# Inhalt der Dokumentation

# Dokumentation Backend-Endpunkte Coderr

## Dokumentation für die Backend-Endpunkte zu Angeboten

Diese Dokumentation beschreibt die Endpunkte, um die Backend-Funktionalität passend zum Frontend zu entwickeln. Die folgende Beschreibung deckt die Struktur und Funktionalität der API-Endpunkte ab, basierend auf den definierten Views, Modellen und Serializern.

### Übersicht

Die API bietet folgende Endpunkte:

1. **GET /offers/** – Liste aller Angebote (Offers) mit Filter- und Suchmöglichkeiten.
2. **POST /offers/** – Erstellen eines neuen Angebots inklusive zugehöriger Details.
3. **GET /offers/{id}/** – Abrufen der Details eines spezifischen Angebots.
4. **PATCH /offers/{id}/** – Aktualisieren eines spezifischen Angebots.
5. **DELETE /offers/{id}/** – Löschen eines spezifischen Angebots UND den zugehörigen Angebotsdetails.
6. **GET /offerdetails/{id}/** – Abrufen der Details eines spezifischen Angebotsdetails.

### Endpunkte

#### 1. GET /offers/

**Beschreibung:**

Dieser Endpunkt gibt eine Liste von Angeboten zurück. Jedes Angebot enthält eine Übersicht der Angebotsdetails, den minimalen Preis und die kürzeste Lieferzeit.

**Anfragemethoden:**

GET

**Query-Parameter:**

**creator_id**: Filtert die Angebote nach dem Benutzer, der sie erstellt hat.
**min_price**: Filtert Angebote mit einem Mindestpreis.
**max_delivery_time**: Filtert Angebote, deren Lieferzeit kürzer oder gleich dem angegebenen Wert ist.
**ordering**: Sortiert die Angebote nach den Feldern “updated_at” oder “min_price”.

**search**: Durchsucht die Felder “title” und “description” nach Übereinstimmungen.
**page_size**: Gibt an, wie viele Ergebnisse pro Seite zurückgegeben werden sollen. Dies ist im Frontend in der config.js definiert, bitte setze die page_size in deiner Pagination genau auf den gleichen Wert. Dieser Query-Parameter wird nicht direkt genutzt.

**Paginierung**: Die Antwort von GET /offers/ ist nach PageNumberPagination paginiert.

**Beispiel für eine Anfrage:**

```
GET /offers/?creator_id=1&min_price=50&ordering=min_price&search=Website
```

**Antwort:**


```
[
  {
    "id": 1,
    "user": 1,
    "title": "Website Design",
    "image": null,
    "description": "Professionelles Website-Design...",
    "created_at": "2024-09-25T10:00:00Z",
    "updated_at": "2024-09-28T12:00:00Z",
    "details": [
      {"id": 1, "url": "/offerdetails/1/"},
      {"id": 2, "url": "/offerdetails/2/"},
      {"id": 3, "url": "/offerdetails/3/"}
    ],
    "min_price": 100.00,
    "min_delivery_time": 7,
    "user_details": {
      "first_name": "John",
      "last_name": "Doe",
      "username": "jdoe"
    }
  }
]
```


## 2. POST /offers/

**Beschreibung:**

Dieser Endpunkt ermöglicht es, ein neues Angebot (Offer) zu erstellen, das genau drei Angebotsdetails (OfferDetail) enthalten muss. Diese Details sollten die Typen basic, standard und premium abdecken.

**Validierung**:

Beim Erstellen eines Angebots müssen genau drei Details angegeben werden (und auch jeweils einmal  der "offer_type": basic, standard, premium).

Außerdem sollte alles vorhanden sein, außer ein “image”.

Die "revisions" sind Integer und fangen bei -1 an (die -1 ist der “unendlich Revisionen”-Fall).

Die "delivery_time_in_days" sind nur positive Integer.

Es sollte mindestens ein Feature drin sein.

**Anfragemethoden:**

POST

**Anfrage-Body (Beispiel):**

```
{
 "title": "Grafikdesign-Paket",
 "image": null,
 "description": "Ein umfassendes Grafikdesign-Paket für Unternehmen.",
 "details": [
  {
   "title": "Basic Design",
   "revisions": 2,
   "delivery_time_in_days": 5,
   "price": 100.00,
   "features": ["Logo Design", "Visitenkarte"],
   "offer_type": "basic"
  },
  {
   "title": "Standard Design",
   "revisions": 5,
   "delivery_time_in_days": 7,
   "price": 200.00,
   "features": ["Logo Design", "Visitenkarte", "Briefpapier"],
   "offer_type": "standard"
  },
  {
   "title": "Premium Design",
   "revisions": 10,
   "delivery_time_in_days": 10,
   "price": 500.00,
   "features": ["Logo Design", "Visitenkarte", "Briefpapier", "Flyer"],
   "offer_type": "premium"
  }
 ]
}
```

**Antwort (bei Erfolg):**


```
{
 "id": 1,
 "title": "Grafikdesign-Paket",
 "image": null,
 "description": "Ein umfassendes Grafikdesign-Paket für Unternehmen.",
 "details": [
  {
   "id": 1,
   "title": "Basic Design",
   "revisions": 2,
   "delivery_time_in_days": 5,
   "price": 100.00,
   "features": ["Logo Design", "Visitenkarte"],
   "offer_type": "basic"
  },
  {
   "id": 2,
   "title": "Standard Design",
   "revisions": 5,
   "delivery_time_in_days": 7,
   "price": 200.00,
   "features": ["Logo Design", "Visitenkarte", "Briefpapier"],
   "offer_type": "standard"
  },
  {
   "id": 3,
   "title": "Premium Design",
   "revisions": 10,
   "delivery_time_in_days": 10,
   "price": 500.00,
   "features": ["Logo Design", "Visitenkarte", "Briefpapier", "Flyer"],
   "offer_type": "premium"
  }
 ]
}
```


### 3. GET /offers/{id}/

**Beschreibung:**

Dieser Endpunkt gibt die Details eines spezifischen Angebots zurück.

**Anfragemethoden:**

GET

**URL-Parameter:**

id: Die ID des gewünschten Angebots.

**Beispiel für eine Anfrage:**

```
GET /offers/1/
```

**Antwort:**


```
{
 "id": 1,
 "user": 1,
 "title": "Grafikdesign-Paket",
 "image": null,
 "description": "Ein umfassendes Grafikdesign-Paket für Unternehmen.",
 "created_at": "2024-09-25T10:00:00Z",
 "updated_at": "2024-09-28T12:00:00Z",
 "details": [
  {
   "id": 1,
   "title": "Basic Design",
   "revisions": 2,
   "delivery_time_in_days": 5,
   "price": 100.00,
   "features": ["Logo Design", "Visitenkarte"],
   "offer_type": "basic"
  },
  {
   "id": 2,
   "title": "Standard Design",
   "revisions": 5,
   "delivery_time_in_days": 7,
   "price": 200.00,
   "features": ["Logo Design", "Visitenkarte", "Briefpapier"],
   "offer_type": "standard"
  }
 ],
 "min_price": 100.00,
 "min_delivery_time": 5,
 "user_details": {
  "first_name": "John",
  "last_name": "Doe",
  "username": "jdoe"
 }
}
```

### 4. PATCH /offers/{id}/

**Beschreibung:**

Aktualisiert ein spezifisches Angebot. Ein PATCH überschreibt nur die angegebenen Felder.

**Anfragemethoden:**

PATCH

**URL-Parameter:**

id: Die ID des zu aktualisierenden Angebots.

**Anfrage-Body (Beispiel für PATCH):**

```
{
  "title": "Updated Grafikdesign-Paket",
  "details": [
    {
      "title": "Basic Design Updated",
      "revisions": 3,
      "delivery_time_in_days": 6,
      "price": 120.00,
      "features": ["Logo Design", "Flyer"],
      "offer_type": "basic"
    }
  ]
}
```

**Antwort (bei Erfolg):**

```
{
  "id": 1,
  "title": "Updated Grafikdesign-Paket",
  "details": [
    {
      "id": 1,
      "title": "Basic Design Updated",
      "revisions": 3,
      "delivery_time_in_days": 6,
      "price": 120.00,
      "features": ["Logo Design", "Flyer"],
      "offer_type": "basic"
    }
  ]
}
```

### 5. DELETE /offers/{id}/

**Beschreibung:**

Löscht ein spezifisches Angebot.

**Anfragemethoden:**

DELETE

**URL-Parameter:**

id: Die ID des zu löschenden Angebots.

**Beispiel für eine Anfrage:**

```
DELETE /offers/1/
```

**Antwort (bei Erfolg) Hinweis, das ist nicht "**204 No Content"**:**

```
{}
```

### 6. GET /offerdetails/{id}/

**Beschreibung:**

Ruft die Details eines spezifischen Angebotsdetails ab.

**Anfragemethoden:**

GET

**URL-Parameter:**

id: Die ID des Angebotsdetails.

**Beispiel für eine Anfrage:**

```
GET /offerdetails/1/
```

**Antwort:**

```
{
  "id": 1,
  "title": "Basic Design",
  "revisions": 2,
  "delivery_time_in_days": 5,
  "price": 100.00,
  "features": ["Logo Design", "Visitenkarte"],
  "offer_type": "basic"
}
```

### 7. Allgemeine Permissions:

-Nur User die Authentifiziert und owner von dem Angebot sind (oder Admin) können dies löschen  oder bearbeiten.
-Nur Anbieter können Angebote erstellen

## Dokumentation für die Backend-Endpunkte zu Bestellungen

Diese API-Dokumentation beschreibt die Endpunkte, die für die Verwaltung von Bestellungen (Orders) implementiert sind. Die Endpunkte decken sowohl das Erstellen als auch das Abrufen, Aktualisieren und Löschen von Bestellungen ab. Zusätzlich wird die Struktur und Funktionalität der Serializer und Modelle erläutert, um ein besseres Verständnis der API-Interaktionen zu ermöglichen.

### API-Endpunkte

Die API bietet folgende Endpunkte für das **Order**-Modell:

1. **GET /orders/** – Liste der Bestellungen des angemeldeten Benutzers.
2. **POST /orders/** – Erstellen einer neuen Bestellung basierend auf einem Angebot (Offer).
3. **GET /orders/{id}/** – Abrufen der Details einer spezifischen Bestellung.
4. **PATCH /orders/{id}/** – Aktualisieren des Status einer spezifischen Bestellung.
5. **DELETE /orders/{id}/** – Löschen einer Bestellung (nur durch Admins).

### Endpunkte im Detail

#### 1. GET /orders/

**Beschreibung:**

Dieser Endpunkt gibt eine Liste der Bestellungen zurück, die entweder von dem Benutzer als Kunde oder als Geschäftspartner erstellt wurden.

**Anfragemethoden:**

GET

**Antwort (Beispiel):**

```
[
```

```
 {
  "id": 1,
  "customer_user": 1,
  "business_user": 2,
  "title": "Logo Design",
  "revisions": 3,
  "delivery_time_in_days": 5,
  "price": 150.00,
  "features": ["Logo Design", "Visitenkarten"],
  "offer_type": "basic",
  "status": "in_progress",
  "created_at": "2024-09-29T10:00:00Z",
  "updated_at": "2024-09-30T12:00:00Z"
 }
]
```


**Query-Parameter:**

Keine spezifischen Parameter.

**Hinweis:**

Nur Bestellungen, die vom angemeldeten Benutzer entweder als **Kunde** oder als **Geschäftspartner** erstellt wurden, werden zurückgegeben.

---

## 2. POST /orders/

**Beschreibung:**

Erstellen einer neuen Bestellung basierend auf den Details eines Angebots (OfferDetail).

**Anfragemethoden:**

POST

**Anfrage-Body (Beispiel):**


```
{
 "offer_detail_id": 1
}
```


**Antwort (bei Erfolg):**

```
{
 "id": 2,
 "customer_user": 1,
 "business_user": 3,
 "title": "Website Development",
 "revisions": 5,
 "delivery_time_in_days": 10,
 "price": 500.00,
 "features": ["Homepage", "Responsive Design"],
 "offer_type": "premium",
 "status": "in_progress",
 "created_at": "2024-09-30T12:30:00Z",
 "updated_at": "2024-09-30T12:30:00Z"
}
```


**Hinweis:**

Nur Benutzer mit einem **CustomerProfile** können Bestellungen erstellen. Der Benutzer gibt eine **OfferDetail ID** an, und die Bestellung wird auf der Grundlage dieses Angebots erstellt.

Beachte das die offer dennoch beinhalten muss wer der Anbieter und wer der Kunde ist, das kann aus der Auth und der Offer entnommen werden

---

### 3. GET /orders/{id}/

**Beschreibung:**

Abrufen der Details einer spezifischen Bestellung anhand der ID.

**Anfragemethoden:**

GET

**URL-Parameter:**

**id**: Die ID der zu abrufenden Bestellung.

**Beispiel für eine Anfrage:**

```
GET /orders/1/
```

**Antwort:**

```
{
 "id": 1,
 "customer_user": 1,
 "business_user": 2,
 "title": "Logo Design",
 "revisions": 3,
 "delivery_time_in_days": 5,
 "price": 150.00,
 "features": ["Logo Design", "Visitenkarten"],
 "offer_type": "basic",
 "status": "in_progress",
 "created_at": "2024-09-29T10:00:00Z",
 "updated_at": "2024-09-30T12:00:00Z"
}
```

### 4. PATCH /orders/{id}/

**Beschreibung:**

Aktualisieren des Status einer Bestellung (z.B. von "in_progress" zu "completed" oder "cancelled").

**Anfragemethoden:**

PATCH

**URL-Parameter:**

**id**: Die ID der zu aktualisierenden Bestellung.

**Anfrage-Body (Beispiel):**

```
{
 "status": "completed"
}
```

**Antwort (bei Erfolg):**

```
{
 "id": 1,
 "customer_user": 1,
 "business_user": 2,
 "title": "Logo Design",
 "revisions": 3,
 "delivery_time_in_days": 5,
 "price": 150.00,
 "features": ["Logo Design", "Visitenkarten"],
 "offer_type": "basic",
 "status": "completed",
 "created_at": "2024-09-29T10:00:00Z",
 "updated_at": "2024-09-30T15:00:00Z"
}
```


**Hinweis:**

Der Benutzer kann nur den Status aktualisieren.
Eine vollständige Aktualisierung der Bestellung (PUT) ist nicht erlaubt.

---

### 5. DELETE /orders/{id}/

**Beschreibung:**

Löschen einer spezifischen Bestellung. Nur Admin-Benutzer (Staff) dürfen Bestellungen löschen.

**Anfragemethoden:**

DELETE

**URL-Parameter:**

**id**: Die ID der zu löschenden Bestellung.

**Beispiel für eine Anfrage:**

```
DELETE /orders/1/
```

**Antwort (bei Erfolg), beachte ,dass das nicht “204 No Content” ist:**

```
{}
```

## Erweiterte API für das Zählen von Bestellungen

Diese Dokumentation beschreibt die zusätzlichen Endpunkte für das Zählen der

Bestellungen eines Geschäftsnutzers (Business User). Die API ermöglicht es, die Anzahl der **abgeschlossenen** und **laufenden** Bestellungen für einen bestimmten Geschäftsnutzer abzurufen.

## API-Endpunkte

Die API bietet zwei Endpunkte, die spezifisch für Geschäftsnutzer sind:

1. **GET /order-count/{business_user_id}/** – Gibt die Anzahl der **laufenden** Bestellungen eines Geschäftsnutzers zurück.
2. **GET /completed-order-count/{business_user_id}/** – Gibt die Anzahl der **abgeschlossenen** Bestellungen eines Geschäftsnutzers zurück.

## Endpunkte im Detail

### 1. GET /order-count/{business_user_id}/

**Beschreibung:**

Dieser Endpunkt gibt die Anzahl der **laufenden** Bestellungen eines bestimmten Geschäftsnutzers (Business User) zurück. Laufende Bestellungen sind solche mit dem Status in_progress.

**Anfragemethoden:**

GET

**URL-Parameter:**

**business_user_id**: Die ID des Geschäftsnutzers, dessen laufende Bestellungen gezählt werden sollen.

**Beispiel für eine Anfrage:**

```
GET /order-count/2/
```

**Antwort (Beispiel):**

```
{
  "order_count": 5
}
```

**Fehlerfälle:**

Wenn der Geschäftsnutzer nicht gefunden wird:

```
{
  "error": "Business user not found."
}
```

**Hinweis:**

Nur Bestellungen mit dem Status in_progress werden gezählt.

### 2. GET /completed-order-count/{business_user_id}/

**Beschreibung:**

Gibt die Anzahl der **abgeschlossenen** Bestellungen eines bestimmten Geschäftsnutzers zurück. Abgeschlossene Bestellungen haben den Status completed.

**Anfragemethoden:**

GET

**URL-Parameter:**

**business_user_id**: Die ID des Geschäftsnutzers, dessen laufende Bestellungen gezählt werden sollen.

**Beispiel für eine Anfrage:**

```
GET /completed-order-count/2/
```

**Antwort (Beispiel):**

```
{
  "completed_order_count": 10
}
```

**Fehlerfälle:**

Wenn der Geschäftsnutzer nicht gefunden wird:

```
{
  "error": "Business user not found."
}
```

**Hinweis:**

Nur Bestellungen mit dem Status in_progress werden gezählt.

### Zusammenfassung:

Beide Endpunkte ermöglichen es, die Anzahl von Bestellungen eines Geschäftsnutzers zu zählen, entweder laufende oder abgeschlossene. Diese Abfrage unterliegt keinen permissions.

## Dokumentation des API-Endpunkts für Basisinformationen

Dieser Endpunkt bietet allgemeine Statistiken über die Plattform, wie z. B. die Anzahl der Bewertungen, das durchschnittliche Bewertungsergebnis, die Anzahl der Geschäftsnutzer und die Anzahl der Angebote.

### 1. GET /base-info/

**Beschreibung:**

Ruft allgemeine Basisinformationen zur Plattform ab, einschließlich der Anzahl der Bewertungen, des durchschnittlichen Bewertungsergebnisses, der Anzahl der Geschäftsnutzer (Business Profile) und der Anzahl der Angebote.

**Anfragemethoden:**

GET

**URL-Parameter:**

Keine

**Beispiel für eine Anfrage:**

```
GET /base-info/
```

**Antwort (Beispiel):**

```
{
  "review_count": 10,
  "average_rating": 4.6,
  "business_profile_count": 45,
  "offer_count": 150,
}
```

**Felder:**

review_count: Die Gesamtzahl der Bewertungen auf der Plattform.
average_rating: Das durchschnittliche Bewertungsergebnis aller Bewertungen (Skala: 0 bis 5, auf eine Dezimalstelle gerundet).
business_profile_count: Die Anzahl der registrierten Geschäftsnutzer (Business Profile).
offer_count: Die Gesamtzahl der erstellten Angebote auf der Plattform.

## API-Endpunkte für Benutzerprofile

### 1. GET /profile/<int:pk>/

**Beschreibung:**

Ruft die detaillierten Informationen eines Benutzerprofils ab (sowohl für Kunden- als auch für Geschäftsnutzer). Ermöglicht auch das Bearbeiten der Profildaten (PATCH).

**Anfragemethoden:**

GET: Liefert die vollständigen Profildaten eines spezifischen Benutzers.
PATCH: Ermöglicht es einem Benutzer oder Admin, bestimmte Profilinformationen zu aktualisieren.

**URL-Parameter:**

pk: Die ID des Benutzers, dessen Profil abgerufen oder bearbeitet wird.

**Erfolgreiche Antwort auf GET (Beispiel):**

```
{
 "user": {
  "pk": 1,
  "username": "max_mustermann",
  "first_name": "Max",
  "last_name": "Mustermann"
 },
 "file": "profile_picture.jpg",
 "location": "Berlin",
 "tel": "123456789",
 "description": "Business description",
```

```
 "working_hours": "9-17",
 "type": "business",
 "email": "max@business.de",
 "created_at": "2023-01-01T12:00:00"
}
```

**Erfolgreiche Antwort auf PATCH (Beispiel):**

```
{
 "location": "Berlin",
 "tel": "123456789",
 "description": "Business description",
 "working_hours": "9-17",
 "type": "business",
 "email": "max@business.de"
}
```

**Statuscodes:**

200 OK: Erfolgreich abgerufen.
400 Bad Request: Ungültige oder fehlende Felder.
403 Forbidden: Berechtigungsfehler.
404 Not Found: Profil nicht gefunden.

## 2. GET /profiles/business/

**Beschreibung:**

Gibt eine Liste aller Geschäftsnutzer auf der Plattform zurück.

**Anfragemethoden:**

GET

**Beispiel für eine Anfrage:**

```
[
 {
  "user": {
   "pk": 1,
   "username": "max_business",
   "first_name": "Max",
   "last_name": "Mustermann"
  },
  "file": "profile_picture.jpg",
  "location": "Berlin",
  "tel": "123456789",
```

```
  "description": "Business description",
  "working_hours": "9-17",
  "type": "business"
 }
]
```

**Statuscodes:**

200 OK: Erfolgreich abgerufen.

### 3. GET /profiles/customer/

**Beschreibung:**

Gibt eine Liste aller Kundenprofile auf der Plattform zurück.

**Anfragemethoden:**

GET

**Beispiel für eine Anfrage:**

```
[
 {
  "user": {
   "pk": 2,
   "username": "customer_jane",
   "first_name": "Jane",
   "last_name": "Doe"
  },
  "file": "profile_picture_customer.jpg",
  "uploaded_at": "2023-09-15T09:00:00",
  "type": "customer"
 }
]
```

**Statuscodes:**

200 OK: Erfolgreich abgerufen.

### Zusammenfassung:

Die API bietet sowohl eine Detailansicht als auch eine Listenansicht für Benutzerprofile. Geschäftsnutzer und Kundenprofile können jeweils einzeln oder als Liste abgerufen werden, wobei sowohl GET als auch PUT/PATCH für das Bearbeiten von Profildaten unterstützt werden.

# Authentifizierungs- und Registrierungs-API

## 1. POST /login/

**Beschreibung:**

Authentifiziert einen Benutzer und liefert ein Authentifizierungs-Token zurück, das für weitere API-Anfragen genutzt wird.

**Anfragemethoden:**

POST

**Erforderliche Felder:**

username: Benutzername des Nutzers.
password: Passwort des Nutzers.

**Antwort:**

Erfolgreiche Authentifizierung gibt ein Token sowie Benutzerinformationen zurück.

```
{
 "token": "abcd1234",
 "username": "max_mustermann",
 "email": "max@beispiel.de",
 "user_id": 1
}
```

**Statuscodes:**

200 OK: Erfolgreiche Anmeldung.
400 Bad Request: Falsche Anmeldeinformationen oder ungültige Eingabe.

## 2. POST /registration/

**Beschreibung:**

Registriert einen neuen Benutzer und erstellt das entsprechende Benutzerprofil. Nach erfolgreicher Registrierung wird ein Authentifizierungs-Token zurückgegeben.

**Anfragemethoden:**

POST

**Erforderliche Felder:**

username: Benutzername des neuen Nutzers.
email: E-Mail-Adresse des neuen Nutzers.
password: Passwort für den neuen Benutzer.
repeated_password: Wiederholung des Passworts zur Bestätigung.

type: Profiltyp (Geschäfts- oder Kundenprofil).

**Antwort:**

Erfolgreiche Registrierung gibt ein Token sowie die Benutzerinformationen zurück.

**Antwort (Beispiel):**

```
{
  "token": "abcd1234",
  "username": "jane_doe",
  "email": "jane@beispiel.de",
  "user_id": 2
}
```

**Statuscodes:**

200 OK: Erfolgreiche Registrierung.
400 Bad Request: Ungültige Eingaben oder wenn z.B. die E-Mail bereits existiert.

### Zusammenfassung:

Diese API bietet Endpunkte für die Benutzeranmeldung und -registrierung. Durch den Login erhält der Benutzer ein Token für die Authentifizierung, und bei der Registrierung wird ein neuer Benutzer erstellt, wobei ein Kunden- oder Geschäftsnutzerprofil automatisch zugewiesen wird.

Um sicherzustellen, dass die Fehler von der API konsistent und benutzerfreundlich dargestellt werden, solltet ihr darauf achten, dass die zurückgegebenen Fehler eine klare Struktur haben, die eure Funktion extractErrorMessages korrekt verarbeiten kann. Hier sind ein paar Tipps, wie die API-Fehler aussehen sollten:

## API Fehlerstruktur:

### 1. Allgemeiner Fehleraufbau:

Fehler sollten als JSON-Objekte zurückgegeben werden. Das Objekt sollte Schlüssel-Werte-Paare enthalten, bei denen der Schlüssel das fehlerhafte Feld oder die Fehlermeldung ist und der Wert entweder eine Liste von Fehlern oder eine einzelne Fehlermeldung.

**Beispiel für eine fehlgeschlagene Validierung:**

```
{
```

```
  "username": ["Dieser Benutzername ist bereits vergeben."],
  "email": ["Diese E-Mail-Adresse wird bereits verwendet."],
}
```

### 2. Wenn mehrere Felder Fehler haben:

Jeder Fehler sollte in einer Liste von Nachrichten (Strings) sein, selbst wenn nur ein Fehler pro Feld vorhanden ist. So kann extractErrorMessages korrekt arbeiten.

**Beispiel für mehrere Feldfehler:**

```
{
  "email": ["E-Mail ist erforderlich.", "E-Mail-Format ist ungültig."],
  "password": ["Das Passwort ist nicht gleich mit dem wiederholten Passwort"]
}
```

### 3. Allgemeiner Fehler ohne spezifische Felder (z.B. Authentifizierungsfehler):

Falls ein allgemeiner Fehler auftritt, sollte eine Fehlermeldung als Liste unter einem allgemeinen Schlüssel wie detail oder non_field_errors zurückgegeben werden.

**Beispiel:**

```
{
  "detail": ["Falsche Anmeldedaten."]
}
```

Beachte, das nicht für jede Anfrage ein toasterror hinterlegt ist.