IXchange

Entwicklerdokumentation

Dokumentation | Call 17 | Projekt ID 6399

Inhalt

1 Einleitung

Das Projekt IXchange hat das Ziel, die gemeinschaftliche und nachhaltige Nutzung von IoT Daten und Sensoren durch die Schaffung einer ökonomischen IoT Plattform zu fördern. Hierbei können Smart-Home Besitzer, Unternehmen und Kommunen Daten von IoT Geräten teilen und auf bereitgestellte Daten zugreifen. Diese Entwicklerdokumentation gibt einen Überblick über die Funktionen des im Rahmen des Netidee-Projektes „IXchange“ entwickelten Software-Bibliotheken.

1.1 Architektur des Gesamtsystems

Das Gesamtsystem von IXchange gliedert sich grundlegend in drei Softwaremodule: eines für den Server, eines für die Clients und eines für die Kommunikation. Der Server fungiert als zentraler Knotenpunkt zur Verwaltung des Gesamtsystems und zur Datenhaltung. Das Modul für die Clients umfasst sowohl Softwaremodule für Benutzeranwendungen als auch für den IoT-Teil von IXchange. Sowohl das Servermodul als auch das Modul für die Clients nutzen das Kommunikationsmodul für den Datentransfer und stellen darüber hinaus Schnittstellen für externe Systeme bereit. Dieser Aufbau ist in **Fehler! Verweisquelle konnte nicht gefunden w erden.** dargestellt.

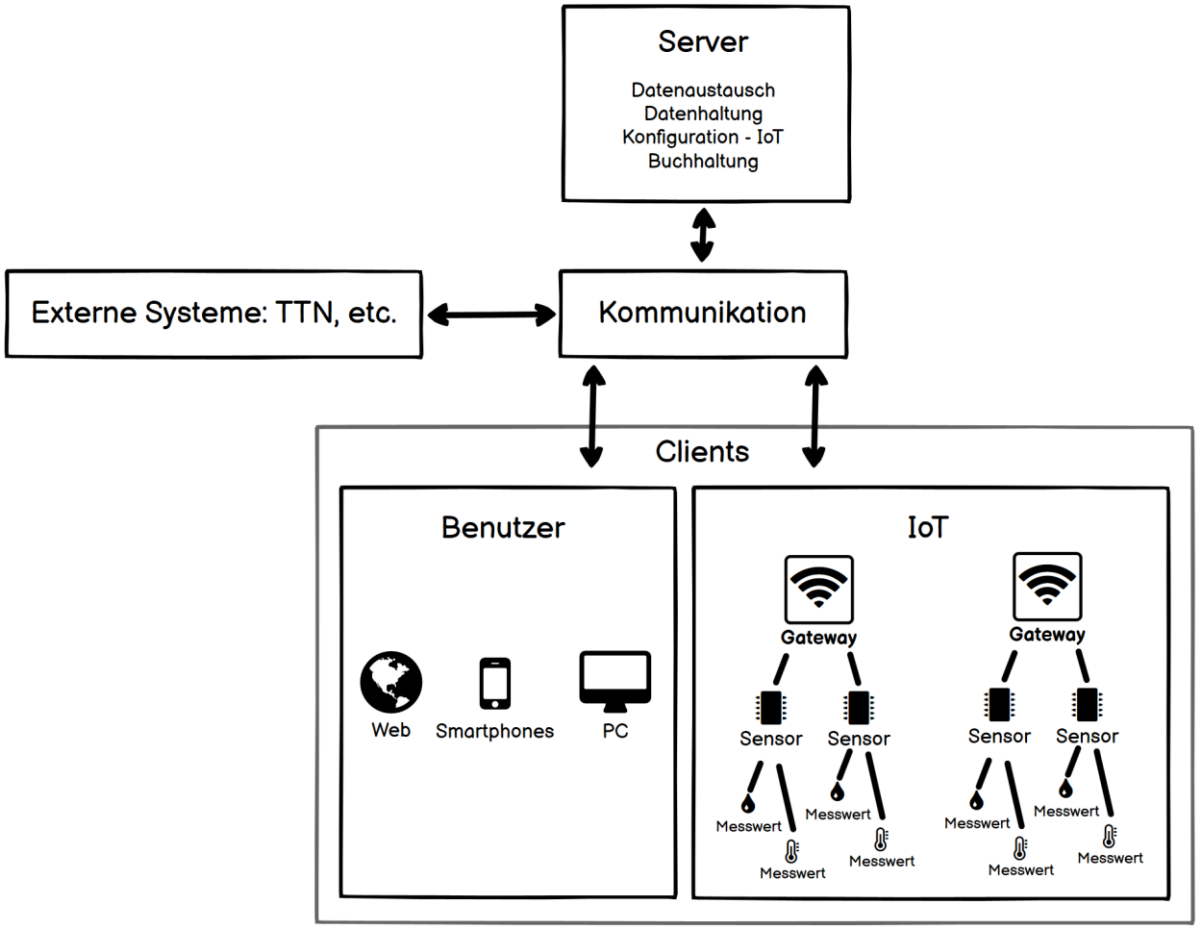

Abbildung 1 Architektur des Gesamtsystems

1.2 Allgemeines/Struktur und Aufbau

Es wurden zahlreiche Software-Projekte realisiert (siehe Abbildung 1 Architektur des Gesamtsystems), die in den kommenden Abschnitten detailliert erläutert werden. Diese dienen als Grundlage und Referenz für zukünftige Entwicklungen. Die Architektur des Gesamtsystems, wie in Abbildung 1 Architektur des Gesamtsystems dargestellt, wird im Source Code angenähert. Dies umfasst den „IXchange.Apps“ Ordner mit „BlazorApp“, „AndroidApp“, „IOsApp“, „WpfApp“ und deren zugehörigen *Views* und *ViewModels*. Im darauffolgenden Abschnitt „IXchange.Communication“ sind die Komponenten für die Kommunikation, darunter „REST“, „GRPC“ und „SignalR“, zu finden. Der Ordner „IXchange.Exchange“ bildet die Schnittstelle zwischen den Projekten. Im „IXchange.Gateway“-Abschnitt befindet sich das *Gateway*, das die Kommunikation zwischen Sensoren und Server ermöglicht. Abschließend spiegelt der Abschnitt „IXchange.Server“ den Server inklusive Datenbank für die Datenhaltung wider.

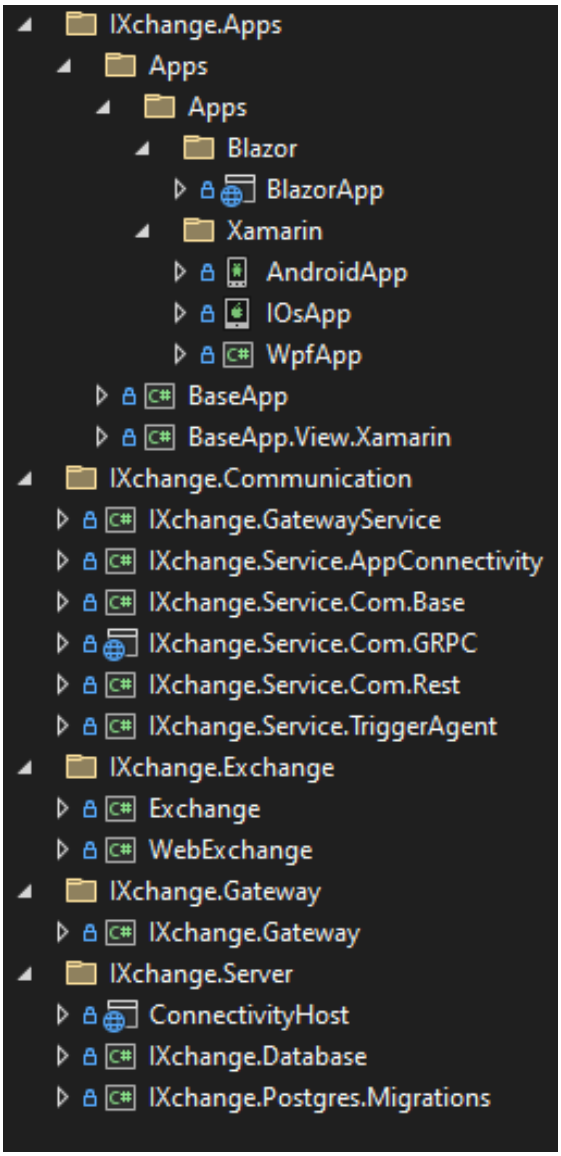

Abbildung 2 Projekt Struktur

1.3 Source-Code

Der Source-Code befindet sich unter

https://github.com/FotecGmbH/FOTEC-IXchange

Der Source-Code ist dokumentiert, detaillierte Informationen über einzelne Funktionen

kann somit direkt im Source-Code nachgelesen werden.

2 IXchange Komponenten

In diesem Abschnitt wird eine umfassende Übersicht über die verschiedenen Projekte, welche die Grundlage für die Funktionalität von IXchange bilden, geboten. Die Schlüsselkomponenten werden hierbei näher erklärt, um Entwicklern einen fundierten Einblick in die Architektur und Implementierung zu geben.

2.1 IXchange.Apps

BlazorApp

Die „Blazor App“ beinhaltet alle *pages*, inklusive den dafür benötigten *modals, components* und *converter,* um das *Graphical User Interface* über Web darzustellen.

Die Url der WebApp kann in dem Ordner „Properties“ des Projektes, in der Datei „launchSettings.json“ umgestellt werden.

AndroidApp

Das Projekt „AndroidApp“ ermöglicht die Ausführung auf Android Geräten. Die *Views* hierzu befinden sich in dem Projekt „BaseApp.View.Xamarin“ und die *ViewModels* in dem Projekt „BaseApp“.

IOsApp

Die Logik und Ansichten für die iOs-Anwendung sind vorbereitet, jedoch noch nicht finalisiert und getestet. Die Finalisierung sollte jedoch in einem akzeptablen Rahmen liegen und nicht übermäßig zeitintensiv sein. (Stand 12/2023)

Die *Views* hierzu befinden sich in dem Projekt „BaseApp.View.Xamarin“ und die *ViewModels* in dem Projekt „BaseApp“.

WpfApp

Das Projekt „WpfApp“ ermöglicht die Ausführung auf Windows Geräten. Die *Views* hierzu befinden sich in dem Projekt „BaseApp.View.Xamarin“ und die *ViewModels* in dem Projekt „BaseApp“.

BaseApp

Das Projekt „BaseApp“ ist die „ViewModel-Schicht“ von IXchange und wird in Abbildung 3 BaseApp Struktur dargestellt. Der Abschnitt „Connectivity“ mit der Klasse *DcProjectBase* bietet die Sammlung der Datenpunkte für die Kommunikation über SignalR. Der Abschnitt „Templates“ bietet die Vorlagen für die verschiedenen Darstellungsmöglichkeiten. Der Abschnitt „ViewModels“ ist der größte Teil des Projektes und umfasst alle *ViewModels* des Projektes. Diese sind für die Aufbereitung der gespeicherten Daten, welche von der Datenbank, über den „ConnectivityHost“, über die *DcProjectBase* zur Benutzeroberfläche transportiert werden. Für jede Ansicht/*View* gibt es ein eigenes *ViewModel*, welches die genannten Daten für die Darstellung und Interaktion aufbereitet.

Die Klassen *VmProjectBase, VmProjectBaseCommands* und *VmProjectBaseConnectivity* bilden gemeinsam die Klasse *VmProjectBase,* von welcher jedes *ViewModel* erbt. Dies bildet die Grundfunktionalität eines *ViewModels* inklusive der Möglichkeit für die Kommunikation und die Navigation. Die Klasse *VmProjectBaseCommands* beinhaltet die Menüpunkte und bietet somit die Möglichkeit von jedem *ViewModel* aus zu jedem Menüpunkt zu navigieren.

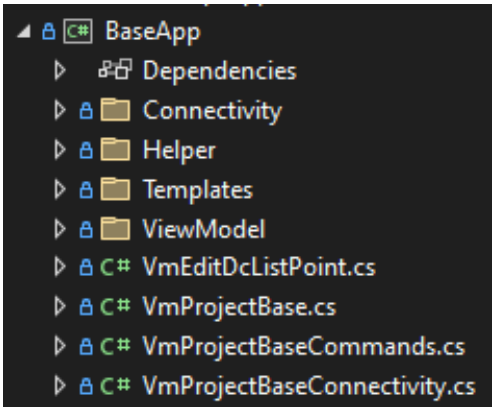

Abbildung 3 BaseApp Struktur

BaseApp.View.Xamarin

Das Projekt „BaseApp.View.Xamarin“ ist die „View-Schicht“ von IXchange für alle Darstellungen, welche Xamarin verwenden. Konkret sind dies die Darstellungen über:

- Windows/Wpf
- Android

- IOs

Der Abschnitt „View“ ist hierbei der größte Teil des Projektes und umfasst alle *Views,* abgesehen von den *pages* für die Web-Darstellung. Für die *Views* sind *Converter* für spezielle Darstellungen in dem Abschnitt „Converter“ und eigene *Controls* in dem Abschnitt „Controls“.

2.2 IXchange.Communication

IXchange.GatewayService

Das Projekt „IXchange.GatewayService“ ist für die Kommunikation des *Gateways* über SignalR zuständig. Die Klasse „GatewayHub“ inkludiert hierbei die Verbindung zum Server. Die Klasse „GatewayHub“ hat zusätzlich noch ein Feld für eine Implementierung des Interfaces „IGatewayConnectedClientsManager“. Dieses beinhaltet die Funktionen für das Senden einer Änderung der Konfiguration an das *Gateway*.

IXchange.Service.AppConnectivity

Das Projekt „IXchange.Service.AppConnectivity“ ist für die Speicherung der Datenpunkte, welche über SignalR transportiert werden, verantwortlich. Der Hauptteil dieses Projektes befindet sich in dem Abschnitt „DataConnector“ und beschreibt für jeden Datenpunkt, wie die Daten in der Datenbank gespeichert werden. Dies beinhaltet die Umkonvertierung von Modell-Klassen zu Tabellen-Klassen, die zusätzliche Speicherung von Klassen mit Beziehungen und die Notifizierung von *Clients*, welche die Änderung betrifft.

IXchange.Service.Com.Base

Das Projekt „IXchange.Service.Com.Base“ bietet die Basis für die Kommunikation. Konkret befinden sich hier *Authorization*-Attribute, Modell-Klassen und weitere *Helper*-Klassen, welche von mehreren Kommunikationen verwendet werden können. Mit dieser Basis, kann IXchange einfach um ein weiteres Kommunikationsprotokoll erweitert werden.

IXchange.Service.Com.GRPC

Die Logik der GRPC-Schnittstelle ist vorbereitet, jedoch noch nicht finalisiert und getestet. Die Finalisierung sollte jedoch in einem akzeptablen Rahmen liegen und nicht übermäßig zeitintensiv sein. (Stand 12/2023) Inkludiert sind hierbei die „.proto“-Dateien und die *Services* für den Abruf.

IXchange.Service.Com.Rest

Das Projekt „IXchange.Service.Com.Rest“ umfasst die *Controller* für die REST-Schnittstelle, einschließlich der OData-Schnittstelle. Die *Controller* bieten hierbei verschiedene Möglichkeiten für das Erhalten und Verwalten der Daten aus der Datenbank. Die *Authorization* der Rest-Schnittstelle erfolgt über Das „IXChangeAuthorize“-Attribut, welches in dem Ordner „IXchange.Service.Com.Base/Helpers“ zu finden ist. Bei dem Abruf der Endpunkte ist der Token, welcher von einem Benutzer über das eigene Profil erstellt werden kann, als *Bearer*-Token anzugeben. Konkret werden hierbei folgende Rest-*Controller* geboten.

- IoT-Device
    a. GET api/iotdevice/list => Listet alle IoT Geräte die der Benutzer besitzt oder von dessen der Benutzer eine Messwerdefinition aboniert hat
- MeasurementDefinition
    a. GET api/measurementdefinition/list => Informationen aller Messwertdefinitionen abfragen, die der Benutzer besitzt oder aboniert hat
    b. GET api/measurementdefinition/{id} => Informationen über eine Messwertdefinition abfragen
- MeasurementResult
    a. GET api/measurementresult/{id} => Details zu einem Messergebnis
    b. GET api/measurementresult/query/{id}/{take}/{skip} => Abfrage für Messungen
    c. GET api/measurementresult/timeperiod/{id}/{fromDate}/{toDate} => Abfrage für Messungen innerhalb eines Zeitraums
    d. POST api/measurementresult/create/{definitionId} => Messergebnis hinzufügen
    e. DELETE api/measurementresult/delete/{definitionId} => Messergebnisse löschen bis bestimmtes Datum

Das Projekt kann einfach um zusätzliche *Controller* und Endpunkte erweitert werden. Die OData Schnittstelle wird in dem Kapitel „Open Data Schnittstelle“ genauer erläutert.

IXchange.Service.TriggerAgent

Das Projekt „IXchange.Service.TriggerAgent“ wird für Benachrichtigungen bei Änderungen der Konfiguration und des Verbindungsstatus des *Gateways* vom Server verwendet.

2.3 IXchange.Exchange

Exchange

Das Projekt Exchange, ist die Schnittstelle zwischen den Projekten und ist in Abbildung 4 Exchange Struktur dargestellt. Darin befindet sich der Großteil der Klassen, die von mehreren Projekten benötigt werden. Neben den selbsterklärenden „Enum“-, „Extensions“- und „Interfaces“-Ordner, befindet sich im „GatewayService“ Ordner, beziehungsweise in der „GatewayConstants.cs“ Klasse, austauschbare Informationen über den *SignalR-Hub*.

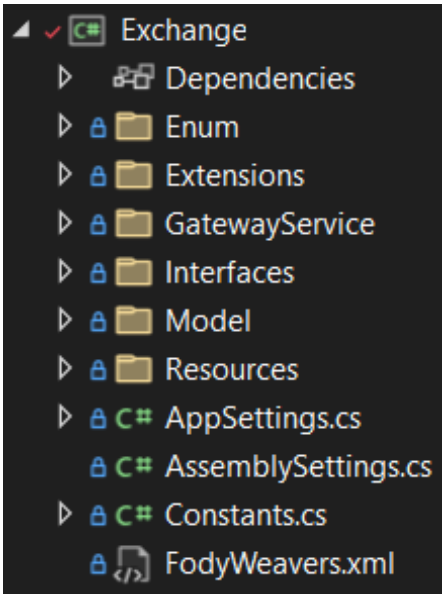

Abbildung 4 Exchange Struktur

In dem Ordner „Model“ befinden sich die Modelle für die einfachere Verarbeitung der Daten. Diese sind von den Datenbank-Tabellen abstrahierte Klassen und beinhalten zum Beispiel kalkulierte *properties*, welche das Darstellen der Klassen erleichtern.

In dem Ordner „Resources“ befinden sich die Ressourcen wie *fonts*, Bilder, Sprachen und Glyphen. Des Weiteren befinden sich in dem Ordner alle .resx Dateien, welche für die Texte in der gesamten Applikation verwendet werden.

WebExchange

Das Projekt „WebExchange“ ist für den Austausch von Konfigurationen, wie dem Datenbank-*Connection-String,* zwischen den Projekten zuständig und ist in Abbildung 5 WebExchange Struktur dargestellt. Aktuell sind die in der Klasse „WebSetting.cs“ angezeigten Konfigurationsmöglichkeiten umgesetzt, jedoch kann die Klasse nach Bedarf beliebig erweitert werden. Die „Interfaces“ geben hierbei *properties* vor, welche von „WebSetting.cs“ implementiert werden müssen.

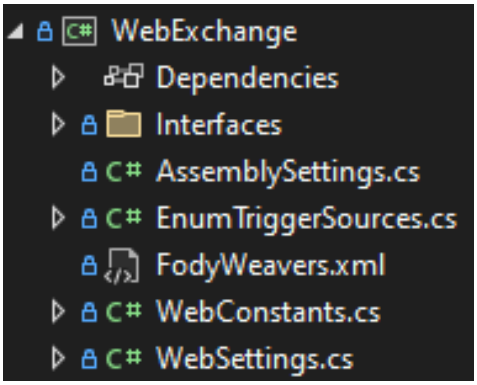

Abbildung 5 WebExchange Struktur

2.4 IXchange.Gateway

IXchangeGateway

Das Projekt „IXchangeGateway“ ist das Bindeglied zwischen Sensoren und Server und transportiert die Daten zwischen diesen. Die Funktionalitäten des Gateways werden hauptsächlich von den Bibliotheken von „BDA“ zur Verfügung gestellt, worauf in dem Kapitel „BISS Data Agent“ eingegangen wird. Für das *Debuggen*, sowie das Veröffentlichen des *Gateways* ist die Umstellung der IP-Adresse wichtig, welche im Kapitel „Umstellung der IP-Adresse für Gateways“ erläutert wird.

2.5 IXchange.Server

ConnectivityHost

Das Projekt „ConnectivityHost“ beschreibt den ausführbaren Server des Systems. Der Server ist unter Windows und Linux ausführbar und kommuniziert über SignalR zu den Gateways.

Unter dem Ordner „Helper“ befindet sich die Klasse „BakgroundIxiesWorker“. Diese ist für die Verbuchung der IXies zuständig, arbeitet in einem eigenen *Task* und handelt fällige Verbuchungen nach einem fixen Intervall ab. Hierbei wird zwischen einer schnellen und einer langsamen Aktualisierungsrate unterschieden. IXies für die Erstellung eines Kommentars, für die Erstellung eines Accounts und für die Erstellung eines Sensors werden mit der schnellen Aktualisierungsrate abgefragt, welche standardmäßig auf 10 Sekunden eingestellt ist. Um diese Rate zu ändern, muss der Wert für das Feld „_waitTimeFast“ verändert werden. IXies für das kontinuierliche Senden von Daten und für abonnierte Sensoren werden mit der langsamen Aktualisierungsrate von 150 Sekunden abgefragt. Um diese Rate zu ändern, muss der Wert für das Feld „_waitTimeFast“ verändert werden. Für das Senden der Daten werden dem Besitzer ab der ersten Nachricht des Sensors, alle 24h ein fixer Betrag, je nach Kommentar, überwiesen, solange der Sensor noch aktiv sendet. Solange ein Abonnement läuft, wird dem Besitzer des Sensors alle 24h ein fixer Betrag, je nach Kommentar, hinzugefügt und dem Verbraucher ein um 1 IXie höherer

Betrag abgezogen. Das Erweitern um zusätzliche Verbuchungen ist an diesem Punkt einfach möglich.

Unter dem Ordner „Helper“ befindet sich die Klasse „BackgroundTriggerWorker“. Diese ist für die Benachrichtigungen zuständig, arbeitet in einem eigenen *Task* und sendet Benachrichtigungen für gewisse Ereignisse. Der „BackgroundTriggerWorker“ wird standardmäßig alle 90 Sekunden aktiv. Das Intervall kann über das Feld „_waitTime“ umgestellt werden. Hierbei gibt es Benachrichtigungen, welche aufgrund eines Abos passieren, wie eine Notifizierung bei Überschreitung, Unterschreitung, Ausfall des Sensors oder Änderung des *moving average*. Des Weiteren gibt es Benachrichtigungen, welche aufgrund einer Messwertdefinition passieren. Darunter fällt zum Beispiel die Notifizierung bei einem neuen Kommentar. Je nach Kategorie kann das System hierbei einfach um neue Notifizierungen erweitert werden.

IXchangeDatabase

Für die Datenhaltung in der Datenbank wird *code-first* mithilfe von *EFCore* verwendet. Die Daten können in einer SQL- oder Postgres-Datenbank gespeichert werden. Die Umstellung hierzu wird in dem Kapitel „Umstellung zwischen SQL- und Postgres-Datenbank“ erläutert. Die Struktur des Projektes ist in Abbildung 6 Database Struktur dargestellt. In dem Ordner „Converter“ befinden sich Klassen für die Konvertierung zwischen Modell-Klasse von „IXchange.Exchange“ und den Datenbank-Tabellen, welche sich in dem Unterordner „Tables“ befinden. In dem Ordner „DbHelpers“ sind Helfer-Klassen für öfter gebrauchte Kombinationen und Abfragen, wie eine Liste aller *MeasurementDefinitions* eines Benutzers. In dem Ordner „Migrations“ befinden sich die Migrationen, welche auf der SQL-Datenbank durchgeführt wurden. Der Ordner „Scripts“ ist für SQL-Skripte vorgesehen. In dem Ordner „Tables“ befinden sich die Datenbank-Tabellen Klassen. Die Klasse „Db“ ist der Datenbankkontext des Systems und beinhaltet *properties* für alle verwendeten Tabellen, sowie Methoden für das Befüllen mit Testdaten und das Löschen der Datenbank.

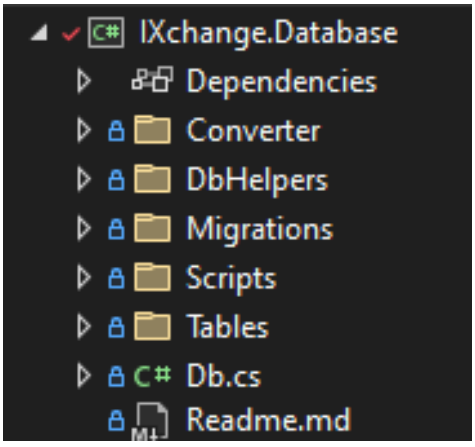

Abbildung 6 Database Struktur

IXchange.Postgres.Migrations

Das Projekt „IXchange.Postgres.Migrations“ ist für die Speicherung der Migrationen der Postgres Datenbank zuständig. Dieses wird benötigt, um die Migrationen der Postgres Datenbank und SQL-Datenbank getrennt voneinander verwalten zu können, ohne die Migrationen einer Datenbank zu überschreiben.

3 Konfigurationen und Zusatzinformationen

In diesem Kapitel wird ein Einblick in die verschiedenen Einstellungen sowie zusätzliche Informationen, die für die Konfiguration und Anpassung von IXchange von Bedeutung sind, gegeben.

3.1 Umstellung zwischen SQL- und Postgres-Datenbank

Als Datenbank kann je nach Konfiguration eine SQL-Datenbank oder eine Postgres Datenbank verwendet werden. Als *Default* ist die Konfiguration mittels SQL-Datenbank eingestellt. Für die Umstellung auf eine Postgres Datenbank sind folgende Schritte durchzuführen.

- In Db.cs: boolean "UsePostgres" auf *true* stellen

```
namespace IXchangeDatabase
{
    /// <summary>
    ///     <para>Projektweite Datenbank - Entity Framework Core</para>
    ///     Klasse Db. (C) 2021 FOTEC Forschungs- und Technologietransfer GmbH
    /// </summary>
    99+ references | 0 changes | 0 authors, 0 changes
    public partial class Db : DbContext
    {
        /// <summary>
        /// Ob Postgres-DB verwendet wird (sonst SQL-Server-DB)
        /// </summary>
        public static bool UsePostgres = false;
```

- In WebSettings.cs: Datenbank-Properties (aus *IAppSettingsDataBase*) richtig stellen

```
/// <summary>
/// Datenbank
/// </summary>
4 references | Matthias Mandl, 139 days ago | 1 author, 1 change
public string ConnectionStringDb => "";

/// <summary>
/// Datenbank-Server
/// </summary>
4 references | Matthias Mandl, 1 hour ago | 3 authors, 5 changes
public string ConnectionStringDbServer => "";

/// <summary>
/// Datenbank User
/// </summary>
3 references | 0 changes | 0 authors, 0 changes
public string ConnectionStringUser => "";

/// <summary>
/// Datenbank User Passwort
/// </summary>
3 references | Matthias Mandl, 133 days ago | 1 author, 2 changes
public string ConnectionStringUserPwd => "";
```

Für die Durchführung von Migrationen für die SQL Datenbank müssen keine Änderungen vorgenommen werden. Für Migrationen der Postgres-Datenbank muss das *Default*-Projekt auf Database.Migrations.Postgres gestellt werden.

Eine kurze Anleitung zur Umstellung auf Postgres-Datenbank befindet sich ebenfalls in dem Projekt IXchangeDatabase in der Readme.md Datei.

3.2 BISS Data Agent

Einige Bibliotheken werden von dem Opensource Projekt BDA (BISS Data Agent) übernommen. Diese werden in Form von NuGet-Packages eingebunden und sind hauptsächlich für die Logik der IoT-Clients verantwortlich. Die Veröffentlichung dieser Bibliotheken befindet sich (Stand 12/2023) in Bearbeitung.

3.3 Umstellung der IP-Adresse für Gateways

Das Gateway kommuniziert zu der IP-Adresse, welche in Exchange/AppSettings.cs angegeben ist. Sowohl beim *Debuggen* als auch vor dem Veröffentlichen von Server und Gateway-Applikation sollte diese IP-Adresse umgestellt werden und ein *rebuild* des Projektes durchgeführt werden um die korrekte Kommunikation von Server und Gateway zu gewährleisten.

4 Open Data Schnittstelle

Dieses Kapitel bietet einen Einblick in die Endpunkte und den *Authorization*-Prozess der Open Data Schnittstelle von IXchange.

4.1 Authorization

Der *Authorization*-Prozess wird für die OData Schnittstelle über einen Token, welcher in der Tabelle „TableResearchInstitutesAccess“ gespeichert wird, abgehandelt. Der Betreiber des Systems muss manuell einen Token anlegen und Interessenten die „AccessToken“-Zeichenkette davon übergeben. Der Token ist nur für die Zeit zwischen „ValidFrom“ bis „ValidUntil“ valide. Der *Authorization*-Prozess wird bei den ausgewählten OData-Endpunkten durch das „IXChangeODataAuthorize“-Attribut automatisch durchgeführt. Dieses befindet sich im Ordner „IXchange.Service.Com.Base/Helpers“. Für die Abfrage muss der Token als *Bearer*-Token mitgegeben werden.

4.2 Endpunkte

Die Endpunkte, welche in Abbildung 7 OData Endpunkte dargestellt sind, können einfach um zusätzliche Endpunkte oder *Controller* erweitert werden und befinden sich im Ordner „IXchange.Service.Com.Rest/Controllers/ODataControllers“.

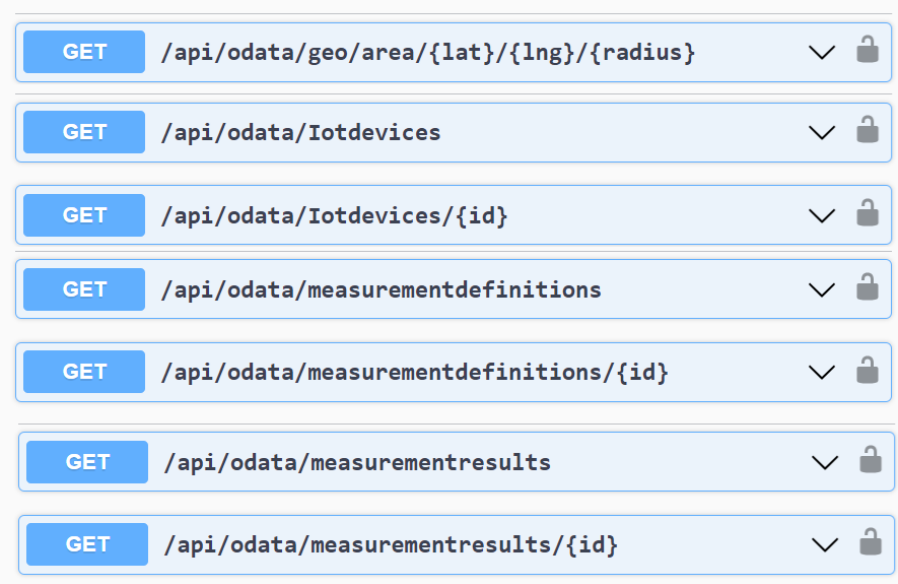

Abbildung 7 OData Endpunkte

Wie die OData Endpunkte abgefragt werden können, inklusiven Beispielen dazu, ist einerseits in der ODataReadme.md in dem Ordner „IXchange.Service.Com.Rest/Controllers/ODataControllers“, als auch in der Anwenderdokumentation näher erläutert.

5 Abbildungsverzeichnis