API Nutzerdokumentation

0 Inhalt

1 Antwortformat

Jede Antwort der API besteht aus einem JSON-Objekt wie folgt aufgebaut:

```
{
   resource : [],  // die angefragte Daten
   code : {
      id : int,   // Erfolgs- oder Fehlercode
      msg: string // Erfolgs- oder Fehlernachricht
   },
   meta : {
      // diverse Zusatzinformationen, z.B.:
      id   : int, // id des neu eingefügten Datensatzes ( POST )
      page : int, // Seitennummer (im Fall von code.id = 16 )
      pages: int  // Gesamtzahl der Seiten für diese Anfrage
      data : {},  // Die übergebenen Daten (falls code.id > 63 )
   },
   log : [] // u.a. Fehlermeldungen, leer im Erfolgsfall
}
```

Die möglichen Werte für code sind wie folgt festgelegt:

Wert	Bedeutung
< 0	Unbekannter Fehler.
0-15	**'SUCCESS'-Rückmeldungen**
0	Anfrage erfolgreich
1	Alle per GET angefragten Daten wurden gesendet
2	Aktualisierung (PUT) erfolgreich abgeschlossen
3	Einfügen des Datensatzes (POST) erfolgreich abgeschlossen
16-31	**'NOTICE'-Rückmeldungen**
16	Der Rückgabewert war insgesamt zu groß. Deshalb wurde nur eine „Seite“ an Daten zurückgemeldet. Das meta-Attribut enthält weitere Informationen (Seite, Anzahl Seiten)
32-63	**'WARNING'-Rückmeldungen**
32	Deprecated: Dieser Pfad funktioniert zwar noch, wird aber in Zukunft durch einen anderen ersetzt. Siehe meta-Attribut für weitere Informationen
64+	**'ERROR'-Rückmeldungen**
64	Es ist ein Fehler aufgetreten, der nicht genauer beschrieben werden konnte
65	Die POST- oder PUT-Anfrage enthielt keine Daten zur Verarbeitung
66	Die mit PUT zu aktualisierenden Daten wurden auf dem Server nicht gefunden
67	Die Aktualisierung ist fehlgeschlagen
68	Zugriff verweigert
69	Die mit GET angefragten Daten wurden nicht gefunden
70	Fehlerhafte Anfrage

2 /address

2.1 Pfadübersicht

#	Methode	Pfad	Aktion
1	GET	/address	Gibt alle „Adressen“ des Benutzers zurück
2	GET	/address/id/**:id**	Gibt die „Adresse“ mit der id :id zurück
3	GET	/address/zip/**:zip**	Gibt alle „Adressen“ zurück, deren PLZ mit :zip beginnen.
4	GET	/adress/name/**:search**	Gibt alle „Adressen“ zurück, bei denen :search im Namen vorkommt.
5	GET	/calendar/series/id/**:id**	Gibt die Terminserie mit der id :id zurück
6	POST	/address	Schreibt eine „Adresse“ in die Datenbank
7	PUT	/address/id/**:id**	Bearbeitet die „Adresse“ mit der id :id

3 /articles

3.1 Pfadübersicht

#	Methode	Pfad	Aktion
1	GET	/articles	Gibt alle Artikel des Benutzers zurück
2	GET	/articles/id/**:id**	Gibt den Artikel mit der id :id zurück
3	GET	/articles/artno/**:artno**	Gibt den Artikel mit der Artikelnummer :artno zurück
4	POST	/articles	Schreibt einen Artikel in die Datenbank
5	PUT	/articles/id/**:id**	Bearbeitet den Artikel mit id :id
6	POST	/articles/external	Fügt eine externe Eigenschaft hinzu

3.2 Details

3.2.1 POST /articles/external

Über diesen Pfad können Sie Ihren Artikeln zusätzliche Eigenschaften geben, die Sie für Ihr System benötigen, aber ebenfalls über fotano Weboffice administrieren möchten. Beim Einfügen von zusätzlichen Eigenschaften *müssen* Sie folgende Werte angeben:

Schlüssel	Inhalt
name	Der technische Name der neuen Eigenschaft. (z.B.: 'color')
default	Wird bei der Ausgabe eines Artikels kein Wert für diese Eigenschaft gefunden, wird stattdessen dieser Wert zurückgegeben. (z.B.: '#ff0000')
label	Sie können sich die Eigenschaft im fotano Weboffice unter einem anderen Namen anzeigen lassen (z.B.: 'Farbe')

Nachdem Sie diese zusätzliche Eigenschaft einmalig erstellt haben, können Sie sie verwenden, als wäre sie eine fotano Weboffice Artikeleigenschaft.

4 /calendar

Methode	Pfad	Aktion
GET	/calendar	Gibt alle Termine des Benutzers zurück
GET	/calendar/id/**:id**	Gibt den Termin mit der id :id zurück
GET	/calendar/series/**:id**	Gibt alle Termine der Terminserie mit der id :id zurück.
GET	/calendar/series	Gibt alle Terminserien zurück
GET	/calendar/series/id/**:id**	Gibt die Terminserie mit der id :id zurück
POST	/calendar	Schreibt einen Termin in die Datenbank
PUT	/calendar/id/**:id**	Bearbeitet den Termin mit der id :id

5 /customer

Methode	Pfad	Aktion
GET	/customer	Gibt alle Kunden zurück.
GET	/customer/id/**:id**	Gibt den Kunden mit der id :id zurück
POST	/customer	Schreibt einen Kunden in die Datenbank
PUT	/customer/id/**:id**	Bearbeitet den Kunden mit der id :id

6 /personal

Methode	Pfad	Aktion
GET	/personal	Gibt alle Mitarbeiter zurück.
GET	/personal/id/**:id**	Gibt den Mitarbeiter mit der id :id zurück
GET	/personal/position/**:pos**	Gibt alle Mitarbeiter mit der Position :pos zurück
POST	/personal	Fügt einen Mitarbeiter hinzu
PUT	/personal/id/**:id**	Bearbeitet den Mitarbeiter mit der id :id

7 /pos

Methode	Pfad	Aktion
GET	/pos	Gibt alle Verkäufe zurück.
GET	/pos/id/**:id**	Gibt den Verkauf mit der id :id zurück
POST	/pos	Fügt einen Verkauf hinzu
PUT	/pos/id/**:id**	Bearbeitet den Verkauf mit der id :id
GET	/pos/id/**:id**/bon	Gibt alle Bons des Verkauf mit der id :id zurück
GET	/pos/bon/id/**:id**	Gibt den Bon mit der id :id zurück
POST	/pos/bon	Fügt einen Bon hinzu
GET	/pos/bon/id/**:id**/article	Gibt alle Artikel des Bons mit der id :id zurück
GET	/pos/bon/article/id/**:id**	Gibt den Bon-Artikel mit der id :id zurück
POST	/post/bon/article	Fügt einen Bon-Artikel hinzu
GET	/pos/id/**:id**/bonprint	Gibt alle Bonprints des Verkaufs mit der id :id zurück
GET	/pos/bonprint/id/**:id**	Gibt den Bonprint mit der id :id zurück
POST	/pos/bonprint	Fügt einen Bonprint hinzu
GET	/pos/help	Gibt die Datenstruktur für einen Verkauf zurück
GET	/pos/bon/help	Gibt die Datenstruktur für einen Bon zurück
GET	/pos/bon/article/help	Gibt die Datenstruktur für einen Bon-Artikel zurück
GET	/pos/bonprint/help	Gibt die Datenstruktur für einen Bonprint zurück

Ein „Verkauf“ im Kontext unserer API ist ein vollständiger Verkaufsvorgang von der ersten Bestellung bis zur Bezahlung. Im Restaurant kann damit z.B. ein Tisch gemeint sein.