

GoTo - Technische Dokumentation

Martin Hochstrasser

27.06.2019

Inhaltsverzeichnis

GoTo.Service	1
Architektur	1
REST-Schnittstelle	2
Deployment	2
GoTo.Lambda	2
Architektur	2
Anbindung an GoTo.Service	2
GoTo.Alexa	3
Intents	3
Permissions	3
GoTo.Client	3
Architektur	3
Anbindung an GoTo.Service	3

GoTo.Service

Die zentrale Schnittstelle von *GoTo* ist mithilfe von ASP.NET Core implementiert. Der Service wird auf *AWS Elastic Beanstalk* deployt als Docker-Container.

Architektur

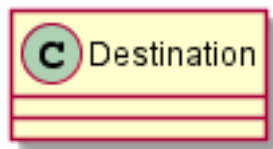


Abbildung 1: Klassendiagramm von *GoTo.Service*

REST-Schnittstelle

Destination		▼
GET	/api/destination Query all available destinations	
TripOffer		▼
GET	/api/tripoffer Query all available trips offered by users	
POST	/api/tripoffer Adds a new offer for a trip	
TripSearch		▼
POST	/api/tripsearch	
User		▼
GET	/api/users Query all registered users	

Abbildung 2: REST-Schnittstelle von *GoTo.Service*

Deployment

GoTo.Lambda

Architektur

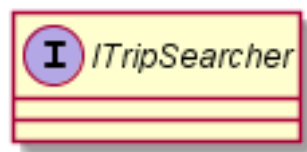


Abbildung 3: Klassendiagramm von *GoTo.Lambda*

Anbindung an GoTo.Service

Die Anbindung erfolgt über simple HTTP-Anfragen, die intern aber nicht gecacht werden.

GoTo.Alexa

Intents

Permissions

GoTo.Client

Angular

Bootstrap

ng-bootstrap

Architektur

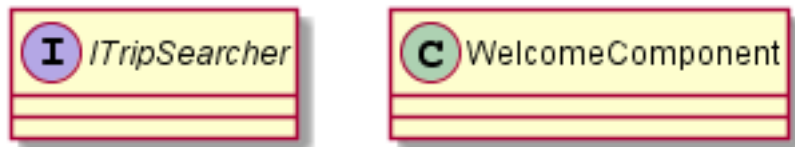


Abbildung 4: Klassendiagramm von *GoTo.Client*

Anbindung an GoTo.Service

Der GoTo.Service erzeugt beim Bauen eine `swagger.json`, aus der dann ein Client generiert werden kann. Um dies möglichen einfach zu machen, wird Docker benützt um `swagger-codegen` zu starten. Dies übernimmt das Skript `GoTo.Client/Update-ApiClient.ps1`.

Der generierte Client ist dann unter `GoTo.Client/src/api-client` verfügbar und dieser kann direkt in Angular verwendet werden, da ein eigenes Angular-Modul generiert wird.

Der benützte Server ist im `environment` hinterlegt und ist beim Entwickeln gesetzt auf den lokalen Host und im Live-Betrieb ist diese auf `goto.eu-west-1.elasticbeanstalk.com` gesetzt.