# Bussio

Inhalt

[Bussio 1](#_Toc95384114)

[Einleitung: 1](#_Toc95384115)

[Webseite: 3](#_Toc95384116)



Bussio Handbuch

## Einleitung:

Bussio ist ein Webservice welcher einem alle Bus Abfahrten für eine Haltestelle anzeigt, sowie das Wetter. Man kann diese Anfragen entweder über unsere User Seite machen oder direkt als GET Anfrage an den Server. Unser Webservice hängt von unserer Community ab, um alle Haltestellen zu lernen. Jede nicht bekannte Haltestelle, welche eine erfolgreiche Abfrage zurückgibt wird in unseren Datenspeicher aufgenommen und ist für alle anderen dann verfügbar.

Dokumentation:

/GET\_BUS\_INFO/:name/:dd?/:mm?/:hh?/:min?/

Parameter:

name: Name der Haltestelle, nicht optional

dd: Tag des Monats, optional

mm: Monat, optional

hh: Stunde, optional

min: Minute, optional

Eine Get Request welche einem alle Busse zurückgibt die von der Haltestelle name abfahren, sind die Parameter: dd, mm, hh, mm nicht angegeben nimmt es einfach die Momentane Zeit für diesen Parameter.

Rückgabe:

Bei erfolgreicher Abfrage gibt es ein JSON(siehe [Seite](https://daten.buergernetz.bz.it/)) zurück mit allen Informationen über die Abfahrten.

Bei fehlerhafter Abfrage oder falls der Datenbank Server des Busunternehmens nicht verfügbar ist gibt es einen Status von 400 zurück und die Nachricht „unable to get info“

Bei einer nicht existierenden Bushaltestelle wird der Status 401 zurückgegeben.

/GET\_WEATHER\_INFO/:name?/

Parameter:

name: Name der Region, optional

Eine Get Request welche einem Wetter Informationen für Heute und Morgen zurückgibt. Sollte der Parameter name angegeben sein gibt er nur die Wetterinformationen für die angegeben Region zurück, wenn er leer ist werden die Wetterinformationen für ganz Südtirol zurückgegeben

Rückgabe:

Bei erfolgreicher Abfrage gibt es ein JSON zurück mit den Wetterinformationen für Heute und Morgen. Sollte die Region falsch sein bekommt man einen Status 400 und die Nachricht: „Unkown region: NAME  
Available regions: REGIONEN“, Regionen sind dabei alle Regionen welche der Server kennt. Sollte ein Fehler bei der Abfrage zum Datenbank Server auftauchen, bekommt man einen Status 400 und die Nachricht „unable to get info“. Sollten keinen Wetterinfos für Heute oder Morgen existieren ist dieses Element in der JSON null.

/GET\_STOPS/

Eine Get Request welche einem alle dem Server bekannten Haltestellen zurückgibt

Rückgabe: JSON mit allen Namen, Beispiel:

„stops“:[

{„name“: „Brixen“},

{„name“: „Brixen2“}

]

/ADD\_STOP/:stop

Parameter:

stop: Name der Haltestelle

Eine Post Request mit welcher man eine Anfrage an den Server schicken kann um eine Haltestelle hinzuzufügen.

Rückgabe:

Bei erfolgreicher Abfrage einen Status von 200 und die Nachricht: „Added stop, thank you :)“

Bei fehlerhafte Abfrage:

Status 400, „cannot add, stop exists“, falls der Server die Haltestelle schon kennt

Status 400, „cannot add servingLines null“, falls die Abfrage keine sinnvolle Rückgabe hat

Status 400, „failed“, falls es einen anderen Fehler gab

## Webseite:

Aufbau:

**index.html**

Das ist als start Seite gedacht. Also auf die Seite wo der Benutzer als erster draufkommt. Bei der Seite hat er einen Table mit allen Bushaltestellen die auf dem Server gespeichert sind. Die Seite bietet auch die Möglichkeit einer Suchfunktion von der Haltestelle. So muss man nicht die ganze Tabelle nach der Haltestelle durchsuchen. Gibt es die gewünschte Haltestelle nicht, dann kann man den Namen der Haltestelle eingeben und trotzdem auf die **info.html** weitergehen. Sollte die eingegebene Haltestelle dann wirklich exestieren, dann merkt sich der Server die Haltestelle und gibt es beim nächsten Aufruf der **index.html** Seite auch aus.

