Technische Hochschule Deggendorf, Fakultät für Angewandte Informatik

# Programmierpraktikum

# Nachhilfeverwaltung

**Dokumentation**

Bachelor Wirtschaftsinformatik mit Schwerpunkt Data Science

Das Team:

|  |  |  |
| --- | --- | --- |
| Vorname | Nachname | Matrikelnummer |
| Johannes | Zitzelsberger | 00813363 |
| Sandro | Podhorodeski | 00802329 |
| Sebastian | Rauch | 00819678 |

Betreuer: Prof. Dr. Andreas Fischer

Abgabedatum: January 19th 2023

## Inhaltsverzeichnis

*Insgesamt max. 20 Seiten*

Inhaltsverzeichnis

[Programmierpraktikum 1](file:////Users/Johannes/Library/Mobile%20Documents/com~apple~CloudDocs/Wirtschaftsinformatik/Semester%205/Programmierprojekt/Dokumentation%20Nachhilfeverwaltung.docx#_Toc124521843)

[Nachhilfeverwaltung 1](file:////Users/Johannes/Library/Mobile%20Documents/com~apple~CloudDocs/Wirtschaftsinformatik/Semester%205/Programmierprojekt/Dokumentation%20Nachhilfeverwaltung.docx#_Toc124521844)

[Inhaltsverzeichnis 2](#_Toc124521845)

[Übersicht 3](#_Toc124521846)

[Installation und Betrieb der Software 4](#_Toc124521847)

[Nutzerdokumentation 5](#_Toc124521848)

[Schüler: 5](#_Toc124521849)

[Lehrer: 7](#_Toc124521850)

[Admin: 8](#_Toc124521851)

[Über Uns 9](#_Toc124521852)

[Architektur der Software 10](#_Toc124521853)

[Aufgetretene und gelöste Schwierigkeiten 11](#_Toc124521854)

[Zusammenfassung 12](#_Toc124521855)

## Übersicht

* *Was macht Ihre Software?*
* *Max. 2 Seiten*

Die Software-Nachhilfeverwaltung bietet eine Plattform für Schüler, schnell und einfach an Nachhilfe-Kursen teilzunehmen und welche anzubieten. Um dies möglichst strukturiert zu organisieren, gibt es verschiedene Rollen im System.

**Schüler** hat die Möglichkeit Nachhilfe zu nehmen aber auch selbst zu geben. Diese „Fähigkeit“ wird ihm von der Rolle **Lehrer** erteilt. Die Rolle **Admin** sorgt für die Pflege aller User am System und der Anpassung von statischen Seiten.

Wenn sich ein Schüler/Lehrer mit seinem Username und Passwort anmeldet, wird er als erstes auf das Dashboard weitergeleitet. Dort sieht er seine bevorstehenden Nachhilfen, wie viele er bereits genommen und wie viel gegeben hat.

Im nächsten Tab „Nachhilfe nehmen“ kann der Schüler nun aus dem Angebot von anderen Schülern auswählen, welches Fach und zu welcher Uhrzeit er eine Nachhilfe buchen möchte. Falls Fragen oder Anmerkungen bestehen, kann mit einer Chat Funktion der Schüler angeschrieben werden, der die gewünschte Nachhilfe anbietet.

Falls die Fähigkeit vom Lehrer dem angemeldeten User zugewiesen ist, kann dieser im nächsten Tab „Nachhilfe geben“ auswählen. Hier sind dann Eigenschaften, wie zu welchem Fach, Preis und welcher Klasse er Nachhilfe für andere angeboten wird, anzugeben.

Bei „Profil verwalten“ können Name, Klasse und E-Mail vom Benutzer angepasst werden. Außerdem kann in dieser Maske sein Passwort geändert werden. Mit einer Logout Möglichkeit, wird der Schüler/Lehrer/Admin wieder zurück zum Login geleitet.

Wenn sich nun ein **Lehrer** anmeldet, erhält dieser einen weiteren Tab „Schülermanagement“. In diesem kann er nach Schüler suchen und bearbeiten, in welchen Fächern der ausgewählte Schüler Nachhilfe geben darf.

Der **Admin** hat Zugriff auf ein Dashboard, welches ihm ermöglicht neue Benutzer anzulegen und bestehende zu bearbeiten. Im zweiten Tab „Settings“ können für die statische Seite „Über Uns“, die dort angezeigten Texte „Impressum“ und „Datenschutz“ angepasst werden.

Die **API** bietet die Möglichkeit, ohne die Benutzeroberfläche Aktionen auszuführen. Dafür wurde ein „GET“, also das Abrufen von Daten und ein „POST“, was das Senden von Daten bedeutet, implementiert. Der GET liefert die Teilnehmer-Liste für den angemeldeten User und der POST fügt eine Nachhilfe mit den gewünschten Parametern hinzu.

## Installation und Betrieb der Software

* *Was sind die Voraussetzungen für den Betrieb?*

Voraussetzung zur Installation unseres Projekts sind Python, Pip und eine lauffähige IDE wie z.B. Visual Studio Code.

* *Wie wird die Software installiert?*

Um dann von GitLab die Software zu laden, müssen im Terminal folgende Befehle ausgeführt werden:

1. git clone https://mygit.th-deg.de/jz03363/programmierprojekt5

2. cd programmierprojekt5

3. cd Nachhilfeverwaltung

4. python3 -m venv venv

5. source venv/bin/activate

6. cd nachhilfeverwaltung

7. pip install -r requirements.txt

Als nächstes kann dann das Projekt in der IDE geöffnet werden. In der IDE sollen dann im nächsten Schritt im Terminal folgende Befehle wieder ausgeführt werden:

1. cd Nachhilfeverwaltung

2. python3 manage.py migrate

3. Testdaten einspielen:

python3 manage.py loaddata faecher\_data.json

python3 manage.py loaddata group\_data.json

python3 manage.py loaddata user\_data.json

python3 manage.py loaddata person\_detail\_data.json

python3 manage.py loaddata personen\_faecher\_data.json

python3 manage.py loaddata sender\_data.json

python3 manage.py loaddata settings\_content\_data.json

Zum Schluss kann dann der Server gestartet und die Software genutzt werden. Dies geschieht anhand dieses Befehls:

python3 manage.py runserver

* *Welche Zusatzpackages verwenden Sie, und warum?*

Bei der initialen Installation wird anhand des Files „requirements.txt“ folgende Zusatzpakete installiert:

1. asgiref==3.5.2 ()
2. Django==4.1.2 ()
3. Pillow==9.3.0 ()
4. sqlparse==0.4.3 ()
5. tzdata==2022.6 (aktuelles Datum und Zeit)
6. djangorestframework==3.14.0 ()
7. python-dateutil==2.2 ()

Unsere Nachhilfeverwaltungssoftware nutzt Tailwind um das Frontend zu gestalten. Tailwind ist ein CDN (Content Delivery Network), der das Gestalten von HTML basierten Projekten vereinfacht. Dieser Dienst wird in der Base HTML eingebunden. (Siehe Screenshot) *Ein Bild, das Text enthält.

Automatisch generierte Beschreibung*

Im Punkt 1 wird mittels Script Tailwind eingebunden. Flowbite.js wird in Punkt 2 eingebunden und bereits fertig gestylte Tailwind Objekte zu nutzen. Das Accordion welches im Chat verwendet wurde nutzt Flowbite. Der letzte Punkt dient zur Nutzung des Dark Modes.

* *Wie wird der Superuser angelegt? Wie werden neue Benutzer angelegt?*

In dem eigenen Admindashboard können Benutzer angelegt werden. Anleitung dazu kann man in dem Kapitel „Nutzerdokumentation“ einsehen. Superuser werden in unserem Projekt nicht benötigt. Diese muss man bei Bedarf über die Kommandozeile erstellen.

* *Was muss ein Administrator ggf. sonst noch beachten?*

Der Admin muss im Dashboard darauf achten, dass einem Schüler nicht direkt Fächer hinzugefügt werden können. Man muss lediglich einen Schüler zuerst anlegen und danach die benötigten Schulfächer auswählen. Das Vorgehen wird in dem Kapitel „Nutzerdokumentation“ beschrieben.

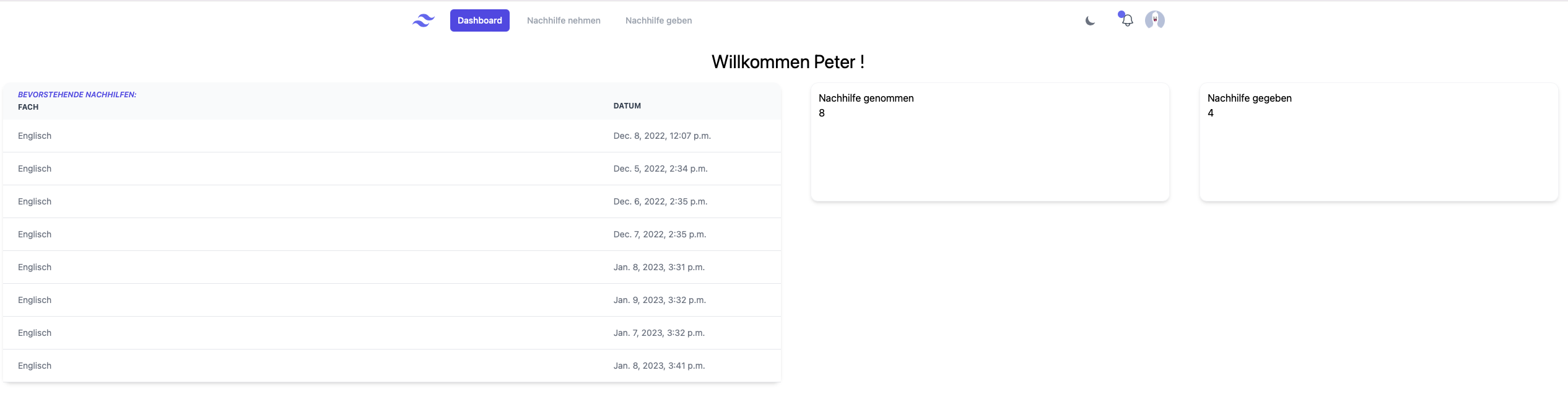
## Nutzerdokumentation

* *Wie wird die Software bedient?*
* *Erläuterung mittels Screenshots*

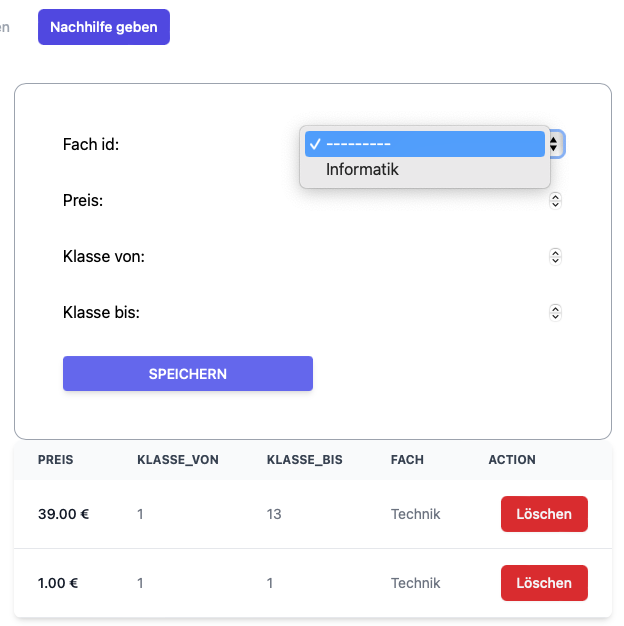
### Schüler:



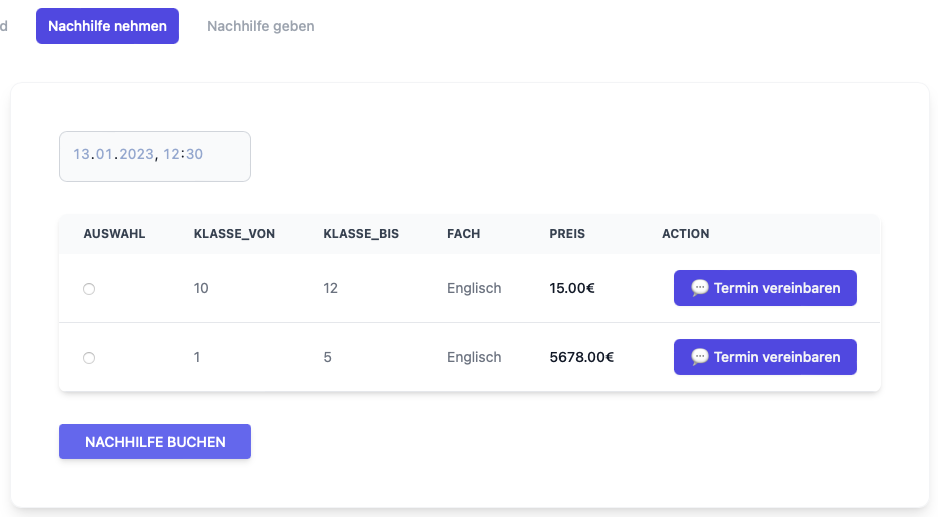
Nach dem Login erscheint das Dashboard:



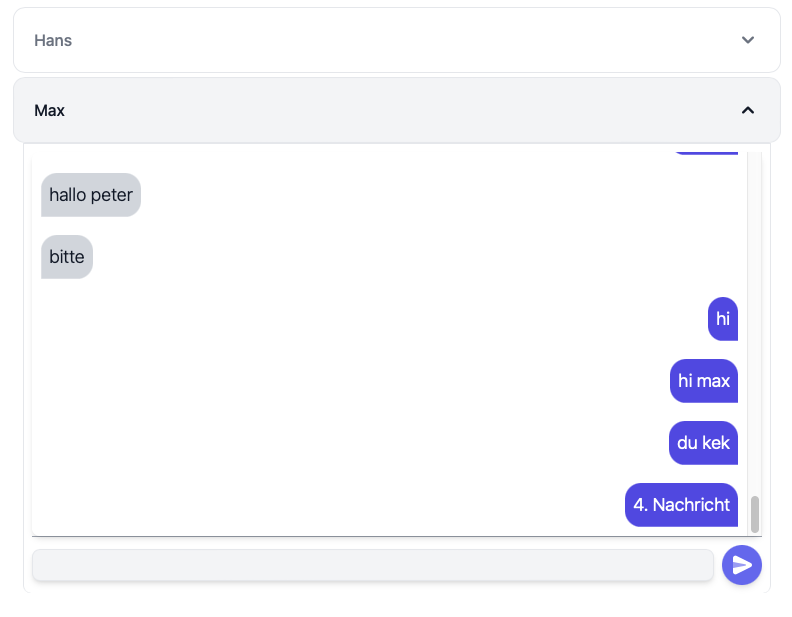
Ein Schüler kann eine Nachhilfe anbieten. Dafür wählt er ein Fach aus, definiert den gewünschten Preis und gibt an für welche Klassen diese Nachhilfe gilt:



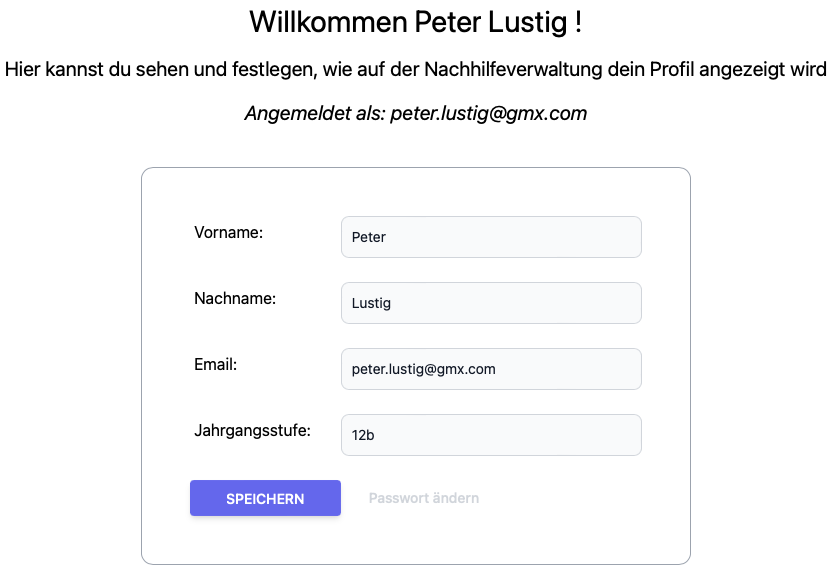
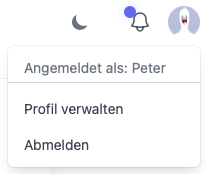
Außerdem besteht die Möglichkeit dann eine Nachhilfe zu nehmen. Hierfür muss ein Datum, die gewünschte Uhrzeit und eine passende Zeile per Radio Button ausgewählt werden. „Nachhilfe buchen“ schließt diesen Prozess ab. Falls der Schüler Rücksprache mit dem der die Nachhilfe anbietet haben möchte, kann er mit einen Klick auf „Termin vereinbaren“ zum Chat wechseln:



Möglichkeit über „Termin vereinbaren“ per Chat in Verbindung mit Schüler der Nachhilfe anbietet treten:

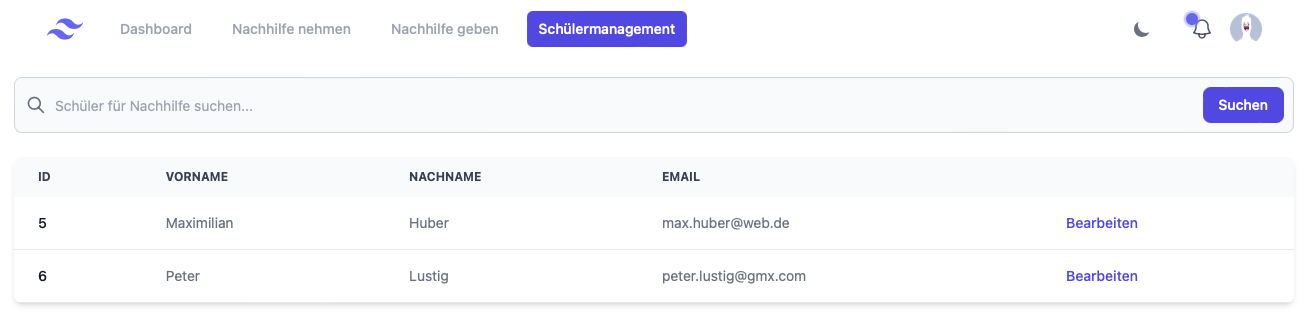


Profil verwalten:

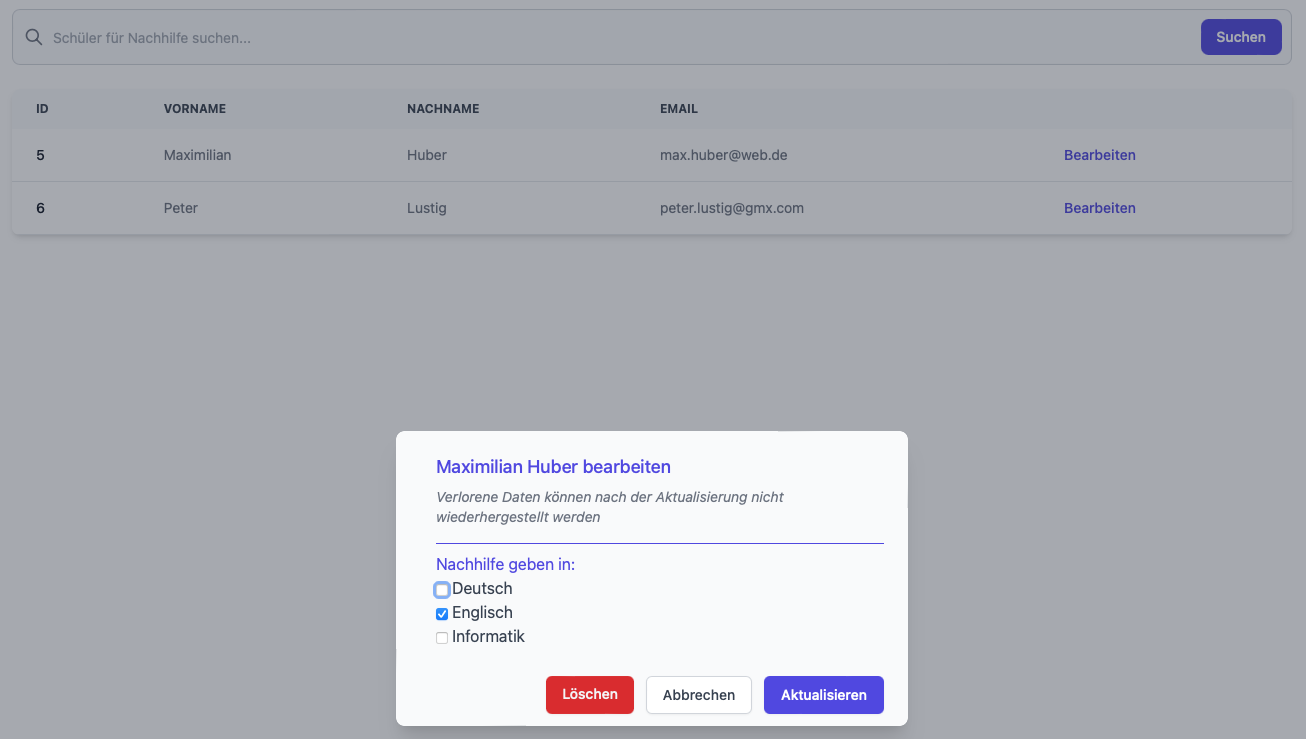


### Lehrer:

Alle Möglichkeiten und Funktionalitäten sind ähnlich zum Schüler. Was nun noch dazukommt ist das „Schülermanagement“:

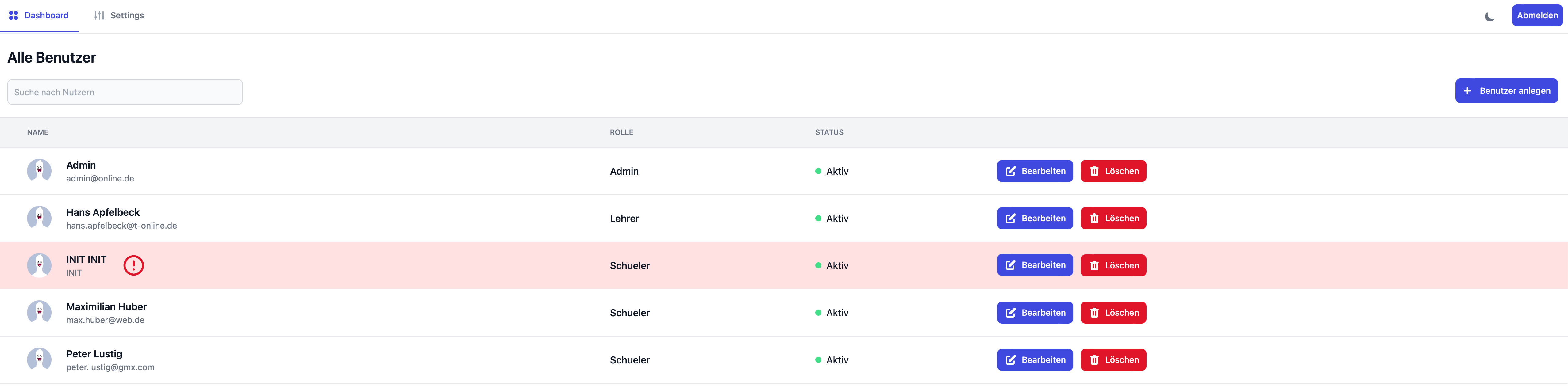


Hier kann ein Lehrer einen Schüler „Bearbeiten“ und zuweisen, für welches Fach er Nachhilfe geben darf:

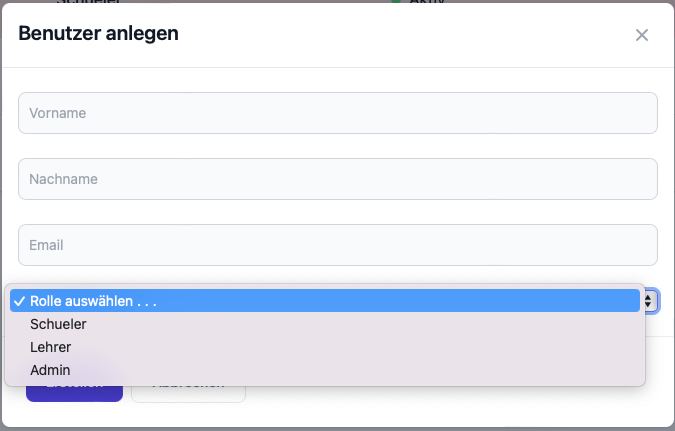


### Admin:

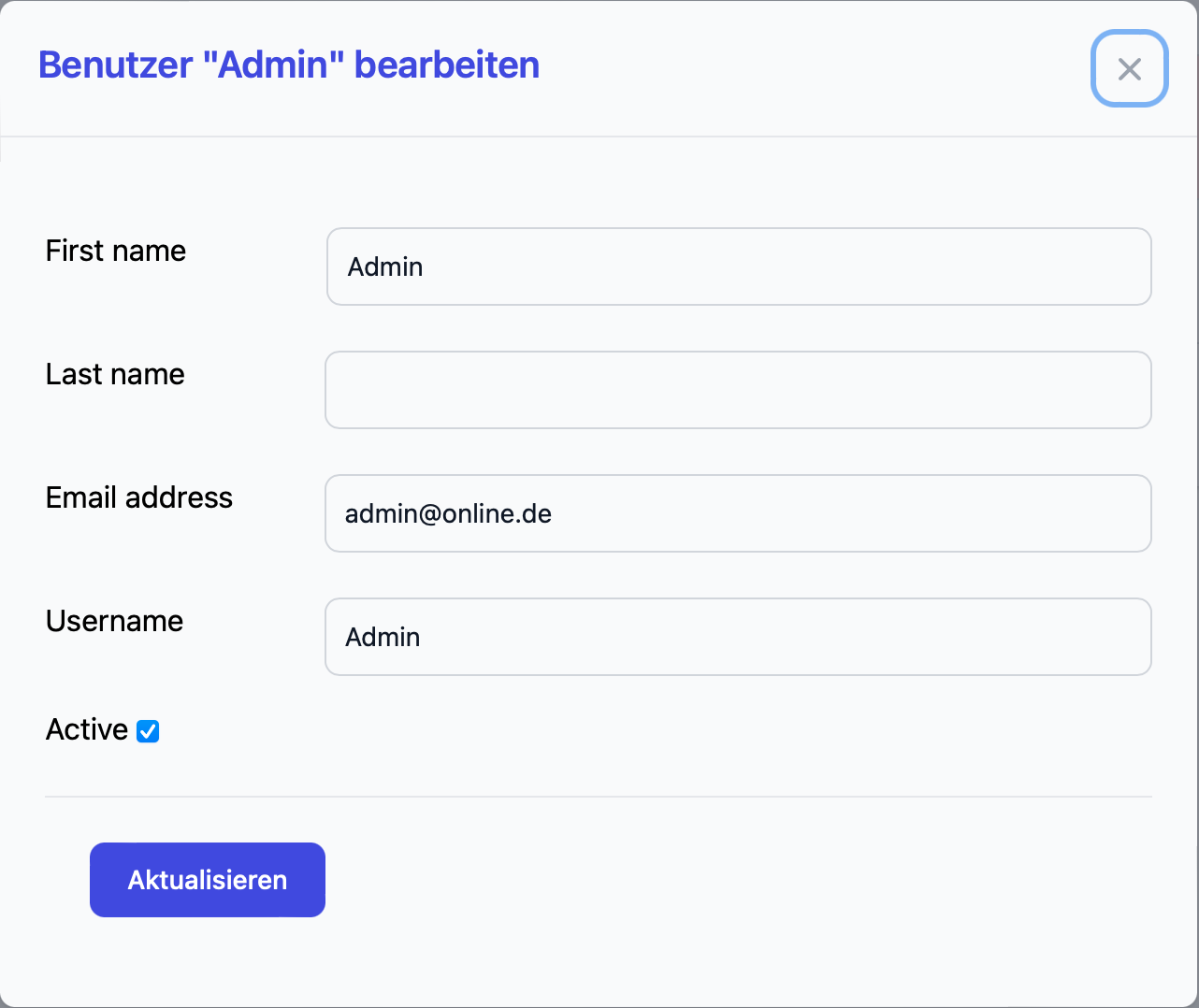
Der Admin meldet sich über die bereit bekannte Login-Seite mit seinen Zugangsdaten ein. Daraufhin erhält er sein eigenes Dashboard, wo alle User aufgelistet sind:



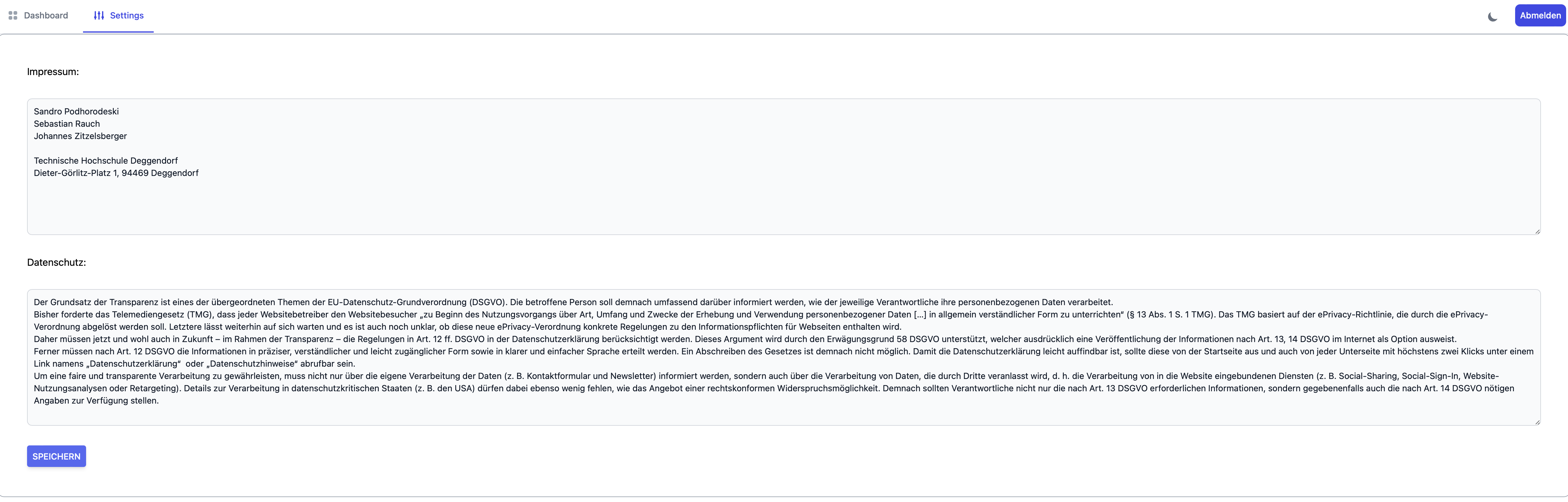
Mit „Benutzer anlegen“ kann ein neuer User mit gewissen Eigenschaften erstellt werden, bei Auswahl der Rolle Schüler muss noch eine Jahrgangsstufe hinzugefügt werden:



Bereits bestehende User können über „Bearbeiten“ angepasst werden:

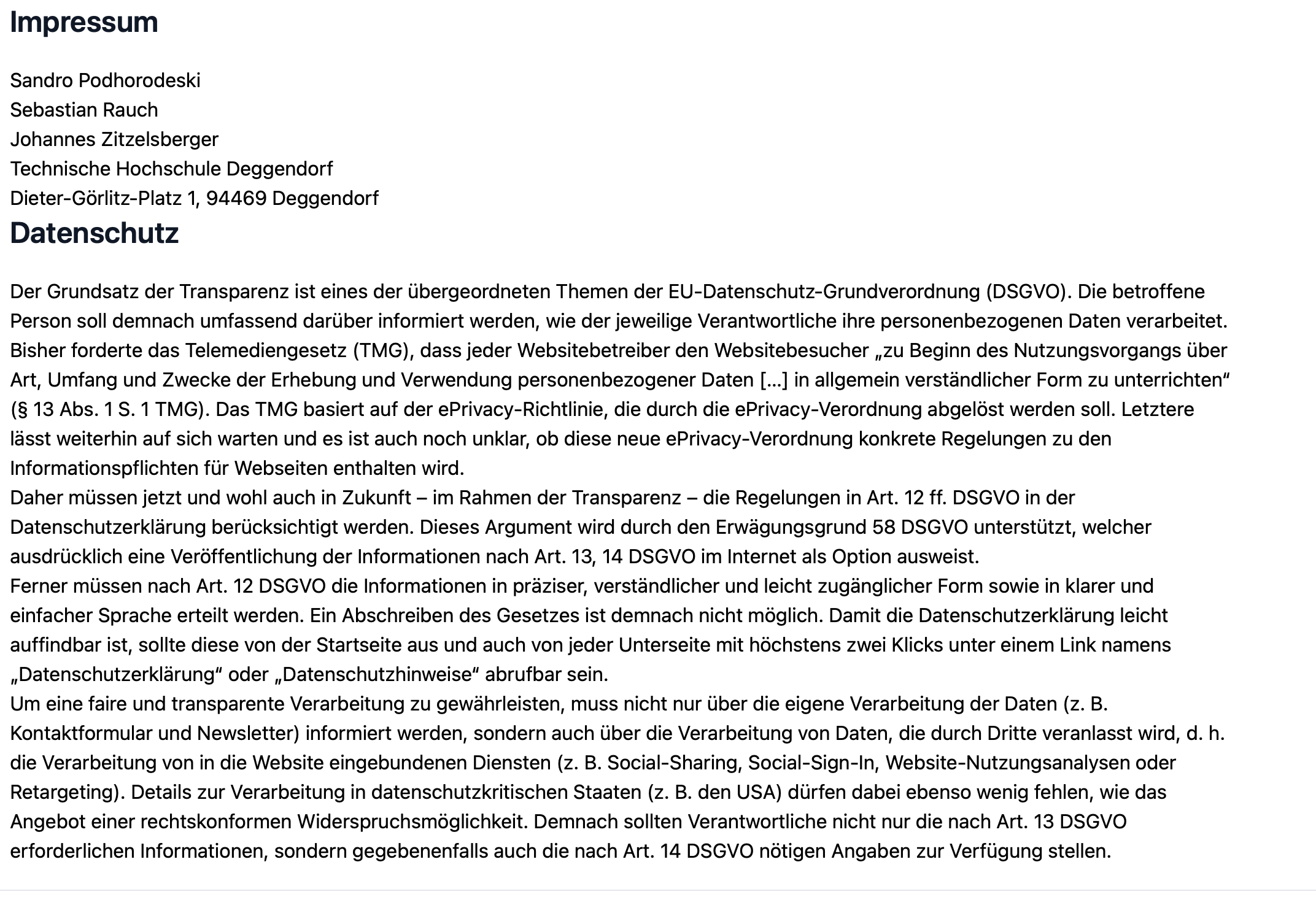


Im zweiten Tab ergibt sich für den Admin die Möglichkeit die Texte auf der statischen Seite „Über Uns“ zu ändern:



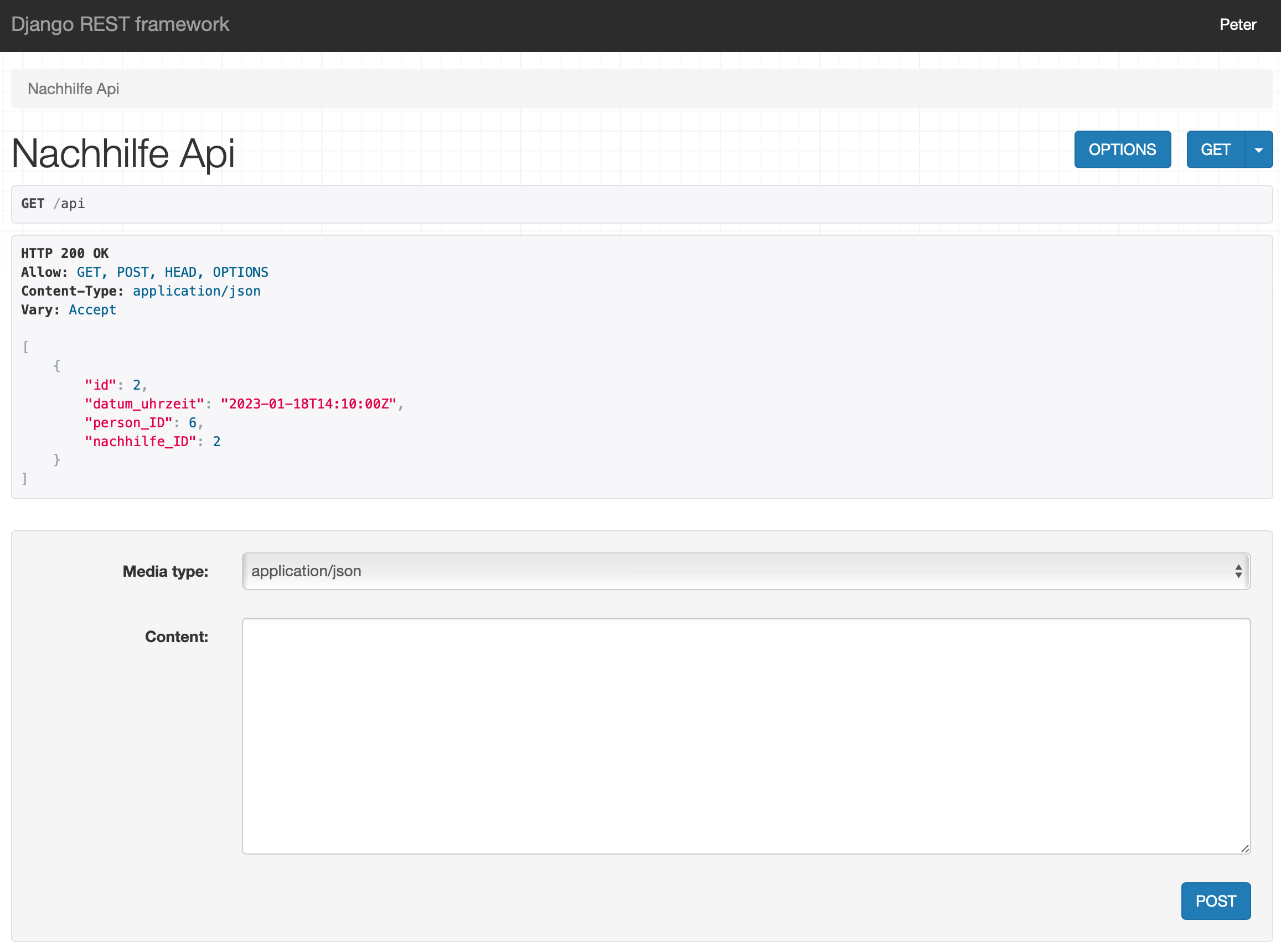
### Über Uns

Auf die „Über Uns“ Seite gelangt man von mit jedem User auf jeder Seite. Mit einem Klick am unter Rand auf „Über uns“ navigiert der Browser zur Seite. Auf dieser gibt es keine weiteren Interaktionsmöglichkeiten bis auf „Zurück“ was den Benutzer zur letzten besuchten Seite bringt:



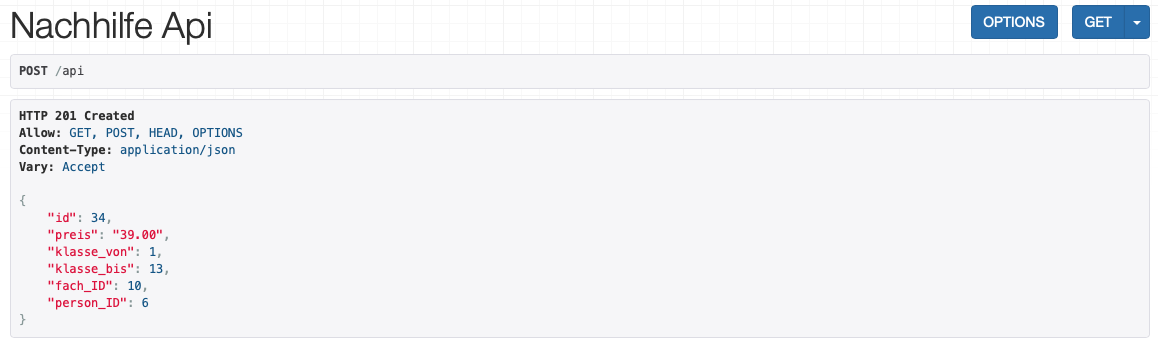
### API

Aufruf über: <http://127.0.0.1:8000/api>



Initial wird ein GET auf die Nachhilfe Teilnahme abgesetzt. Über folgendes .json kann ein Datensatz als Nachhilfe hinzugefügt werden:

{ "fach\_ID": 10, "preis": 39, "klasse\_von": 1, "klasse\_bis": 13, "person\_ID\_id": 5 }



## Architektur der Software

* *Wie wurde der Code organisiert?*

Der Quellcode wird mit Git, einem Versionskontrollsystem, organisiert, indem es in ein lokales Repository auf dem Entwickler-Computer eingecheckt und mit dem zentralen Repository auf einem Server synchronisiert wird. Jede Änderung des Quellcodes wird als Commit gespeichert, der eine Nachricht enthält, die die Änderungen beschreibt. Dies ermöglicht es, die Änderungen nachverfolgen und gegebenenfalls zu einem früheren Zustand des Quellcodes zurückkehren zu können. Branches können verwendet werden, um parallele Entwicklungsstränge zu verwalten, bevor sie in den Master-Branch integriert werden, allerdings wurde in dem Projekt aus Aufwandsgründen verzichtet. Git ermöglicht es auch, Änderungen von anderen Entwicklern zu akzeptieren oder abzulehnen und Konflikte beim Zusammenführen von Änderungen zu lösen.  
Zur Kommunikation und Organisation wurde Notion verwendet. Es wurde eine Notion-Seite als Hauptprojektseite erstellt und Unterseiten für jeden Aspekt des Projekts anlegt, wie z.B. Anforderungen, Design, Entwicklung und Testen. Innerhalb dieser Seiten gibt es Listen, um Aufgaben und Fortschritte zu verfolgen, sowie Datenbanken, um relevante Informationen wie Dokumentation, Ressourcen und Abhängigkeiten zu speichern.

Notion ermöglicht es auch, Dateien wie Dokumente, Bilder und Videos direkt in die Seiten einzubetten und mit Teammitgliedern zu teilen. Es gibt auch die Möglichkeit, Benachrichtigungen zu erstellen und zu verwalten, sowie Zugriffsrechte für bestimmte Teammitglieder zu definieren.

Durch die Verwendung von Schlagworten und Verknüpfungen wurden die Projektseiten vernetzt, die eine einfache Navigation und Übersicht über das Projekt ermöglichen.

* *Wie ordnet sich der Code in die Django-Umgebung ein, bzw. wo weichen Sie ggf. auch von Django-Konventionen ab?*

Django ist ein webbasiertes Framework, das auf der Konvention "Don't Repeat Yourself" (DRY) basiert. Dies bedeutet, dass Code so organisiert werden sollte, dass er wiederverwendbar ist und Redundanzen vermieden werden. Deshalb wurde die Geschäftslogik, welche wiederverwendbar ist in seperate .py Dateien ausgelagert und bei Bedarf importiert.

In Django werden Projekte und Anwendungen verwendet, um den Code zu organisieren. Dieses Projekt enthält nur eine Anwendung. Die Anwendung hat ihren eigenen Ordner mit Unterordnern für Modelle, Ansichten, Templates und Statische Dateien.

Modelle sind die Datenbank-Schemas, die in Django verwendet werden. Sie definieren die Struktur der Datenbanktabellen und die Beziehungen zwischen ihnen. Modelle befinden sich in der Datei "models.py" innerhalb der Anwendung.

Ansichten sind die logische Schicht, die die Daten aus den Modellen verarbeitet und sie an die Templates weitergibt. Sie befinden sich in einem Unterordner "views" innerhalb der Anwendung.

Templates sind die Schicht, die für die Darstellung der Daten verantwortlich ist. Sie befinden sich in einem Unterordner "templates" innerhalb der Anwendung und verwenden die Django Template Language (DTL). Dabei gibt es eine base.html, die eine Grundstruktur fast jeder Seite darstellt und von den anderen .html Dateien verwendet wird.

Statische Dateien wie Bilder, CSS und JavaScript befinden sich normalerweise einem Unterordner "static" innerhalb der Anwendung. Bei diesem Projekt wurde darauf verzichtet, da für das Styling das CSS Framework Tailwind benutzt wurde, welches über einen CDN eingebunden wurde.

Die URL-Verwaltung in Django erfolgt durch den sogenannten URL-Conf, in dem die URLs der Anwendung definiert werden und auf die entsprechenden Ansichten verweisen. Die URL-Conf wurde in der „setting.py“ deklariert.

Django folgt auch der Konvention, dass Dateinamen und Klassennamen in "CamelCase" geschrieben werden, während Variablen und Methoden in "snake\_case" geschrieben werden.

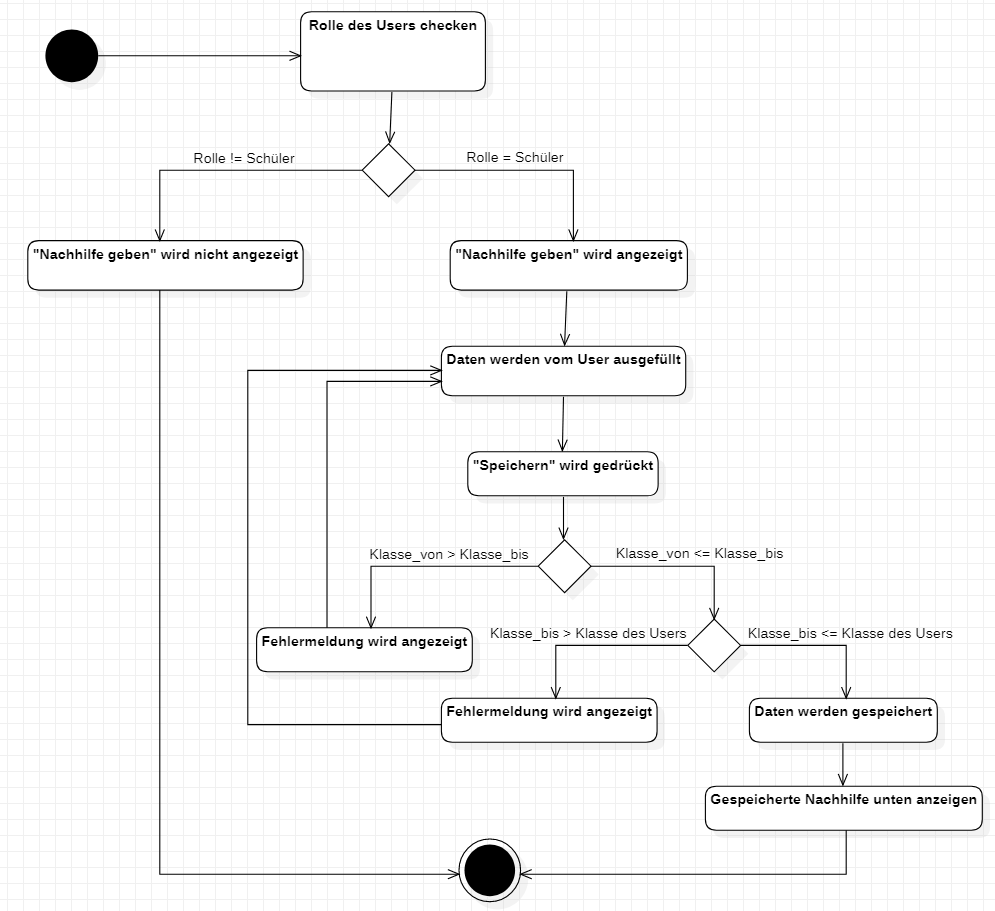
Insgesamt hilft die strikte Einhaltung dieser Konventionen und Ordnerstrukturen bei der Organisation und Wartbarkeit des Codes, erleichtert die Zusammenarbeit im Team und ermöglicht es, schnell und effektiv Änderungen vorzunehmen.

* *Was sind mögliche Anknüpfungspunkte, um den Code mit weiterer Funktionalität zu versehen?*

Es gibt viele Möglichkeiten, Django-Code mit weiterer Funktionalität zu versehen.

* Verwendung von Drittanbieter-Apps: Django bietet eine große Anzahl von Drittanbieter-Apps, die bereits viele häufig benötigte Funktionen wie Authentifizierung, Formulare, Datei-Uploads und vieles mehr bereitstellen. Eine weiter geplante Funktion wäre eine Drittanbieter-App, um Profilbilder zu ändern.
* Erstellung von benutzerdefinierten Anwendungen: Es wäre möglich, eigene Anwendungen zu erstellen, die speziell auf die Anforderungen des Projekts zugeschnitten sind.
* Erweiterung von bestehenden Modellen: Es wäre möglich, die bestehenden Modelle durch Hinzufügen von neuen Felder oder Methoden zu erweitern, um weitere Funktionalität bereitzustellen.
* Verwendung von Signalen: Django bietet die Möglichkeit, Signal-Handler zu erstellen, die auf bestimmte Ereignisse im System reagieren und entsprechende Aktionen ausführen können. Das wäre sinnvoll bei der Implementierung der Chat Funktion, um Sie im System agiler und zuverlässiger zu gestalten.
* Verwendung von Middleware: Django ermöglicht es, Middleware zu erstellen, die auf jede Anfrage angewendet wird und so weitere Funktionalitäten wie Authentifizierung, Berechtigungen und mehr bereitstellen kann.
* Erweiterte Verwendung des Django REST framework: Über das Rest Framework gibt es die Möglichkeit externe Systeme zu integrieren und Daten zu übertragen. Hier könnte man überlegen mit bestehenden Systemen einer Schule zu kommunizieren.
* *Min. 2 Architekturdiagramme mit Beschreibung (z.B. Sequenzdiagramm eines Vorgangs; Klassendiagramm; Flussdiagramm; …)*

UML Activity diagram zum Vorgang „Nachhilfe geben“:



TODO: Use Case Diagramm.

## Aufgetretene und gelöste Schwierigkeiten

* *Wo mussten Sie von Ihrem Plan abweichen?*
* *Welche unvorhergesehenen Probleme konnten Sie lösen?*
* *Welche zusätzlichen Eigenschaften haben Sie realisiert?*
* *Gibt es Probleme, die sich nicht lösen ließen bzw. ggf. mit zukünftigen Erweiterungen gelöst werden müssen?*

Der zeitliche Plan musste etwas angepasst werden. Als Ziel wurde sich die Fertigstellung noch vor Weihnachten gesetzt. Bug Fixes und kleinere Anpassungen schoben den Termin dann doch bis in den Januar.

Als zusätzliches Feature wurde der Chat entwickelt. Dieser wäre als Echtzeitanwendung zu aufwändig gewesen, aber mit dieser Umsetzung nun praktisch für den Anwender.

Unvorhergesehen Probleme waren beim erwähnten Chat, dass die Schlussendliche Lösung dann über das Schreiben von Tabellen umgesetzt wurde. Das schien Anfangs die beste Wahl, stellte sich dann aber als äußerst komplex heraus. Des Weiteren hat sich das Team anfangs für ein Java Script Framework entschieden, was schnell abgesetzt wurde, weil es zu viel Zeit in Anspruch genommen hat.

Durch das Verschieben der Fertigstellung, konnte bis zur Abgabe die Möglichkeit der Fächerpflege für Lehrer oder Admin nicht mehr implementiert werden. Dies muss aktuell mit einem Excel Import geschehen und sollte als zukünftige Erweiterung mit einer Anwendermaske umgesetzt werden. Somit kann der Lehrplan und somit die Fächer bequem vom Lehrer/Admin gepflegt werden.

## Zusammenfassung

* *Max. 1 Seite*

Nach der letzten Teamsitzung wurde das Projekt „Nachhilfeverwaltung“ abgeschlossen für Hr. Podhorodeski und Hr. Zitzelsberger. Der Kollege Hr. Rauch möchte die Anwendung weiter verfeinern und eventuell in einer Schule als Prototyp laufen lassen. Das würde die Möglichkeit bieten im „Produktivbetrieb“ die Performance, Usability und Stabilität der Anwendung zu testen.

Zusammenfassend lässt sich sagen, dass diese Praktische Studienarbeit eine gute Grundlage für die Python-Programmierung darstellt. So fällt es einfach, weitere und aufbauende Kurse oder Projekte in solch einem Umfeld zu meistern.