# http Endpunkte:

## AlleController

/// Alle Anrufe, in der Datenbank zurückgeben. Max: DefaultLimit

[HttpGet("/Alle/")]

/// Alle Anrufe, in der Datenbank zurückgeben. beginnent bei Eintrag: Skip Max: DefaultLimit

[HttpGet("/Alle/{skip}")]

public IActionResult GetAllDefLimit(int skip)

{

return GetAll(skip, DEFAULTLIMIT);

}

/// Alle Anrufe, in der Datenbank zurückgeben. beginnent bei Eintrag: Skip Max: limit, limit ist Limitiert auf MaxLimit

[HttpGet("/Alle/{skip}/{limit}")]

## EntryController

/// Erzeugt einen Neuen Eintrag ohne Captcha Abfrage

[HttpGet("/Entry“)]

## FrameController

/// Seite um einen Neuen Eintrag in die Datenbank zu bringen.

/// Designt zur einbindung als iFrame.

/// Geschützt durch captcha

[HttpGet("/Frame/Add")]

/// Gibt die Grafik des Captcha zurück.

/// einbindung über img src

[HttpGet("/Frame/Captcha/{id}")]

/// Erzeugt oder Bearbeitet einen Eintrag im iFrame. Gbit zusätzlich eine Fehlermeldung Zurück.

/// Wird Verwendet um bei einem Captcha Fehler, den Fehler zurück zu geben

public IActionResult AddFrame(Entry entry, string msg = null)

/// <summary>

/// Speichert einen Einen Neuen Eintrag

/// </summary>

/// <param name="id">Muss eine leere id sein, Bearbieten noch nicht Möglich</param>

/// <param name="phone">Telefennummer</param>

/// <param name="request">siehe enum EntryRequest</param>

/// <param name="zip">Postleitzahl</param>

/// <param name="captchasecret">Hoffentlich gelöste Captcha</param>

/// <param name="captchaid">die id unter wlecher des Captcha in der Datenbank geführt wird</param>

/// <returns></returns>

[HttpPost]

public IActionResult Send(string id, string phone, EntryRequest request, string zip, string captchasecret, string captchaid)

/// Status des Enpunktes abrufen gibt einfach einen nicht cachebares 200 Zurück.

/// Wird verwendet, damit der Load Balancer die Anwendung als online erkennt.

/// ist im Frame Controller, da es hier keine httpauth gibt

[ResponseCache(NoStore = true, Location = ResponseCacheLocation.None)]

[HttpGet("/Frame/CheckStatus")]

public IActionResult CheckStatus()

## HomeController

/// Gibt eine ÜBersichtsseite Zurück, welche alle einträge ohne Zip Nummer enthält

[HttpGet]

public IActionResult Index()

/// Speichert einen Bearbeiteten Eintrag

/// Es kann sich um einen existierenden sowie einen neuen Eintrag handeln

/// <param name="id">id des eintrages, welcher gesucht werden soll</param>

/// <param name="phone">telefonnummer</param>

/// <param name="request">siehe enum EntryRequest</param>

/// <param name="zip">Postleitzahl</param>

[HttpPost]

public IActionResult Index(string id, string phone, EntryRequest request, string zip)

## OrganizationController

/// Gbit eine Suchmaske bzw. Alle Organisationen zurück

public IActionResult Index()

/// Gbit die eingabemaske für eine neue ORganisation zurück

[HttpGet("/Organization/Add")]

/// Gbit die eingabemaske für eine neue Organisation zurück

/// <param name="id">id der ORganisation</param>

[HttpGet("/Organization/Add/{id}")]

/// Gibt Alle Organisationen, in der Datenbank zurück

[HttpGet("/Organization/Search/")]

public IEnumerable<OrganizationTrasport> SearchOrganisation()

/// Gibt alle ORganisationen für einen anfang einer bestimmten PLZ Zurück

[HttpGet("/Organization/Search/{search}")]

public IEnumerable<OrganizationTrasport> SearchOrganisation(string search)

/// Dreht die suche um, Fallback Leere Suche

[HttpGet("/Organization/SearchRev/{zipreserve}")]

/// Sucht nach einer Organisation mit einem bestimmten plz

/// <param name="zipreserve">Bestimmt ob nach match oder anfang gesucht werden soll</param>

/// <param name="search">Anfang oder Match der PLZ</param>

[HttpGet("/Organization/SearchRev/{zipreserve}/{search}/")]

/// Eine neue bzw. zu bearbeitende Organisation mit der Möglichkeit einen Fehler zurück zu geben

public IActionResult AddOrganization(Organization entry, string msg = null)

/// Speichert eine neue oder Bearbeitete Organisation

[HttpPost]

public IActionResult Send(string id, string name, string ansprechpartner, string email, string zip, int NotifyRequest1, int NotifyRequest2, int NotifyRequest3, int NotifyRequest4)

# SignalR Endpunkte

Signal r ist eine Abstraktionsschicht über eine Server Client Kommunikation.

In den meisten Fällen werden websockets verwendet.

Fallback auf Longpolling bzw. was halt tut 😉

<https://dotnet.microsoft.com/apps/aspnet/signalr>

/// Markiert einen Eintrag als nicht mehr bearbeitet.

public Task FreeEntry(string id)

/// Markiert einen Eintrag als bearbeitet.

public Task MarkEntry(string id)

/// Speichert, eine neuen eintrag, wird ein verwendeter eintrag gespiechert, wird ein neuer mit altem timestamp erzeugt.

public Task AddOrModifyEntry(string id, string phone, string zip, string request)

/// Löscht einen Enitrag aus der Datenbank

public Task DeleteEntry(string id)

## Funktionen der Datenbank

/// Eine Bearbeitete Organisation Finden und Ersetzen

internal void UpdateOrganization(Organization entry)

/// Findet eine Orgnisation über die id

internal Organization FindOrganization(string id)

internal Organization FindOrganization(ObjectId id)

/// Findet ein Captcha Element in der Datenbank

internal Captcha GetCaptcha(string id)

/// Fügt ein Captcha Element in die Datenbank hinzu

internal void AddCaptcha(Captcha captcha)

/// Löscht einen einzelnen eintrag in der Datenbank

/// Dies passiert wenn das captcha erfolgreich "verwendet" wurde

internal void RemoveCaptcha(Captcha captcha)

/// Löscht Alle Captchas aus der Datenbank, deren Laufzeit abgelaufen ist.

internal void CleanupCaptcha()

/// Fügt eine Organisation in die Datenbank hinzu

internal void AddOrganization(Organization entry)

/// Listener Für Änderungen in der Entry Datenbank

/// Sorgt mit hilfe von SignalR dafür, das die Frontends aktualsiert werden.

/// Prüft ob es eine Organisation gibt, welche eine benachrichtigung z.b. über email aboniert hat und versendert die.

/// Es ist auch möglich sich auf die PLZ 00000 zu Registrieren.

private async void Listen()

/// Gibt alle Einträge Sortiert zurück

/// <param name="skip">Erstes Element</param>

/// <param name="limit">Anzahl der Elemente</param>

public List<Entry> GetAll(int skip, int limit)

/// Anzahl aller einträge in der datenkbank

public long CountAll()

/// Gibt alle einträge zurück, welche keine PLZ Besitzen

public List<Entry> GetNoZip()

/// Anzahl aller Einträge welche keine plz besitzen

public long CountNoZip()

/// Anzahl aller Anrufe in den letzten 60 Minuten

internal long CountCallHour()

/// Anzahl im Frontend Bearbeiteten Einträge in der Letzten Stunde

internal long CountEditHour

/// Anzahl aller Anrufe in den letzten 24 Stunden

internal long CountCallDay()

/// Anzahl im Frontend Bearbeiteten Einträge in der Letzten 24 Stunden

internal long CountEditDay()

/// Löscht einen Telefon Anruf

internal void Remove(ObjectId id) => Remove(Find(id));

internal void Remove(Entry entry)

/// SPeichert einen Neuene Anruf in der Datenbank

internal void Add(Entry entry) /// <summary>

/// Versucht einen Neuen Eintrag in die Datenbank zu Speichern.

/// Die Software ist darauf ausgelegt auf mehreren Servern gleichzeitig zu laufen

/// in der Methode Listen werden Alle instanzen dieser Software auf einmal informiert.

/// die Datenbank hat einen Unique Index auf die relevanten Felder (siehe Konstruktor)

/// Die Schnellste Instanz wird den Eintrag einfügen können die restlichen bekommen eine Exeption

/// Bei Exception wird false zurück gegeben und bei erfolg true. So wird die Benachrichtigung nur vom Schnellsten Server versendet.

public bool TryAddNotifkation(Notifikation notifikation)

/// Findet einen Telefonanruf

internal Entry Find(ObjectId id)

/// Gibt alle Organisationen Zurück

internal IEnumerable<Organization> GetOrganisations()

/// Ersetzt Einen Telefonanruf in der datenbank

internal void Replace(Entry entry)

## Benachrichtigung:

Zu versendene Benachrichtgungen Landen in der NotifikationFactory

Hier ist allerdings noch nichts Implementiert.