sind dabei die selben wie clientseitig
  - Falls Daten über reine HTTP-Anfrage kommen:
    - Statuscode von 406 und eine Erläuterung welche Datenformate erwartet wurden und warum die gesendeten Daten ungültig sind

# Registrierung– Erstellung neuer Benutzer

- mit Hilfe von Prisma wird ein neuer Nutzer angelegt
  - Passwort wird gehasht mit argon 2
  - zunächst wird aus einer Umgebungsvariable ein Secret ausgelesen
  - Danach werden zufällig 128 Bits generiert mit Hilfe der in Node.js eingebauten Methode randomBytes -> Salt
  - HMAC und dem SHA-512 Hashverfahren ein Secret generiert, das dann bei Argon2 dem vorher generierten Salt verwendet wird, um das Passwort zu hashen
  - Dieses Verfahren macht es unmöglich die Passwörter der Benutzer zu Bruteforcen, selbst bei Wissen über das Verwendete Hashverfahren.
  - Somit sind die Benutzeraccounts sicher.
  - Nach der Erstellung des Benutzers ist die Registrierung erfolgt und man wird automatisch eingeloggt durch das Loginverfahren.

# Funktionen - Anmeldung

## Validierung


- zod für die Datenvalidierung
- Es wird sich mit Benutzernamen und Passwort angemeldet
- Bei ungültiger Dateneingabe -> Fehlermeldung
- Bedingungen für die Datenvalidierung sind die identisch zur Registrierung für den Benutzernamen und das Passwort

## Überprüfung

- Nach Eingabe der gültigen Anmeldedaten wird das Passwort überprüft
- Gehasht mit selben Verfahren wie bei Registrierung
- Ist Hash identisch erfolgt Erstellung der Session

Login

Enter your username and password



[forgot password?](#)

[Not registered yet?](#)

Register



# Anmeldung – Erstellung der Session

- Session-ID mit Hilfe des UUIDv4-Verfahren erstellt
- garantiert zufällige, nicht-deterministische Session-IDs, um angemeldete Benutzer zu identifizieren
- in der Datenbank gespeichert und mit einem Benutzer assoziiert
- Standardmäßig ist eine Session eine Stunde gültig
- Zusätzlich zur Session wird ebenfalls eine Auffrischungssession erstellt und mit dem Benutzer assoziiert -> 30 Tage gültig
- Ist die Session abgelaufen, wird die Auffrischungssession verwendet, und eine neue Session erstellt
- Nach 30 Tagen muss sich der Benutzer erneut anmelden
- Um die Last der Datenbank zu minimieren, wird zusätzlich Session ein **JWT** erstellt und mit dem Benutzer assoziiert.

# Anmeldung– Sicherheitsmaßnahmen

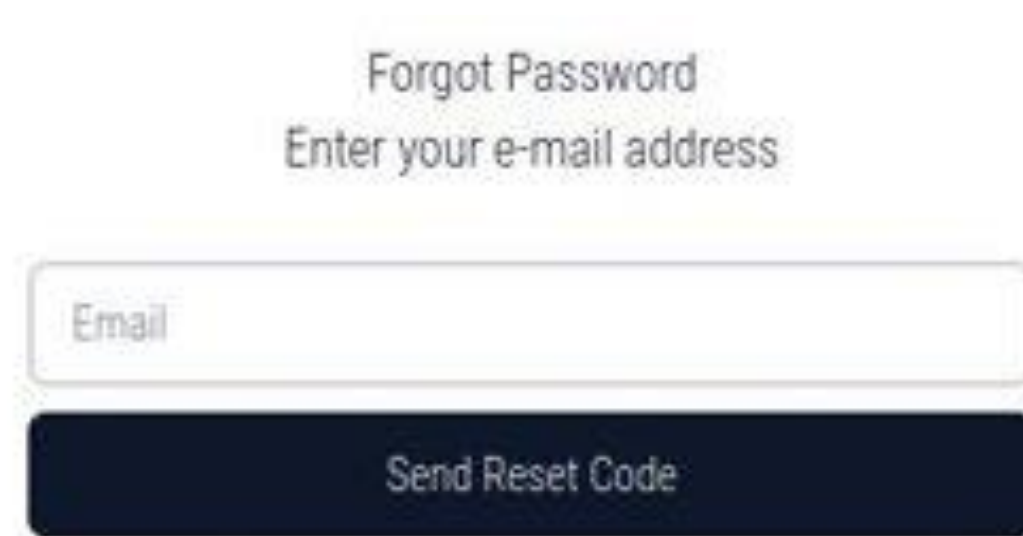
- Die Session-IDs werden als sichere und signierte HTTP-Only Cookies an das Frontend zurückgesendet
- Die enthalten die gleiche Gültigkeitsdauer
- Zusätzlich können die Cookies nur von der selben Domain aus versendet werden
- Das Signieren der Cookies erfolgt durch einen aus der Umgebung sicher ausgelesenen Schlüssel
- Die Cookies können nur mit HTTPS versendet werden
- Der JWT verwendet das Standardhashverfahren SHA-256
- Der JWT ist ebenfalls nur eine Stunde gültig
- Der JWT wird mit einem aus der Umgebung sicher ausgelesenen Schlüssel signiert

# Anmeldung – weiteres Vorgehen

- Cookies und JWTs werden automatisch beim Verwenden der Anwendung in nachfolgenden Anfragen an das Backend verwendet
  - sichere und dennoch angenehme Benutzung der Anwendung
- Läuft eine Session aus, bekommt der Nutzer davon nichts mit
- Auffrischungen der Sessions und der JWTs laufen automatisiert bei jeder Anfrage mit Hilfe von Axios Interceptors ab
- Der JWT und die Session werden im Client im LocalStorage gespeichert
- Loggt der Benutzer sich aus, werden alle Cookies und Sessions gelöscht und der LocalStorage komplett geleert
- Als angemeldeter Nutzer ist es nun nicht mehr möglich die Landingseite, Anmeldeseite oder Registrierungsseite aufzurufen
- Ohne gültige Session oder JWT kann man lediglich die vorher genannten Seiten aufrufen

## Funktionen – Passwort vergessen

- Wurde das Passwort vergessen, kann man in ein Formular eine E-Mail eingeben



Forgot Password  
Enter your e-mail address

Email

Send Reset Code

The image shows a 'Forgot Password' form. It has a title 'Forgot Password' and a subtitle 'Enter your e-mail address'. Below this is a text input field with the placeholder text 'Email'. At the bottom of the form is a dark blue button with the text 'Send Reset Code'.

## Funktionen – Passwort vergessen

- Auch dieses Formular ist mit zod abgesichert
- Hierbei gibt es keinerlei Informationen darüber, ob die E-Mail zum Benutzer gehört oder nicht, ob eine E-Mail gesendet werden konnte, zum Zurücksetzen des Passworts, oder nicht
- Existiert die E-Mail für einen Benutzer in der Datenbank, wird eine neue UUIDv4 generiert und mit dem Benutzer assoziiert
- Falls nicht, wird still ohne Fehlermeldung die Funktion beendet
- Die generierte UUIDv4 ist für nur fünf Minuten gültig
- Die E-Mail wird auf dem Server mit Hilfe von nodemailer versendet
- Der SMTP-Transporter ist Mailhog

# Funktionen – Passwort vergessen

- Das E-Mail-Postfach ist auf dem Port 8025 erreichbar
- Die E-Mail, die an die eingegebene E-Mail-Adresse im Erfolgsfall gesendet wurde, enthält nun einen Link mit der UUIDv4
- Somit kann nur der Benutzer, dem die E-Mail gehört und auf der Seite angemeldet ist, die Seite aufrufen
- Auf der aufgerufenen Seite befindet sich ein Formular zur Eingabe des neuen Passworts und zum Bestätigen
- Auch hier wird wie gewohnt mit zod validiert und sichergestellt, dass das Passwort sicher ist
- Sendet man das Formular erfolgreich ab, wird das Passwort für den Benutzer unter Verwendung des Passwortabsicherungsverfahrens geändert
- Im Erfolgsfall wird man nun auf die Anmeldeseite weitergeleitet und man kann sich mit dem neuen Passwort anmelden

# Funktionen - Autorisierung

- Erfolgt über Cookies und JWT Tokens
- Jegliche Anfragen an das Backend mit Ausnahme der Anmeldung, Registrierung und Passwort vergessen Funktionalität werden durch Autorisierungsguards geschützt
- Damit eine Anfrage an den Controller kommt, muss entweder ein Gültiger JWT, Session-Cookie oder Auffrischungs-Cookie mitgesendet werden
- Bei den Endpunkten in der Anwendung ist es möglich Administrationsendpunkte zu kennzeichnen, die nur von Administratoren aufgerufen werden können
- Zusätzlich gibt es rollenbasierte Autorisierung
- Nutzer haben die Rolle "USER"
- Administrationsfunktionalität wurde nicht implementiert

## Funktionen – Notiz erstellen

- Können mit Markdown oder HTML erstellt werden
- Es steht eine Vorschau der Notiz zur Verfügung



## Funktionen – Notiz erstellen

# My amazing note

☐ Public note

YouTube URL or video ID

You can attach a YouTube video to the note



# Notiz erstellen - Sicherheitsmaßnahmen

- Die Eingabe im Formular zum Erstellen der Notizen wird mit zod validiert
- Nur angemeldete Benutzer können Notizen erstellen
- Beim Erstellen wird die Benutzer-ID aus der Session entnommen und die Notiz mit dem Benutzer assoziiert
- Jeglicher Inhalt der Notiz wird zunächst mit der Bibliothek Showdown von Markdown zu HTML geparst
- Dabei besteht das Risiko für XSS, welches mit einer weiteren anschließenden Parsung durch isomorphic-dompurify bereinigt wird
- Sowie im Client als auch auf dem Server wird der Payload Pipes geparst bevor die Route zum Erstellen der Notizen überhaupt erst aufgerufen wird

# Notiz erstellen – YouTube Video als Anhang

- Eingabefeld für Link zu einem Video
  - transformiert die Eingabe nun mit Hilfe von zod zu einer YouTube-Video-ID
- YouTube-Video-ID an das Backend mitgesendet und dort strikt validiert
- Im Backend wird zusätzlich die YouTube-ID genommen und eine Anfrage an YouTube gesendet
  - Sicherstellen, dass es ein YouTube Video ist
- Dies birgt das Risiko für Serverside-Request-Forgery
  - Durch strikte Validierung nicht möglich
- Ist die Video-ID gültig, wird diese mit der Notiz assoziiert
- Im Client kann nun diese Video-ID aus der Notiz abgefragt werden und eine korrekte URL zu dem YouTube-Video konstruiert werden

## Notiz erstellen - Datenschutz

- Zwei – Klick Methode für das Einbinden von YouTube Videos
- Jedes Video muss mit einem Klick bestätigt werden
- Der Nutzer erklärt sich somit einverstanden
- Nach Einverständnis wird das Video in den DOM geladen

## Funktionen – Notiz suchen

- Aufzurufen über Strg + k oder Knopf in der Navigationsleiste
- In dem erscheinenden Eingabefeld kann nun ein Text eingegeben werden, um öffentliche Notizen zu suchen, die diesen Text beinhalten
- eingebener Suchbegriff wird im Client auf der Seite und in der URL als Query-Parameter angezeigt
- Sind keine Notizen mit dem Inhalt vorhanden, wird dies im Client angezeigt
- Ansonsten werden die Notizen in einer Liste angezeigt
- Bei Klick auf einer Notiz kann man sich die Notiz auf der Notizseite ansehen

Search results for: my

my amazing note

# Notiz suchen - Sicherheitsmaßnahmen

- Das Eingabefeld wird mit zod abgesichert
- Die Eingabe wird mit Reacts eingebauten Textparser zu einem String geparkt und im Client angezeigt
- Der Parameter der URL wird mit isomorphic-dompurify und showdown geparkt und gegen XSS abgesichert

# Quellen

- <https://icons8.com/icon/123603/react-native>
- <https://icons8.com/icon/54087/nodejs>
- <https://icons8.com/icon/aqb9SdV9P8oC/prisma-orm>
- <https://icons8.com/icon/3tC9EQumUAuq/github>
- <https://icons8.com/icon/32sNCVhNAX9Y/webstorm>
- <https://icons8.com/icon/17842/linux>
- <https://icons8.com/icon/tplcYSg4KMn0/windows-10>
- <https://icons8.com/icon/2mIguSGquJFz/discord-logo>
- <https://icons8.com/icon/cdYUIRaag9G9/docker>
- <https://icons8.com/icon/9OGlyU8hrxW5/visual-studio-code-2019>
- <https://icons8.com/icon/38561/postgresql>