

# Einführung

## Hintergrund

Die meisten Kunden nutzen zur Verwaltung ihrer App die browserbasierte Bedienoberfläche von Appack – das Content-Management-System (CMS), auch „Portal“ genannt. Dies ist jedoch nur ein Teil der Appack-Plattform. Der technisch umfangreichere und für Entwickler besonders interessante Bereich befindet sich im „Backend“. Im Zuge unserer Modernisierungsmaßnahmen haben wir dem Backend eine Programmierschnittstelle (API) vorgeschaltet, die auch intern für die neu entwickelten Benutzeroberflächen verwendet wird.

## Was kann die API?

Grundsätzlich können Sie mit der API alles durchführen, was auch im CMS möglich ist. Besonders geeignet ist sie für bestimmte Anwendungsfälle (Use Cases), in denen sie ihre Stärken zeigt. Dazu gehören unter anderem:

- **Nutzer-Synchronisation:** Die API ermöglicht eine effiziente Synchronisation von Benutzerdaten zwischen verschiedenen Systemen oder Apps.
- **Push-Nachrichten versenden:** Mit der API lassen sich gezielt Push-Nachrichten an Nutzer oder Gruppen versenden.

Weitere Details und spezifische Anwendungsfälle finden Sie in der Dokumentation unter [UseCases](#).

# Verwendung

Die Appack Programmierschnittstelle ist technisch ein GraphQL Endpoint. Dies bedeutet, dass HTTP POST Requests mit einem JSON Payload gemäß der [GraphQL Spezifikation](#) gesendet werden müssen.

Der GraphQL Endpoint lautet: <https://api.appack.de/graphql>

Alle Aufrufe müssen authentifiziert sein, was durch ein Bearer Token erfolgt, das als Header mit jedem Request gesendet wird. Dieses Token muss dem JWT-Standard entsprechen, damit Informationen über den Aufrufer, seine Berechtigungen und die Gültigkeit des Tokens darin enthalten sind. Der initiale API-Token, den Sie erhalten, dient dazu, ein Access Token für API-Aufrufe zu erzeugen.

## Token-Typen

Token	Beschreibung
API-Token	Gültig so lange der API-Zugriff besteht, in der Regel so lange, wie die App betrieben wird. Es enthält Informationen darüber, wer im Kontext welcher App welche Operationen durchführen darf.
Access Token	Wird zur Authentifizierung von API-Aufrufen verwendet und ist nur 1 Stunde gültig. Es enthält Informationen über Benutzer, App-Kontext und Berechtigungen zum Zeitpunkt der Token-Erstellung.

## Beispiel: Access Token erzeugen

So erzeugen Sie mithilfe Ihres API-Tokens ein Access Token:

```
curl -g -X POST -H "Content-Type: application/json" \
  -H "Authorization: Bearer <API-Token>" \
  -d '{"query":{"getToken}}' \
  https://api.appack.de/graphql | jq '.data.getToken'
```

```

fetch("https://api.appack.de/graphql", {
  method: 'POST',
  headers: {
    'Authorization': 'Bearer <API-Token>',
    'Content-Type': 'application/json',
    'Accept': 'application/json'
  },
  body: JSON.stringify({query: "{getToken}"})
}).then(r => r.json()).then(response => {
  console.log(response.data)
})

```

```

public class GraphQLFetcher {
  public static void main(String[] args) throws Exception {
    OkHttpClient client = new OkHttpClient();
    RequestBody body = RequestBody.create("{\"query\":\"{getToken}\"}", MediaType
.get("application/json; charset=utf-8"));
    Request request = new Request.Builder()
      .url("https://api.appack.de/graphql")
      .post(body)
      .addHeader("Authorization", "Bearer <API-Token>")
      .addHeader("Accept", "application/json")
      .build();

    try (Response response = client.newCall(request).execute()) {
      if (response.body() != null) {
        System.out.println(response.body().string());
      }
    }
  }
}

```

```

<?php

$ch = curl_init("https://api.appack.de/graphql");
curl_setopt($ch, CURLOPT_POST, 1);
curl_setopt($ch, CURLOPT_POSTFIELDS, '{ "query": "{getToken}" }');
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
curl_setopt($ch, CURLOPT_HTTPHEADER, [
  'Authorization: Bearer <API-Token>',
  'Content-Type: application/json',
  'Accept: application/json'
]);

echo json_decode(curl_exec($ch), true)['data'];
curl_close($ch);

```

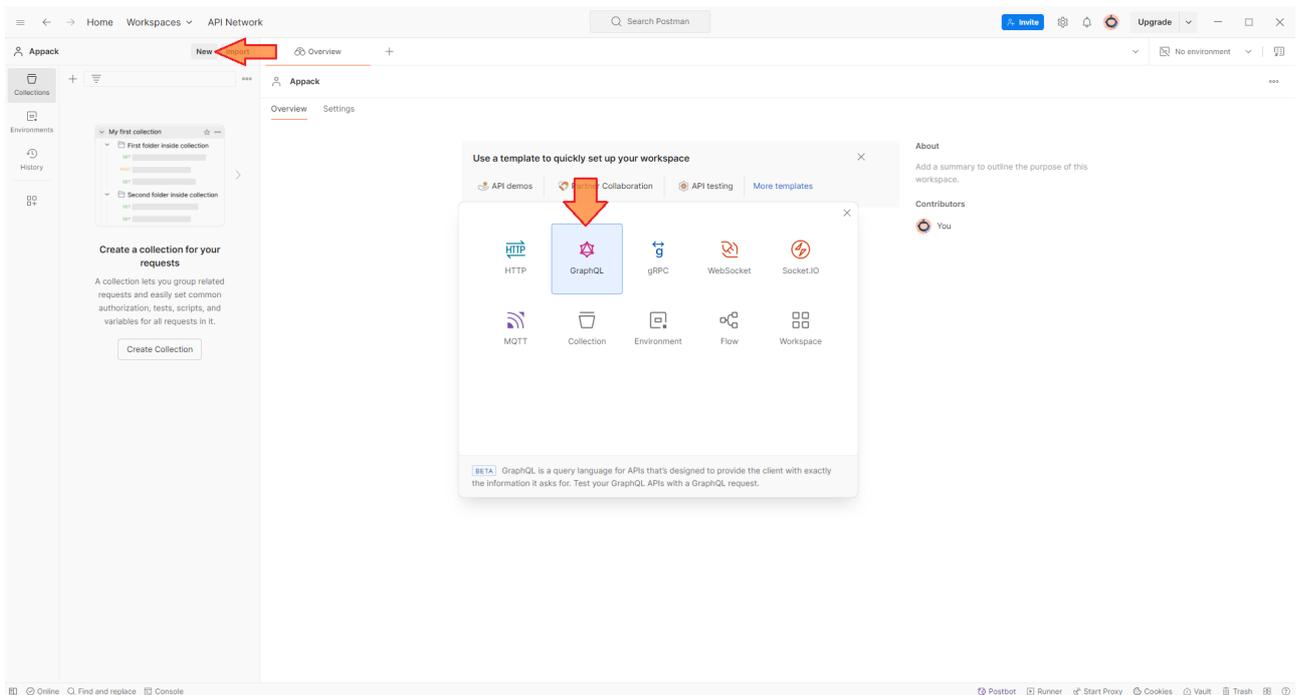
# API testen



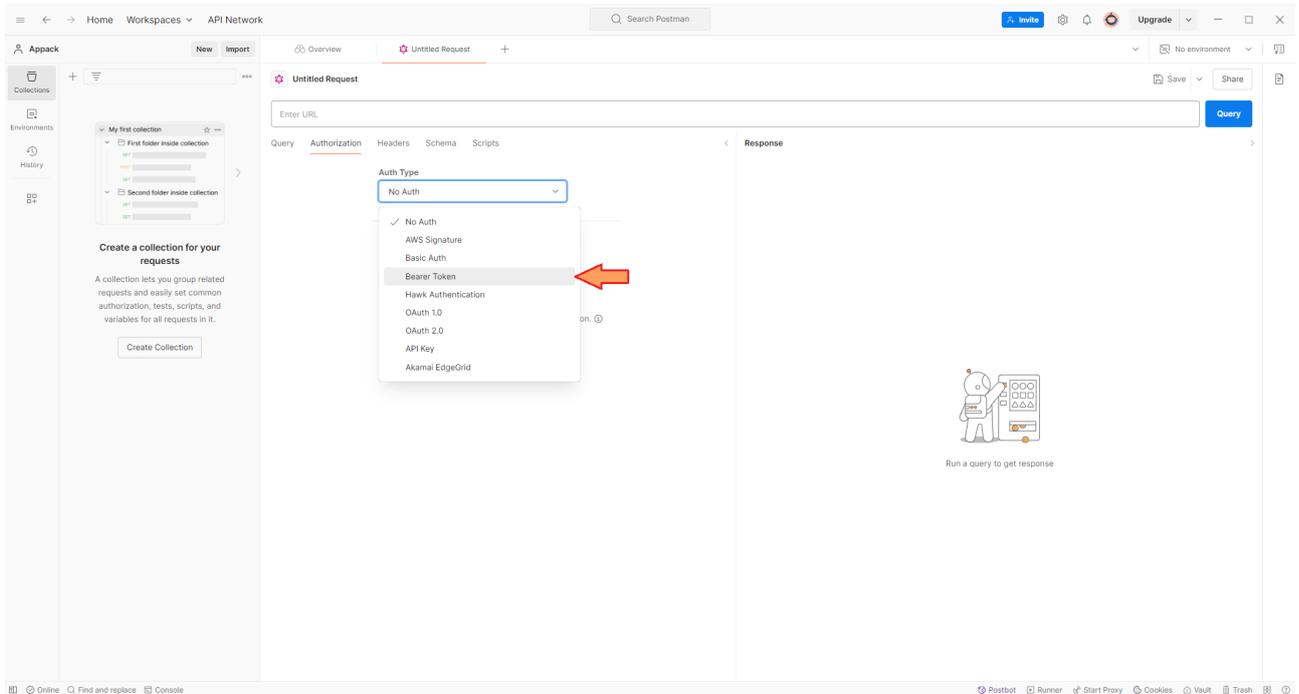
Mutations ändern direkt Inhalte im CMS. Achten Sie beim Testen von Funktionen wie dem Anlegen oder Bearbeiten von Benutzern darauf, dass keine unbeabsichtigten Änderungen vorgenommen werden, da diese nicht rückgängig gemacht werden können.

## Mit Postman

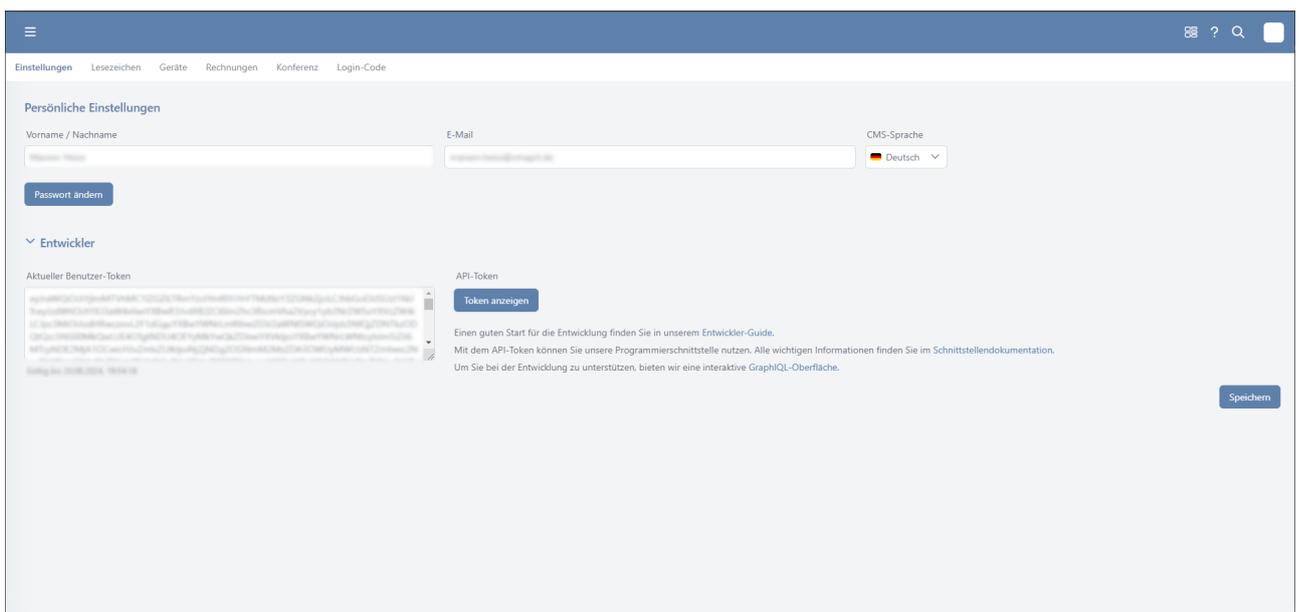
1. **Postman herunterladen**, installieren und **Einloggen**.
2. Klicken Sie oben links im Menü auf **New** und wählen Sie **GraphQL** aus.



3. Unter **Authorization** wählen Sie bei **Auth Type** die Option **Bearer Token** aus.



4. Geben Sie hier den API-Token ein, den Sie im CMS unter **Einstellungen** > **Entwickler** durch **Klick auf Token anzeigen** erhalten.



5. Tragen Sie unter **URL** die folgende Adresse ein: <https://api.appack.de/graphql>.
6. Um Ihr Access Token zu bekommen, führen Sie folgende Query aus:

```
query {
  getToken
}
```

7. Tauschen Sie nun den API-Token durch das erhaltene Access Token aus.
8. Klicken Sie anschließend unter **Schema** auf **Use GraphQL Introspection**, um das Schema zu laden.

Im **Query-Tab** sehen Sie nun alle verfügbaren Queries und Mutations. Sie können Funktionen auswählen, die Sie verwenden möchten, indem Sie den entsprechenden Haken setzen. Postman erstellt automatisch den passenden Query.

Für eine detailliertere Anleitung zu Postman können Sie folgende Seite besuchen:

[GraphQL mit Postman testen](#)

## Mit GraphiQL

Alternativ können Sie die API über die Web-Oberfläche testen, ohne Software herunterladen zu müssen. Gehen Sie einfach auf: <https://cdn.appack.de/graphiql/index.html>. Hier können Sie Queries und Mutations direkt im Browser testen, ohne viel konfigurieren zu müssen.

# Use Cases

Die Appack-Programmierschnittstelle (API) bietet vielseitige Möglichkeiten zur Integration, Verwaltung und Automatisierung von Funktionen innerhalb Ihrer App. Im Folgenden werden einige zentrale Anwendungsfälle (Use Cases) beschrieben, die aufzeigen, wie Sie die API effektiv nutzen können, um Benutzer zu synchronisieren, Termine zu verwalten, Push-Nachrichten zu versenden und mehr.

## Gruppen

Das Konzept einer **Gruppe** bei Appack dient dazu, eine Menge von Personen zu definieren, beispielsweise eine Mannschaft. Besonders praktisch ist, dass Gruppen auch dynamisch gebildet werden können, indem sie auf Rollen basieren, etwa alle Benutzer mit bestimmten Rollen. Diese Flexibilität reduziert den Verwaltungsaufwand erheblich. So kann ein Verein beispielsweise festlegen, dass nur bestimmte Benutzergruppen, wie Trainer und Vorstandsmitglieder, Push-Benachrichtigungen versenden dürfen, indem sie diese Rollen der Systemgruppe "Push-Administratoren" zuordnen. Dies ermöglicht eine effiziente Verwaltung und Zuweisung von Rechten. Wie Sie Gruppen über die API verwalten, erfahren Sie unter [Gruppen](#).

## Mediathek

Mit der **Mediathek**-Funktion können Sie multimediale Inhalte wie Videos, Bilder und Dokumente direkt in der App bereitstellen. Wie Sie die Mediathek über die API verwalten, erfahren Sie unter [Die Mediathek](#).

## News

Mit der **News**-Funktion können Sie manuell News erstellen und hochladen sowie neue Quellen anlegen und bearbeiten. Mehr Informationen finden Sie unter: [News](#).

## Nutzer synchronisieren

Ein gängiger Anwendungsfall ist die Synchronisierung von Benutzerdaten. Mit der API können Sie Nutzerprofile aus externen Systemen wie CRMs oder Vereinsverwaltungssoftware nahtlos integrieren. Weitere Informationen darüber, wie Sie Benutzerprofile erstellen, aktualisieren, suchen oder löschen können, finden Sie im Abschnitt zur **Nutzer-Synchronisation** unter [Nutzer synchronisieren](#).

## Push-Nachrichten Versenden

Eine effiziente Kommunikation mit den Nutzern Ihrer App ist entscheidend. Mit der API können Sie gezielt Push-Nachrichten an Benutzer oder Gruppen senden, beispielsweise um auf kurzfristige Änderungen hinzuweisen. Informationen dazu, wie Push-Nachrichten gezielt und effizient versendet

werden können, finden Sie unter [Push-Nachrichten Versenden](#).

## Termine verwalten

Die Terminverwaltung ist ein weiterer wesentlicher Anwendungsfall. Mithilfe der API können Sie Kalenderereignisse wie Trainingszeiten oder Veranstaltungen erstellen, aktualisieren oder anzeigen lassen. Dies erleichtert die Organisation von Terminen und Events direkt in der App. Wie Sie Kalender effektiv verwalten, erfahren Sie im Abschnitt zu **Terminen** unter [Termine bearbeiten](#).

## Workbook

Das **Workbook** ist eine leistungsstarke Funktion, die es ermöglicht, dynamische und interaktive Inhalte wie Umfragen oder Formulare innerhalb der App zu erstellen. Diese Inhalte können über die API verwaltet und an die Benutzer verteilt werden, um etwa Feedback einzuholen oder Daten zu sammeln. Weitere Details zur Nutzung des Workbooks finden Sie unter [Workbook](#).

# Gruppen

Gruppen bieten eine effektive Möglichkeit, Nutzer zu organisieren.

## Eine Gruppe erstellen

Mit der *Mutation* `createGroup` können Sie eine neue Gruppe anlegen. Im folgenden Beispiel wird eine Gruppe für die A-Jugend erstellt. Diese Gruppe ist privat, also nicht öffentlich sichtbar, und bestimmte Nutzer werden automatisch hinzugefügt. Alle Nutzer, die die Rolle „A-Jugend“ oder „App-Admin“ besitzen, sollen der Gruppe automatisch zugeordnet werden. Dafür tragen wir im `roles`-Attribut die Rollen-Keys ein. In diesem Fall sind das `"A-Jugend"` und `"AA"`.

```
mutation CreateGroup {
  createGroup(
    group: {
      name: "A-Jugend"
      publicGroup: false
      listedGroup: false
      description: "Gruppe der A-Jugend"
      roles: ["A-Jugend", "AA"]
    }
  ) {
    name
    memberIds
  }
}
```

Die Response zeigt die neu erstellte Gruppe mit den zugeordneten Mitgliedern anhand ihrer ID:

```
{
  "data": {
    "createGroup": {
      "name": "A-Jugend",
      "memberIds": [
        "66e41ca2eb07c6643266248c",
        "87g58sg8wv80s8684984654w"
      ]
    }
  }
}
```

## Alle Gruppen der App laden

Mit der *Query* `listGroups` können Sie eine Liste aller Gruppen der App abrufen. Dabei können Sie mit der Variable `ListedGroupsOnly` festlegen, ob nur gelistete Gruppen angezeigt werden sollen.

```
query ListGroups {
  listGroups(listedGroupsOnly: false) {
    id
    name
    description
  }
}
```

Die Response zeigt eine Liste der verfügbaren Gruppen:

```
{
  "data": {
    "listGroups": [
      {
        "id": "00_Appack_TEMPLATE_app_admin",
        "name": "App-Administratoren",
        "description": null
      },
      {
        "id": "00_Appack_TEMPLATE_push_admin",
        "name": "Push-Administratoren",
        "description": null
      },
      {
        "id": "8458s454fx4885e48s48982a",
        "name": "A-Jugend",
        "description": "Gruppe der A-Jugend"
      }
    ]
  }
}
```

## Daten einer Gruppe abrufen

Mit der *Query* `getGroup` können Sie die Informationen zu einer bestimmten Gruppe abrufen, einschließlich der Mitgliederliste. Im folgenden Beispiel werden die Mitglieder der Gruppe mit der ID „8458s454fx4885e48s48982a“ abgefragt.

```
query GetGroup {
  getGroup(id: "8458s454fx4885e48s48982a") {
    members {
      name
      firstname
    }
  }
}
```

Die Response gibt die Namen und Vornamen der Gruppenmitglieder zurück:

```
{
  "data": {
    "getGroup": {
      "members": [
        {
          "name": "Mustermann",
          "firstname": "Max"
        },
        {
          "name": "Musterfrau",
          "firstname": "Marta"
        }
      ]
    }
  }
}
```

### Weitere nützliche *Queries*:

- `listGroupRequests` – Listet offene Beitrittsanfragen für Gruppen auf.

### Weitere nützliche *Mutationen*:

- `updateGroup` – Aktualisiert die Informationen einer Gruppe.
- `deleteGroup` – Löscht eine Gruppe.
- `addGroupMember` – Fügt ein Mitglied zu einer Gruppe hinzu.
- `deleteGroupMember` – Entfernt ein Mitglied aus einer Gruppe.

- `promoteToGroupLeader` – Befördert ein Mitglied zum Gruppenleiter.
- `demoteToGroupMember` – Degradiert einen Gruppenleiter zu einem normalen Mitglied.
- `requestGroupAccess` – Fordert den Beitritt zu einer Gruppe an.
- `inviteToGroup` – Lädt einen Nutzer in eine Gruppe ein.
- `answerGroupAccessRequest` – Beantwortet eine Beitrittsanfrage.
- `cancelGroupAccessRequest` – Storniert eine ausstehende Beitrittsanfrage.

# Die Mediathek

In der Mediathek können Sie verschiedene Daten direkt laden oder hochladen.

## Dateien aus der Mediathek laden

Um alle Dateien aus der Mediathek zu laden, können Sie die *Query* `findResources` verwenden. Hierbei können Sie im `search`-Attribut mithilfe der `mimeType`s die Art der gewünschten Dateien angeben. Folgendes Beispiel lädt die ersten 5 Dateien aus der Mediathek, sortiert nach dem Dateinamen:

```
query FindResources {
  findResources(
    page: { page: 0, size: 5, sort: { direction: ASC, property: "name" } }
    search: { mimeType: ["video", "audio", "pdf", "image"] }
  ) {
    totalCount
    content {
      id
      url
      title
    }
  }
}
```

In der Response erhalten wir eine Liste von Dateien, darunter Bilder, Videos und PDFs:

```
{
  "data": {
    "findResources": {
      "totalcount": 50,
      "content": [
        {
          "id": "48561asdfg330c3e5a14561f",
          "url":
"https://cdn.appack.de/00_Appack_TEMPLATE/images/banner.png",
          "title": "banner.png"
        },
        {
          "id": "84a8dasd5255485e5s18gw8g",
          "url": "https://cdn.appack.de/00_Appack_TEMPLATE/media/04
Airwolf.mp3",
          "title": "Hymne.mp3"
        },
        {
          "id": "66aw4adsdw8a48de5a14ff8f",
          "url": "https://cdn.appack.de/00_Appack_TEMPLATE/images/logo.png",
          "title": "logo.png"
        },
        {
          "id": "66a8daf36b7865a58d2",
          "url": "https://cdn.appack.de/00_Appack_TEMPLATE/pdf/Satzung.pdf",
          "title": "Satzung.pdf"
        },
        {
          "id": "66a8daf36bwdfa484sd8w4ad",
          "url": "https://cdn.appack.de/00_Appack_TEMPLATE/media/video.mp3",
          "title": "video.mp4"
        }
      ]
    }
  }
}
```

## Spezifische Datei-Informationen laden

Um detaillierte Informationen zu einer spezifischen Datei zu erhalten, können Sie die *Query* `findResourcesByIds` verwenden. Dabei wird die ID der Datei übergeben, um Informationen wie Erstellungsdatum und MIME-Typ zu erhalten. Im folgenden Beispiel laden wir die Informationen zur Datei mit der ID `66aw4adsdw8a48de5a14ff8f`:

```
query FindResourcesByIds {
  findResourcesByIds(ids: ["66aw4adsdw8a48de5a14ff8f"]) {
    id
    url
    creationDate
    lastModificationDate
    mimeType
    title
    creator
  }
}
```

Die Response gibt folgende Informationen zur Datei zurück:

```
{
  "data": {
    "findResourcesByIds": [
      {
        "id": "66aw4adsdw8a48de5a14ff8f",
        "url": "https://cdn.appack.de/00_Appack_TEMPLATE/images/logo.png",
        "creationDate": "2024-07-30T12:22:11.632Z",
        "lastModificationDate": "2024-07-30T12:22:12Z",
        "mimeType": "image/png",
        "title": "logo.png",
        "creator": "Mustermann, Max"
      }
    ]
  }
}
```

### Weitere nützliche *Queries*

- `findResourceByFilename` - liefert Daten einer Ressource

### Weitere nützliche *Mutationen*

- `deleteResources` - löscht mehrere Ressourcen
- `updateStatistics` - Zeigt den verwendeten Speicherplatz

# News

Es ist möglich, neue News-Artikel zu erstellen und bestehende News aus dem Modul zu laden.

## Alle News-Quellen abrufen

Mit der *Query* `newsFeedDatasources` können alle verfügbaren News-Quellen geladen werden. Die `id` einer Quelle wird für weitere Operationen benötigt.

```
query NewsFeedDatasources {
  newsFeedDatasources {
    id
    type
    title
    color
    active
    url
  }
}
```

Die Antwort liefert eine Übersicht über alle News-Quellen:

```
{
  "data": {
    "newsFeedDatasources": [
      {
        "id": "6736f26b1955e13a0da1dfd7",
        "type": "SOCIAL_NETWORK",
        "title": "Instagram",
        "color": "#eb144c",
        "active": true,
        "url": null
      },
      {
        "id": "6720a794a9f8fe011ed9a83c",
        "type": "RSS",
        "title": "RSS Tagesschau",
        "color": "#045dd1",
        "active": true,
        "url": "https://www.tagesschau.de/index~rss2.xml"
      },
      {
        "id": "6720d5a30ae9790950c841ee",
        "type": "MANUAL",
        "title": "App-News",
        "color": "#9900ef",
        "active": true,
        "url": null
      }
    ]
  }
}
```

```
    }
  ]
}
}
```

## Neue News-Quellen erstellen

Mit der *Mutation* `link:https://developer.appack.de/api/#mutation-`createNewsFeedDatasource`]` können neue News-Quellen hinzugefügt werden. Die folgenden Typen sind verfügbar: **RSS**, **MANUAL** und **SOCIAL\_NETWORK**.

```
mutation CreateNewsFeedDatasource {
  createNewsFeedDatasource(
    input: {
      type: MANUAL
      active: true
      title: "App-News"
      color: "#333"
      autoSubscribe: false
      subscribable: false
      writerGroupIds: []
    }
  ) {
    id
    type
    title
    color
    active
    url
  }
}
```

Die Antwort zeigt die neu erstellte News-Quelle:

```
{
  "data": {
    "createNewsFeedDatasource": {
      "id": "677e6b1814dc2d0bbaa19d90",
      "type": "MANUAL",
      "title": "App-News",
      "color": "#333",
      "active": true,
      "url": null
    }
  }
}
```

## News aus dem Modul laden

Mit der *Query* `newsFeed` kann ein chronologisch sortierter Newsfeed basierend auf den angegebenen Datenquellen erstellt werden.

```
query NewsFeed {
  newsFeed(datasources: ["6720a794a9f8fe011ed9a83c", "6720d5a30ae9790950c841ee"],
  limit: 3) {
    id
    publishedDate
    title
    body
    link
    mediaRefs {
      url
    }
  }
}
```

Die Antwort liefert die neuesten drei News:

```
{
  "data": {
    "newsFeed": [
      {
        "id": "677e5d6372c7195f0f026da5",
        "publishedDate": "2025-01-08T11:05:00Z",
        "title": "Frohes Neues",
        "body": "<div>Wir w&uuml;nnschen allen Mitgliedern und ihren Familien ein frohes neues Jahr!</div>",
        "link": "",
        "mediaRefs": []
      },
      {
        "id": "677e5ccb07361654e6ccbf84",
        "publishedDate": "2025-01-08T11:00:04Z",
        "title": "Außenminister Schallenberg wird Übergangskanzler in Österreich",
        "body": "Nach dem Rücktritt von Kanzler Nehammer soll Österreichs Außenminister Schallenberg übergangsweise die Regierung leiten. Der Auftrag zur Regierungsbildung liegt inzwischen bei der rechtspopulistischen FPÖ.",
        "link": "https://www.tagesschau.de/ausland/europa/oesterreich-schallenberg-kanzler-100.html",
        "mediaRefs": [
          {
            "url": "https://images.tagesschau.de/image/aec56568-8c97-4498-b7db-d4be53b69907/AAAB1EU2bzE/AAABkZLhkrw/16x9-1280/alexander-schallenberg-104.jpg"
          }
        ]
      }
    ]
  }
}
```

```

    },
    {
      "id": "677e5ccb07361654e6ccbf81",
      "publishedDate": "2025-01-08T10:55:53Z",
      "title": "Iran lässt italienische Journalistin Cecilia Sala frei",
      "body": "Knapp drei Wochen war die italienische Journalistin Sala im Iran inhaftiert. Teheran warf ihr Verstöße gegen Mediengesetze des Landes vor. Nun wurde die 29-Jährige laut der italienischen Regierung freigelassen.",
      "link": "https://www.tagesschau.de/ausland/asien/iran-italienische-journalistin-100.html",
      "mediaRefs": [
        {
          "url": "https://images.tagesschau.de/image/3ac1031e-2097-43f4-9cce-ef1a1dda3a57/AAAB1EWLeRc/AAABkZLhkrw/16x9-1280/cecilia-sala-108.jpg"
        }
      ]
    }
  ]
}
}

```

## News-Eintrag erstellen

Mit der *Mutation* `createManualNewsFeedEntry` kann ein eigener News-Eintrag erstellt werden. Die *Mutation* `updateManualNewsFeedEntry` ermöglicht die Bearbeitung bestehender Einträge, während mit der *Mutation* `deleteManualNewsFeedEntry` Einträge gelöscht werden können. Der `body`-Inhalt wird in HTML bereitgestellt.

```

mutation CreateManualNewsFeedEntry {
  createManualNewsFeedEntry(
    input: {
      datasourceId: "6720d5a30ae9790950c841ee"
      title: "Erfolgreicher Helfereinsatz"
      mediaRefs: [],
      body: "<div>Am Wochenende fand ein erfolgreicher Helfereinsatz statt.</div>"
      link: ""
      author: "Vorstand"
      publishedDate: "2025-01-08T12:20:00.000Z"
      expirationDate: null
      active: true
    }
  ) {
    id
    datasourceId
    publishedDate
    expirationDate
    title
    body
  }
}

```

```
    link
    author
    active
  }
}
```

Die Antwort zeigt den erstellten News-Eintrag:

```
{
  "data": {
    "createManualNewsFeedEntry": {
      "id": "677f98a07acfb7c9708b5f7",
      "datasourceId": "6720d5a30ae9790950c841ee",
      "publishedDate": "2025-01-08T12:20:00Z",
      "expirationDate": null,
      "title": "Erfolgreicher Helfereinsatz",
      "body": "<div>Am Wochenende fand ein erfolgreicher Helfereinsatz
statt.</div>",
      "link": "",
      "author": "Vorstand",
      "active": true
    }
  }
}
```

# Nutzer synchronisieren

Ein zentraler Anwendungsfall für die Appack-API, der von vielen Kunden häufig angefragt wurde, ist die Anbindung einer eigenen Nutzerverwaltung. Diese kann z.B. eine selbst entwickelte Lösung, eine cloudbasierte Vereinsverwaltungssoftware oder ein bestehendes CRM-System sein. Mit den bereitgestellten *Mutationen* und *Queries* können Sie Benutzerprofile anlegen, suchen, aktualisieren und löschen.

## Benutzerprofil anlegen

Um ein Benutzerprofil anzulegen, nutzen wir die *Mutation* `createUserProfile`. In diesem Beispiel wird ein neues Profil für "Marta Musterfrau" angelegt, die der Abteilung "Billard" zugeordnet wird:

```
mutation CreateUserProfile {
  createUserProfile(
    userProfile: {
      departments: [
        {
          enumId: "00_Appack_TEMPLATE-department"
          value: "Billard"
          enumKey: "dezernat"
        }
      ]
      name: "Musterfrau"
      firstname: "Marta"
      email: "marta@musterfrau.de"
    }
  ) {
    id
    name
    firstname
    email
    departments {
      enumId
      enumKey
      value
    }
  }
}
```

Die folgende Response zeigt das erstellte Benutzerprofil:

```
{
  "data": {
    "createUserProfile": {
      "id": "66e41ca2eb07c6643266248c",
      "name": "Musterfrau",
      "firstname": "Marta",
      "email": "marta@musterfrau.de",
      "departments": [
        {
          "enumId": "00_Appack_TEMPLATE-department",
          "enumKey": "dezernat",
          "value": "billard"
        }
      ]
    }
  }
}
```

## Benutzerprofil suchen

Um nach bestehenden Benutzern zu suchen, verwenden wir die *Query* `findUserProfile`. Hier suchen wir nach dem Namen "Marta" und erhalten eine paginierte Liste von Benutzerprofilen:

```
query FindUserProfile {
  findUserProfile(
    search: "Marta"
    page: {size: 25, page: 0, sort: { direction: ASC, property: "name" } }
  ) {
    totalcount
    content {
      id
      name
      firstname
    }
  }
}
```

Die folgende Response zeigt das gefundene Benutzerprofil:

```
{
  "data": {
    "findUserProfile": {
      "totalcount": 1,
      "content": [
        {
          "id": "66e41ca2eb07c6643266248c",
          "name": "Musterfrau",
          "firstname": "Marta"
        }
      ]
    }
  }
}
```

Als Suchwert (`search`) kann jedes Attribut aus dem `UserProfile` verwendet werden, wie z.B. der Name, die E-Mail-Adresse oder andere Identifikationsinformationen. So lassen sich gezielt Benutzerprofile anhand bestimmter Kriterien finden.

## Benutzerprofil aktualisieren

Um die Informationen eines Benutzerprofils zu aktualisieren, z.B. die Rolle, verwenden wir die *Mutation* `updateUserProfile`. Hier wird die Rolle von Marta auf "Mitglied" gesetzt:

```
mutation UpdateUserProfile {
  updateUserProfile(id: "66e41ca2eb07c6643266248c",
    userProfile: { roles: [
      {enumId:"00_Appack_TEMPLATE-roles", enumKey:"M", value:"Mitglied"}
    ] }
  ) {
    id
    email
    name
    firstname
    roles {
      enumId
      enumKey
      value
    }
  }
}
```

Die Response zeigt das aktualisierte Profil:

```
{
  "data": {
    "updateUserProfile": {
      "id": "66e41ca2eb07c6643266248c",
      "email": "marta@musterfrau.de",
      "name": "Musterfrau",
      "firstname": "Marta",
      "roles": [
        {
          "enumId": "00_Appack_TEMPLATE-roles",
          "enumKey": "M",
          "value": "Mitglied"
        }
      ]
    }
  }
}
```

**Weitere nützliche *Queries*:**

- `findUserProfilesModifiedSince` - Listet alle Benutzerprofile, die seit dem genannten Datum geändert wurden
- `findOneUserProfile` - Lädt einen App-Nutzer

**Weitere nützliche *Mutationen*:**

- `deleteUserProfile` - Löscht ein Benutzerprofil

# Push-Nachrichten Versenden

Mit Push-Nachrichten können gezielt Informationen an Benutzer der App gesendet werden. Dies kann alle registrierten Benutzer betreffen oder spezifische Zielgruppen wie Abteilungen, Gruppen oder Rollen. Typische Anwendungsfälle umfassen Ankündigungen, Veranstaltungsänderungen oder wichtige Benachrichtigungen.

## Verfügbare Push-Kanäle abrufen

Zuerst müssen wir die verfügbaren Push-Kanäle ermitteln, um eine Nachricht gezielt an einen Kanal zu senden. Dazu nutzen wir die *Query* `listPushChannels`. Die Antwort liefert uns eine Liste der Kanäle, inklusive der ID, welche für den gezielten Versand benötigt wird:

```
query ListPushChannels {
  listPushChannels {
    id
    number
    name
    description
    imageResourceId
    soundId
    autoSubscribe
    initialSubscribe
    active
    subscriptionCount
    allowedGroupIds
  }
}
```

Die Response zeigt uns eine Übersicht aller Push-Kanäle:

```
{
  "data": {
    "listPushChannels": [
      {
        "id": "5e8dcabb34fc37d152ae23cf",
        "number": 1,
        "name": "Allgemeine News",
        "description": "Hier werden allgemeine News veröffentlicht",
        "imageResourceId": null,
        "soundId": null,
        "autoSubscribe": true,
        "initialSubscribe": true,
        "active": true,
        "subscriptionCount": 400,
        "allowedGroupIds": null
      },
    ],
  },
}
```

```

    {
      "id": "5ec3de8434fc8a61a1c3f2ea",
      "number": 2,
      "name": "Beispielkanal 1",
      "description": "",
      "imageResourceId": null,
      "soundId": null,
      "autoSubscribe": false,
      "initialSubscribe": false,
      "active": false,
      "subscriptionCount": 0,
      "allowedGroupIds": null
    }
  ]
}

```

## Push-Nachricht versenden

Mit der *Mutation* `sendPushNotificationToDevices` können wir nun eine Push-Nachricht versenden. Im folgenden Beispiel senden wir die Nachricht nur an den Kanal "Allgemeine News". Es besteht jedoch auch die Möglichkeit, gezielt an bestimmte Abteilungen, Gruppen oder Rollen zu senden, so wie Sie es aus dem Modul der App bzw. dem CMS kennen.

Um dies zu erreichen, müssen Sie die entsprechenden IDs in die Felder `departmentIds` (Schlüssel der Abteilung, abrufbar mit `listEnumValues` oder durch Nachschlagen im CMS), `groupIds` (abrufbar mit `listGroups`) und/oder `roleIds` (Schlüssel der Rolle, ebenfalls über `listEnumValues` oder im CMS auffindbar) eintragen. Bei der Auswahl mehrerer Kriterien (z. B. Abteilungen, Gruppen und Rollen) wird eine logische UND-Verknüpfung angewendet. Das bedeutet, dass ein Benutzer alle ausgewählten Kriterien erfüllen muss, um die Nachricht zu erhalten.

```

mutation SendPushNotificationToDevices {
  sendPushNotificationToDevices(
    notification: {
      mode: NORMAL
      message: "Unsere Mannschaft hat gewonnen!!"
      channelId: ["5e8dcabb34fc37d152ae23cf"]
    }
  ) {
    id
    message
  }
}

```

Die folgende Response zeigt die versendete Nachricht:

```
{
  "data": {
    "sendPushNotificationToDevices": {
      "id": "5e8dcabb34fc37d152ae23cf",
      "message": "Unsere Mannschaft hat gewonnen!!"
    }
  }
}
```

#### Weitere nützliche *Queries*:

- `calculatePushNotificationMaxReceivers` – Berechnet die maximale Anzahl der Empfänger für eine Push-Nachricht.
- `findPushNotifications` – Findet alle gesendeten Push-Nachrichten.

#### Weitere nützliche *Mutationen*:

- `updatePushNotification` – Aktualisiert eine bereits gesendete Push-Nachricht.
- `deletePushNotification` – Löscht eine Push-Nachricht.
- `sendPushNotificationToProfile` – Sendet eine Push-Nachricht an ein bestimmtes Benutzerprofil.
- `sendPushNotificationToDevices` – Versendet Push-Nachrichten im Modus "NORMAL" oder "TEST".
- `sendPushNotificationToEventAttendees` – Sendet Nachrichten an Teilnehmer eines Events.

# Termine bearbeiten

Die Terminverwaltung ist eine zentrale Funktion der Appack-Plattform, die es ermöglicht, Kalender zu erstellen und Ereignisse zu organisieren. Sie erlaubt es, Kalender für bestimmte Zielgruppen (wie z. B. Trainingszeiten einer Jugendmannschaft) anzulegen und Termine in der App anzuzeigen, bei denen sich Nutzer anmelden können. Es können auch externe Kalender per iCal-Feed integriert werden.

## Kalender auflisten

Mit der *Query* `listCalendars` können alle Kalender im System aufgelistet werden. Diese Abfrage liefert einen Überblick über alle verfügbaren Kalender. Die angezeigte Kalender-ID wird benötigt, um Termine abzurufen oder neue Termine hinzuzufügen.

```
query ListCalendars {
  listCalendars {
    id
    title
  }
}
```

Die Response zeigt die vorhandenen Kalender, z. B.:

```
{
  "data": {
    "listCalendars": [
      {
        "id": "6357900543d4db038c24ed28",
        "title": "Veranstaltungen"
      },
      {
        "id": "66df0b1c7940f377a85773c4",
        "title": "Trainings"
      }
    ]
  }
}
```

## Termine des Kalenders Anzeigen

Um die Termine eines Kalenders zu verwalten, können Sie entweder alle Ereignisse oder nur bevorstehende Ereignisse anzeigen lassen. Die Abfrage `findCalendarEvents` zeigt alle Ereignisse eines bestimmten Kalenders, während `listUpcomingCalendarEvents` nur zukünftige Termine zurückgibt.

Die folgende Abfrage zeigt alle Termine eines Kalenders:

```
query FindCalendarEvents {
  findCalendarEvents(calendarIds: "6357900543d4db038c24ed28") {
    id
    title
    description
    dateStart
    dateEnd
  }
}
```

Die Response zeigt die Termine des Kalenders:

```
{
  "data": {
    "findCalendarEvents": [
      {
        "id": "66f245asdfw32b1801984asd",
        "title": "Weihnachtsfeier",
        "description": "Es ist bald Weihnachten und das Fest wird stattfinden",
        "dateStart": "2024-12-15T16:00:00Z",
        "dateEnd": "2024-12-15T18:00:00Z"
      },
      {
        "id": "66s2aa44545asd831185c0de",
        "title": "Neujahrsfeier",
        "description": "Das neue Jahr hat begonnen",
        "dateStart": "2025-01-05T12:30:00Z",
        "dateEnd": "2025-01-05T14:30:00Z"
      }
    ]
  }
}
```

## Termin erstellen

Um einen Termin in einem bestimmten Kalender hinzuzufügen, nehmen wir die ID des gewünschten Kalenders und verwenden die *Mutation* `createCalendarEvents`. Diese Mutation ermöglicht es, den Titel, die Start- und Endzeiten, sowie weitere Informationen festzulegen.

```
mutation CreateCalendarEvents {
  createCalendarEvents(calendarId: "6616459142fcd13fa516e44d", data: {
    title: "Kinderfest"
    dateEnd: "2024-09-22T08:30:00.000Z"
    dateStart: "2024-09-22T07:30:00.000Z"
    description: "Am kommenden Samstag ist wieder Kinderfest"
  }) {
    id
    title
    description
    dateStart
    dateEnd
  }
}
```

Die folgende Response bestätigt das erfolgreiche Erstellen des Termins:

```
{
  "data": {
    "createCalendarEvents": [
      {
        "id": "66ed255c2e11a56e400ece0f",
        "title": "Kinderfest",
        "description": "Am kommenden Samstag ist wieder Kinderfest",
        "dateStart": "2024-09-22T07:30:00Z",
        "dateEnd": "2024-09-22T08:30:00Z"
      }
    ]
  }
}
```

**Weitere nützliche *Queries*:**

- `listCalendarByComponentId` – Zeigt Kalender nach Technischen Id an.
- `findCalendarEvents` – Sucht nach spezifischen Terminen in Kalendern.
- `listUpcomingCalendarEvents` – Listet anstehende Termine auf.

**Weitere nützliche *Mutationen*:**

- `updateCalendar` – Aktualisiert die Details eines Kalenders.
- `updateCalendarEvents` – Aktualisiert die Informationen eines bestimmten Termins.
- `createCalendar` – Erstellt einen neuen Kalender.

# Workbook

Workbooks bieten eine vielseitige Möglichkeit, Daten zu lesen oder zu schreiben. Dies kann beispielsweise genutzt werden, um aktuelle News zu erstellen oder neue Einträge im Schwarzen Brett zu veröffentlichen. Zusätzlich können Einträge aus anderen Datenbanken, wie der Mitglied-werden-DB, ausgelesen und verarbeitet werden.

## Arbeitsblätter aus einem Workbook laden

Mit der *Query* `LoadWorkbookComponent` können Sie alle Arbeitsblätter eines Workbooks auslesen. Dies ist hilfreich, wenn Sie Informationen über die verfügbaren Arbeitsblätter in einem bestimmten Workbook benötigen. Das folgende Beispiel zeigt, wie die Abfrage strukturiert ist:

```
query LoadWorkbookComponent {
  loadWorkbookComponent(workbookId: "00_Appack_TEMPLATE_Workbook_1718272727939") {
    id
    adminMail
    worksheetIds
    worksheets {
      id
      name
    }
  }
}
```

Diese Abfrage gibt Informationen über das Workbook zurück, wie die ID, die E-Mail-Adresse des Administrators und eine Liste der enthaltenen Arbeitsblätter. Die folgende JSON-Response zeigt die Rückgabe der Query:

```
{
  "data": {
    "loadWorkbookComponent": {
      "id": "00_Appack_TEMPLATE_Workbook_1718272727939",
      "adminMail": "mail@verein.de",
      "worksheetIds": [
        "6672746e0c08835b02533cb4"
      ],
      "worksheets": [
        {
          "id": "6672746e0c08835b02533c71",
          "name": "Anmeldungen"
        }
      ]
    }
  }
}
```

In diesem Fall gibt es ein Arbeitsblatt namens "Anmeldungen", dessen ID `6672746e0c08835b02533c71` ist.

## Arbeitsblattdaten laden

Nachdem Sie die IDs der Arbeitsblätter haben, können Sie mit der *Query* `loadWorksheetData` die Daten eines spezifischen Arbeitsblatts laden. In der folgenden Abfrage laden wir die Daten aus dem Arbeitsblatt mit der ID `6672746e0c08835b02533c71`:

```
query LoadWorksheetData {
  loadWorksheetData(filter: {
    worksheetId: "6672746e0c08835b02533c71",
    page: 0,
    size: 50,
    sort: {
      property: "_autoId",
      direction: DESC
    }
  }) {
    totalcount
    content {
      id
      autoId
      created
      lastModified
      user
      deviceId
      userProfileId
      data
      userProfile {
        email
      }
    }
  }
}
```

Diese Abfrage lädt die ersten 50 Einträge des Arbeitsblattes, sortiert nach dem Attribut *autoId* in absteigender Reihenfolge. Die folgende JSON-Response zeigt einen Beispiel-Datensatz:

```
{
  "data": {
    "loadWorksheetData": {
      "totalcount": 1,
      "content": [
        {
          "id": "66f67d82f282e42c565de558",
          "autoId": 10,

```

```

"created": "2024-09-04T06:27:40.356Z",
"lastModified": "2024-09-04T06:27:40.402Z",
"user": null,
"deviceId": "esdvHCPLS-2tgh1FgTaR1",
"userProfileId": "h5opjj3u92masd28u3hj398h",
"data": {
  "firstName": "Maxi",
  "lastName": "Musermann",
  "birthDate": "20.05.1995",
  "address": "Straße 1",
  "postalCode": "12345",
  "livingPlace": "Stadt",
  "phoneNumber": "0123456789",
  "eMail": "maxi@musermann.de",
  "familyMember_01": null,
  "familyMember_02": null,
  "familyMember_03": null,
  "bankAccountHolder": null,
  "bankIBAN": null,
  "bankName": null,
  "userPic": null
},
"userProfile": {
  "email": "maxi@musermann.de"
}
}
]
}
}
}
}

```

## Neue Einträge erstellen

Mit der *Mutation* `createWorksheetRow` können Sie neue Einträge in einem Arbeitsblatt erstellen. Die Daten werden im `data`-Feld als JSON-Objekt übergeben, wobei die Keys den keys der Spalten des Arbeitsblattes entsprechen. **Wichtig:** Der Name der Spalte im CMS ist nicht der Key der Spalte, den Key bekommen sie mit der *Query* `loadWorksheetData` aus dem `data` Attribut.

Hier ist ein Beispiel für das Erstellen eines neuen Eintrags:

```
mutation CreateWorksheetRow {
  createWorksheetRow(worksheetId: "6672746e0c08835b02533cb4", data: {
    Titel: "Veranstaltung",
    Beschreibung: "Kommenden Freitag ist eine Veranstaltung"
  }) {
    id
    autoId
    created
    lastModified
    user
    deviceId
    userProfileId
    data
  }
}
```

Die Response zeigt den neu erstellten Eintrag:

```
{
  "data": {
    "createWorksheetRow": {
      "id": "84a927d82f836e42c565de345",
      "autoId": 4,
      "created": "2024-09-27T09:40:18.636Z",
      "lastModified": "2024-09-27T09:40:18.636Z",
      "user": "berta",
      "deviceId": "esdvHCPLS-2tgh1FgTaR1",
      "userProfileId": "h5opjj3u92masd28u3hj398h",
      "data": {
        "Titel": "Veranstaltung",
        "Beschreibung": "Kommenden Freitag ist eine Veranstaltung"
      }
    }
  }
}
```

**Weitere nützliche *Mutationen*:**

- `updateWorksheetRow` - Ändert eine bestehende Row
- `duplicateWorksheetRow` - Dupliziert eine bestehende Row
- `deleteWorksheetRows` - Löscht mehrere Einträge

# Programmierschnittstelle