

# Dokumentation zum Routen- und Abfahrtsfinder - 2022

## Allgemeine Informationen:

**Autoren und Entwickler:** Philipp Olivotto - [stoliphi@bx.fallmerayer.it](mailto:stoliphi@bx.fallmerayer.it)  
Noel Rovara – [rovnoe@bx.fallmerayer.it](mailto:rovnoe@bx.fallmerayer.it)

**Aktuelle Version vom 11.02.2022:** 1.0

**Status:** Abgeschlossen – Fehler könnten auftreten

**GitHub Repository:** <https://github.com/zbaakez/zbaakez.github.io>

**Adresse der Website:** <https://zbaakez.github.io/>

**Auftraggeber:** Alexander Larcher

## Einleitung

Wir sollten ein Projekt erstellen, welches die APIs des Südtiroler Verkehrsnetzes verwendet. Wir haben uns an die Arbeit gemacht und haben so eine nützliche Applikation geschaffen, welche Verbindungen von Haltestellen mit öffentlichen Verkehrsmitteln ausgibt und alternativ Abfahrten von den Öffentlichen an bestimmten Haltestellen ausgibt. Uns war es wichtig, dass unsere Web-Applikation auf allen Geräten einwandfrei funktioniert und sie sollte benutzerfreundlich sein. Das Design soll minimalistisch und modern sein. Zusätzlich haben wir uns entschieden, eine kleine Wetteranzeige einzurichten, um dem Benutzer das aktuelle Wetter anzuzeigen.

## Beschreibung und Erklärung zur Applikation

### 1. Die Routensuche

#### Overview:

The screenshot shows the routing application interface. On the right side, there are eight arrows pointing to specific elements of the interface, each with a descriptive text label:

- Header mit Button "Zu Abfahrten" - wechselt das Fenster Rechts - aktuelle Zeit
- Eingabe des Startpunkts (Haltestelle) - Autocompletehilfe
- Eingabe des Endpunkts (Haltestelle) - Autocomplete
- Weitere Optionen - ausgeklappt
- Wahl zwischen Abfahrtszeit und Ankunftszeit
- Wahl der Zeit [Standardmäßig jetzt]
- Abfahrtsdatum [Standardmäßig heute]
- Art der Verbindung - wirkt sich auf die Berechnung der Route aus
- Klick auf den Button -> Routensuche startet

Die Routensuche ist das Hauptaugenmerk der Applikation. Sie wird dazu verwendet, eine passende Route mit öffentlichen Verkehrsmitteln im Landesbereich Südtirol zu finden.

**Verpflichtende Eingaben** dafür sind ein Startpunkt und ein Endpunkt. In diesen zwei Eingabefeldern gibt man die Starthaltestelle und die Endhaltestelle ein. Um eine korrekte Eingabe zu erreichen, klickt man auf die Vorschläge, welches einem ein Vorschlagfeld liefert, nachdem man eine Teileingabe getätigt hat. Die Eingabe muss korrekt sein. Diese Vorschläge können unter Umständen eine kleine Weile dauern [wenige Sekunden].

Hat man diese Verpflichtenden Eingaben getätigt klickt man auf den Button mit dem Text „Verbindung suchen!“. Dieser schickt die Eingaben an den Server und man erhält passende Verbindungen angezeigt. Gelegentlich kann es dabei zu Problemen kommen und es wird ein Fehler ausgegeben. Fehlerquellen sind eine fehlende Internetverbindung, falsche Eingabe, Serverprobleme oder es gibt einfach keine Route bei den gewählten Eingaben. Bei korrekter Eingabe sollten Fehler der Ausnahmefall sein.

**Optionale Eingaben** sind unter dem Dropdown-Menü „Weitere Optionen“ zu finden. Dabei handelt es sich um eine Abfahrts- bzw. Ankunftszeit, ein Abfahrtsdatum und um die Art der Verbindung.

Standardmäßig sind diese Eingaben wie folgt eingestellt:

- Abfahrtszeit: *jetzt*
- Abfahrtsdatum: *heute*
- Art der Verbindung: *schnellste*

Unterschied Abfahrtszeit und Ankunftszeit:

Ist der Text Abfahrtszeit grün, so ist diese Option eingestellt, ist der Text „*Ankunftszeit*“ grün, so ist diese Option ausgewählt.

Mit der Option „*Abfahrtszeit*“ sucht man Routen, welche im Bereich der eingestellten Uhrzeit starten. Mit der Option „*Ankunftszeit*“ sucht man Routen, welche zur eingestellten Uhrzeit ankommen.

Es werden meistens auch Routen angegeben, welche vor der eingestellten Zeit Abfahren bzw. Ankommen.

Abfahrtsdatum: ändert man diesen Parameter, bestimmt man das Datum, an welchen Routen gesucht werden sollen. Zur Eingabe hilft eine passende Datumsanzeige.

Art der Verbindung: hier kann man zwischen drei Punkten entscheiden – „schnellste“, „wenig Umstiege“, „kurze Fußwege“. Dies berechnet die Routen, falls möglich, nach diesem Parameter. Oft unterscheiden sich die Ergebnisse kaum.

Beispiel, wo Eingaben und Optionen ausgenutzt werden:

**Zu Abfahrten** 16:10:30

Startpunkt  
Brixen, Bahnhof Brixen

Endpunkt  
Vernagt (Schnals), Vernagt

Weitere Optionen

Abfahrtszeit **Ankunftszeit**  
14:34

Abfahrtsdatum  
15.02.2022

Art der Verbindung  
☒ schnellste  
☐ wenig Umstiege  
☐ kurze Fußwege

**Verbindung suchen!**

Ausgabe:

Temperatur am Startort: 8.3 °C - Wolken  
Temperatur am Ankunftsort: 9.8 °C - Wolken

- R** > **251** > **261** 07:25 - 10:42, 03:17 Std, Von: Brixen, Bahnhof Brixen, 15.02.2022
- R** > **R** > **A 250B** > **R** > **261** 07:55 - 10:42, 02:47 Std, Von: Brixen, Bahnhof Brixen, 15.02.2022
- R** > **251** > **261** 08:25 - 11:42, 03:17 Std, Von: Brixen, Bahnhof Brixen, 15.02.2022
- R** > **261** 08:25 - 12:42, 04:17 Std, Von: Brixen, Bahnhof Brixen, 15.02.2022
- R** > **R** > **A 250B** > **R** > **261** 08:55 - 11:42, 02:47 Std, Von: Brixen, Bahnhof Brixen, 15.02.2022

Wie man erkennen kann, werden Routen ausgegeben, welche vor der angegebenen Zeit, ihr Ziel erreichen (auf der Ausgabe kommen noch weitere Verbindungen vor, ich habe nur kein Bild der kompletten Ausgabe eingefügt).

Durch Klick auf eine bestimmte Route werden Details angezeigt.

Klick auf den ersten Output oben:

**R** > **251** > **261** 07:25 - 10:42, 03:17 Std, Von: Brixen, Bahnhof Brixen, 15.02.2022 ^

**R 17129**  
Regionalzug

07:25-08:45, Von: Brixen, Bahnhof Brixen, Gleis: 2, Nach: Meran, Bahnhof Meran, Endstation: Meran, Bahnhof Meran

▼

 3 min, 7 min Umstiegszeit

**Bus**  
**251**

08:52-09:20, Von: Meran, Bahnhof Meran, Plattform: M, Nach: Naturns, Rathaus, Endstation: Naturns - Staben

▼

 1 min, 37 min Umstiegszeit

**Bus**  
**261**

09:57-10:42, Von: Naturns, Rathaus, Nach: Vernagt, Endstation: Kurzras

▼

Unter den Ausgaben befindet sich nun ein weiterer Button „Weitere Routen suchen“. Mit diesem können Routen ausgegeben werden, welche an einem späteren Zeitpunkt als der Vorgänger starten.

Beispiel (andere Route als vorher) – vor dem Klick auf den Button:

**320.1** > **RV** 17:31 - 18:29, 00:58 Std, Von: Albeins, Grundschule

**320.1** > **R** 18:01 - 18:59, 00:58 Std, Von: Albeins, Grundschule

**Weitere Routen suchen!**

Nach dem Klick – es werden zwei zusätzliche Routen abgezeigt:

**320.1** > **R** 18:01 - 18:59, 00:58 Std, Von: Albeins, Grundschule

**320.1** > **R** 18:31 - 19:21, 00:50 Std, Von: Albeins, Grundschule

**320.1** > **R** 19:01 - 19:59, 00:58 Std, Von: Albeins, Grundschule

**Weitere Routen suchen!**

## Erklärung von verschiedenen Ausgaben:

Temperatur am Startort: 8.0 °C - Wolken

Temperatur am Ankunftsort: 9.3 °C - Wolken

Das aktuelle Wetter des Startorts und Ankunftsorts wird ausgelesen und ausgegeben.  
Daten werden ausgelesen von openweathermap [später genauer]

**R** > **A 250B** > **R** > **261** 06:49 - 09:42, 02:53 Std, Von: Brixen, Bahnhof Brixen, 15.02.2022

**R** > **A 250B** > **R** > **261**

R: Regionalzug -> Züge sind schwarz gekennzeichnet

A -> vorübergehendes Ersatzverkehrsmittel, braun gekennzeichnet

250B Nummer des Ersatzverkehrsmittels

261 -> Nummer des Busses, Busse sind blau gekennzeichnet

„>“ steht für einen Umstieg zum nächsten Verkehrsmittel

06:49 - 09:42, 02:53 Std, Von: Brixen, Bahnhof Brixen, 15.02.2022

Abfahrtszeit-Ankunftszeit, Dauer, Abfahrtsort, Datum der Abfahrt

Bei Klick erhält man dieses Fenster, wo Informationen detailliert angezeigt werden:

**R** > **A 250B** > **R** > **261** 06:49 - 09:42, 02:53 Std, Von: Brixen, Bahnhof Brixen, 15.02.2022 ^

**R 1830**  
Regionalzug

06:49-08:15, Von: Brixen, Bahnhof Brixen, Gleis: 2, Nach: Meran, Bahnhof Meran,  
Endstation: Meran, Bahnhof Meran

⌵

 2 min, 4 min Umstiegszeit

**Andere -**  
Ersatzbus 250B

, 08:19-08:30, Von: Meran, Bahnhof Meran, Plattform: Z, Nach: Partschins, Bahnhof  
Töll, Endstation: Partschins, Bahnhof Töll

⌵

 0 min, 5 min Umstiegszeit

**R 7007**  
Regionalzug

08:35-08:42, Von: Partschins, Bahnhof Töll, Gleis: 1, Nach: Naturns, Bahnhof Naturns,  
Endstation: Mals, Bahnhof Mals

⌵


 2 min, 10 min Umstiegszeit

**Bus 261**

08:52-09:42, Von: Naturns, Bahnhof Naturns, Nach: Vernagt, Endstation: Kurzras

⌵

Die Ausgaben sollten selbsterklärend sein.

 2 min, Dieser Output zeigt die geschätzte Zeit an, um zur nächsten Station zu gehen, in diesem Fall 2 Minuten.

Bei einem weiteren Klick auf die jeweilige Verbindung werden alle Zwischenhalte mit Ankunftszeit angezeigt.

Beispiel:

**R 1830**  
Regionalzug 06:49-08:15, Von: Brixen, Bahnhof Brixen, Gleis: 2, Nach: Meran, Bahnhof Meran,  
Endstation: Meran, Bahnhof Meran ^

Dauer: 86 Min, Zwischenhalte: 12

Brixen, Bahnhof Brixen 06:49  
Klausen, Bahnhof Klausen 06:56  
Waidbruck, Bahnhof Waidbruck - Lajen 07:03  
Bozen, Bahnhof Bozen 07:23  
Bozen, Bahnhof Bozen Süd 07:39  
Bozen, Bahnhof Sigmundskron 07:45  
Terlan, Bahnhof Terlan - Andrian 07:52  
Gargazon, Bahnhof Gargazon 07:59  
Burgstall, Bahnhof Lana - Burgstall 08:03  
Meran, Bahnhof Meran Untermais 08:09  
Meran, Bahnhof Meran 08:15

### Beispiele von Fehlern:

Gib gültige Haltestellen ein.

Keine Verbindung gefunden!

## 2. Der zweite Punkt der Applikation ist die Abfahrtsanzeige.

Durch Klick auf den Button „Zu Abfahrten“ (im oberen Bereich des Fensters) wechselt man zu diesem Abschnitt. Der Text des Buttons wird sich ändern zu „Zu Verbindungen“. Man kann durch diesen Button nämlich wieder zurücknavigieren.

### Overview:


Haltestelle

Haltestelle	Eingabe der Haltestelle, an welcher Abfahrten gesucht werden		
-------------	--	--	--

NR	Start	Straßenseite	Ankunftsort
↑	↑	↑	↑
Nummer des Verkehrsmittel	Startzeit des Verkehrsmittels	Straßenseite, Plattform oder Gleis wo das Verkehrsmittel startet. Funktioniert nicht an allen Haltestellen.	Zielort, wo das jeweilige Verkehrsmittel hinfährt

Hier steht nur eine verpflichtende Eingabe zur Verfügung. „Haltestelle“ – hier gibt man die Haltestelle ein, von welcher Abfahrten angezeigt werden sollen. Auch hier gibt es eine Vervollständigung, damit man eine korrekte Eingabe erreicht. Bei Klick auf diese Vervollständigung werden Abfahrten gesucht. Es kann zu Fehlern kommen, es gibt keine wirklichen Fehlermeldungen, aber es werden keine Abfahrten angezeigt. Dies könnte auch passieren, falls von dieser Haltestelle momentan [in den nächsten zwei Tagen] keine öffentlichen Verkehrsmittel abfahren.

## Beispiel einer Ausgabe – Haltestelle: Brixen, Dantestraße:

 [Zu Verbindungen](#) 16:52:48

Haltestelle

Temperatur: 7,8 °C - Wolken

NR	Start	Straßenseite	Ankunftsort
320.1	16:54	Südseite	Milland, Zeffer
401	16:54	Nordseite	Bruneck, Busbahnhof
310	16:58	Südseite	Brixen, Bahnhof Brixen
328	17:00	Südseite	Brixen, Bahnhof
401	17:02	Südseite	Brixen, Bahnhof Brixen
401	17:02	Südseite	Brixen, Progress
170	17:03	Südseite	Sels, Busbahnhof
320.1	17:04	Nordseite	Vahrn, Post
170	17:08	Südseite	Kastelruth, Busbahnhof
320.1	17:09	Südseite	Albeins, Grundschule
310	17:15	Nordseite	Sterzing, Nordpark
310	17:15	Nordseite	Sterzing, Nordpark
328	17:17	Nordseite	Natz - Schabs
320.1	17:19	Nordseite	Vahrn, Post
321	17:21	Südseite	Brixen, Don Bosco
401	17:22	Nordseite	Mühlbach, Bahnhof Mühlbach

Oben wird das aktuelle Wetter ausgegeben, unten wird die Tabelle mit den nächsten 20 Abfahrten ergänzt (nicht alle Abfahrten sind auf dem Bild zu erkennen).

Die Ausgaben sollten selbsterklärend sein. Die Ausgaben werden alle 30 Sekunden aktualisiert. Ändert man die Eingabe der Haltestelle, so ändern sich die Ausgaben dementsprechend.

## Daten - verwendete APIs

### 1. Öffentliche Verkehrsmittel

Die Daten stammen von einer API, welche vom Land Südtirol zur Verfügung gestellt wird.

#### 1) Stopfinder Request

Um dem Benutzer Vorschläge (autocomplete) bei seiner Eingabe bieten zu können, wird die Liste dauernd aktualisiert. Die Daten welche dafür benötigt werden, werden mit einer fetch() Request vom Server geholt.

```
'https://efa.sta.bz.it/apb/XML_STOPFINDER_REQUEST?locationServerActive=1&outputFormat=JSON&type_sf=any&name_sf=' + text;
```

In der Variable urlStops wird diese URL gespeichert. Text ist die Eingabe des Users. Gibt der User „Brix“ ein, wird nach Haltestellen gesucht, welche „Brix“ enthalten.

```
fetch(urlStops).then(res => res.json()).then(data => { handleStops(data); }).catch(function () {return;});
```



Die fetch() Abfrage, Daten werden vom Server geholt und an eine Funktion weitergeleitet, welche die Daten in ein Array von JSON Objects speichert. Die Daten, welche gespeichert werden, sind der Name der Haltestelle und die ID der Haltestelle. Das Array sieht wie folgt aus (Daten sind nur Beispiele):

```
stops[0] = {
  stopName: "Bozen, Bahnhof",
  stopID: "602013"
}
```

## 2) Trip Requests

Diese Requests dienen dazu, um die Verbindungen von 2 Haltestellen auslösen zu können. In der Applikation wird dabei eine URL verwendet, welche zahlreiche Parameter enthält:

```
urlSud =
'https://efa.sta.bz.it/apb/XML_TRIP_REQUEST2?locationServerActive=1&sta
teless=%201&type_origin=any&name_origin=' + srcString +
'&type_destination=any&name_destination=' + destString + '&' +
arrOrDepTimeString + '&itdTime=' + startTime + '&itdDate=' + startDate
+ '&calcNumberOfTrips=' + howManyTrips + '&maxChanges=' + maxChanges +
routeTypeString +
'&useProxFootSearch=1&coordOutputFormatTail=4&outputFormat=JSON&coordOu
tputFormat=WGS84[DD.DDDDD]';
```

Um die Parameter zu verstehen, kann die offizielle Dokumentation der API durchgelesen werden.

<http://daten.buergernetz.bz.it/de/dataset/southtyrolean-public-transport>

Die Daten, welche ausgelesen werden, werden Abschließend in eine Klasse gespeichert. Jede Verbindung wird in die Klasse HoleTrip gespeichert, welche aus mehreren SubTrip Klassen besteht.

HoleTrip:

```
class HoleTrip {
  constructor(trip) {
    this.trip = trip;
    this.holeDuration = this.trip["duration"];
    this.subTrip = [];
  };
}
```

SubTrip – speichert zu jedem Abschnitt einer Route alle Zwischenschritte.

Ein HoleTrip besteht aus mehreren SubTrips.

```
class SubTrip {
  //evtl busseite usw
  constructor(tripId, duration, rtStartTime, srclatitude, srclongitude, destLatitude, destLongitude) {
    this.tripId = tripId;
    this.duration = duration;
    this.numberOfTransfer = numberOfTransfer; //busnr
    this.nameOfTransfer = nameOfTransfer;
    this.srcStation = srcStation;
    this.destStation = destStation;
    this.lastDesination = lastDesination;
    this.startTime = startTime;
    this.arrivalTime = arrivalTime;
    this.typeOfVehicle = typeOfVehicle; // 0 zug, 5-7 bus, 8 seilbahn; more on documentation
    this.startDate = startDate;
    this.arrivalDate = arrivalDate;
    this.trainPlatformStart = trainPlatformStart; //
    this.trainPlatformEnd = trainPlatformEnd; //
    this.timeOfNextBus = timeOfNextBus;
    this.timeToTransferinMinutes = timeToTransfer;
    this.expectedTransferTime = expectedTransferTime;
    this.stops = stops.slice();
    this.depTimes = allStopDepTimes.slice();
    this.arrTimes = allStopArrivalTimes.slice();
    this.rtStartTime = rtStartTime;
    this.rtArrivalTime = rtArrivalTime;
    this.srcStationLatitude = srclatitude;
    this.srcStationLongitude = srclongitude;
    this.EndStationLatitude = destLatitude;
    this.EndStationLongitude = destLongitude;
  }
}
```

Die Ausgabe wird direkt mit JavaScript generiert, es werden keine Frameworks benutzt. Dazu kann man sich die Funktion buildFrontend(holeTrip) genauer anschauen.

### 3) Abfahrt Requests

Abfahrt Requests an die API sind weniger komplex, es werden dementsprechend weniger Daten ausgelesen.

Es wird ein fetch() auf folgenden Link ausgeführt: Für die Variable station wird die stopID übergeben (funktioniert auch mit dem stopName, dies führt bei einigen Stationen aber zu einem Fehler)

```
let haltestellenLink =
"https://efa.sta.bz.it/apb/XML_DM_REQUEST?&locationServerActive=1&outputFormat=JSON&stateless=1&type_dm=any&name_dm="+station+"&mode=direct&coordOutputFormatTail=4&outputFormat=JSON&coordOutputFormat=WGS84[DD.DDDD]";
```

Die Daten werden in Arrays gespeichert und anschließend in JavaScript in eine Tabelle hineingeschrieben, welche dem User angezeigt wird.

### 4) Wetter Requests

Die Wetter Requests werden mit der API von openweather durchgeführt.

<https://openweathermap.org/api>

Durch die Benutzerfreundlichkeit der API hat man schnell aktuelle Temperatur und aktuelles Wetter abgelesen.  
Dem Link werden einfach die Koordinaten der Haltestelle übergeben und schon hat man, was man braucht.

```
const openWeather = await  
fetch('https://api.openweathermap.org/data/2.5/weather?lat=' + latitude  
+ '&lon=' + longitude +  
'&appid=dc704448494ba8187b5e3cf65aafac7f&units=metric')
```

Dies wird in der Funktion findTempByCoord(longitude, latitude) durchgeführt.  
Bei Fehlern bei der Request wird entsprechend gehandelt.

## Verwendete Librarys

Wir verwenden die Bibliotheken jQuery und underscore.js für hilfreiche Funktionen, für die Frontend Gestaltung hilft uns Bootstrap 5.

jQuery ist vor Allem bei der Funktion der Autovervollständigung der Eingabefelder im Einsatz. Für unseren Einsatzzweck eignet es sich perfekt!

<https://jqueryui.com/autocomplete/>

### Startpunkt

Brixen|

Albeins (Brixen), Grundschule

Brixen, Tschuggmall Schule

Brixen, Bahnhof Brixen

Brixen, Busbahnhof

Brixen, Millanderau

Afers (Brixen), Palmschoss

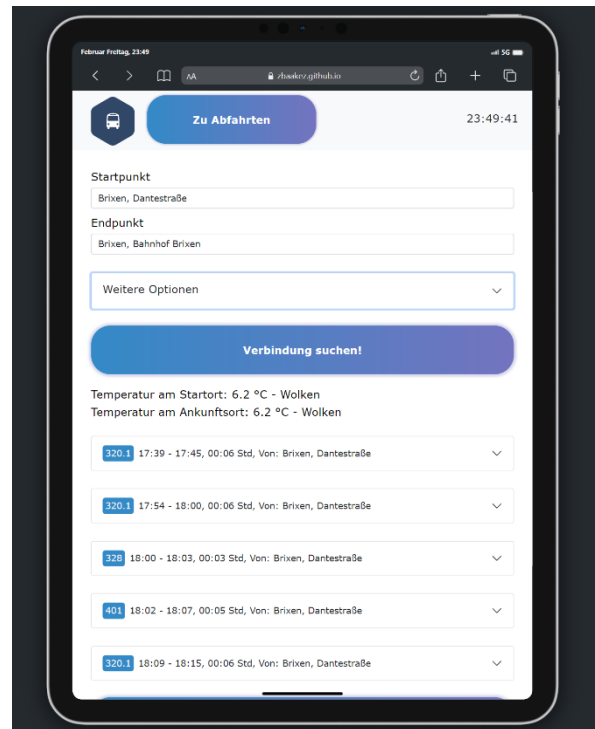
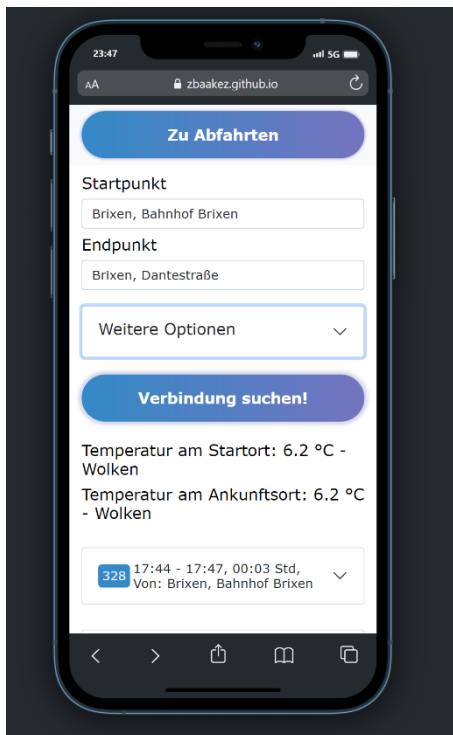
Brixen, Dantestraße

Brixen, Krankenhaus

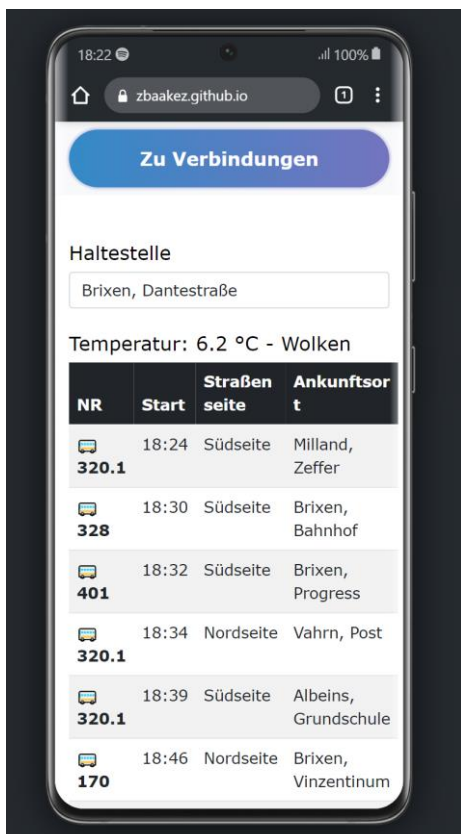
Brixen, Säbener Tor

Brixen, Maria-Hueber-Platz

Das minimalistische und ansprechende Bootstrap Design zieht sich über die ganze Seite. Durch Bootstrap und einigem an CSS erreichen wir eine fast perfekte „Responsivness“ für alle Geräte (Mobilgeräte, Laptops, Große Bildschirme, ...).  
Beispiele: am iPhone 12 PRO und iPad PRO 11



Auch die Tabelle des Abfahrtstables wird angepasst angezeigt (Galaxy S 21 Ultra):



Diese korrekten Anzeigen waren uns wichtig, da so eine Applikation hauptsächlich von Smartphone Usern verwendet wird.

Außerdem sollte alles benutzerfreundlich sein.

Um die Smartphone Benutzung weiter zu optimieren haben wir ein Swipe-Feature eingebaut. Wenn man nach rechts swipet kommt man zur Abfahrtstabelle, nach links zur Verbindungssuche. Dazu verwenden wir ein kleines Script von GitHub:

<https://github.com/john-doherty/swiped-events>

Am besten probieren Sie dieses Feature selbst!

## Hosting und Backups

Für beides haben wir uns für GitHub entschieden. Durch Hosting mit GitHub war die Website innerhalb weniger Minuten öffentlich im Internet erreichbar.

Für Backups und Zusammenarbeit haben wir die übliche Plattform GitHub benutzt.

## Probleme bei der Entwicklung

Die Probleme in der Entwicklung waren überschaubar. Typische CSS-Probleme, aufgrund von mangelndem Wissen und Datenprobleme mit der API für die Öffentlichen Verkehrsmittel, diese ist nämlich öfters für kurze Zeit nicht online, oft ist sie sehr langsam, was das Testen der Neuerungen erschweren kann.

Auch haben wir von der Südtiroler Wetter API auf die openweather API gewechselt, da diese sich besser eignet um das Wetter an bestimmten Koordinaten auslesen.

Bei einem größeren Projekt würden wir von nun an auf ein Framework für das Frontend setzen (z.B. React), dies würde die Frontend Entwicklung vereinfachen und den Code lesbarer machen.

## Updates

Bei dem Projekt könnte man noch lange weiterarbeiten und zusätzliche Funktionen einbauen. Vor allem eine Karte, welche die gewählte Route anzeigt, wäre nützlich. Als Orientierung könnte man die offizielle Applikation von Südtirol verwenden.

Jedoch sind größere Updates momentan nicht geplant, vielleicht ändert sich dies aber.

## Abschluss

Wir haben eine Applikation geschaffen, welche einem jeden von uns brauchbare Daten liefern kann. Diese ist ansprechend gestaltet und läuft auf jedem Gerät. Trotzdem kann es immer wieder zu Bugs

kommen, deshalb würden wir uns wünschen, wenn diese gleich gemeldet werden. Am besten auf GitHub oder direkt über die Kontakt-E-Mail, welche im Footer(unterer Teil der Seite) der Seite angegeben ist. Auch bei anderen Anmerkungen können Sie sich gerne an uns wenden.

***Philipp Olivotto / Noel Rovara***

**11.02.2022**

<https://zbaakez.github.io/>