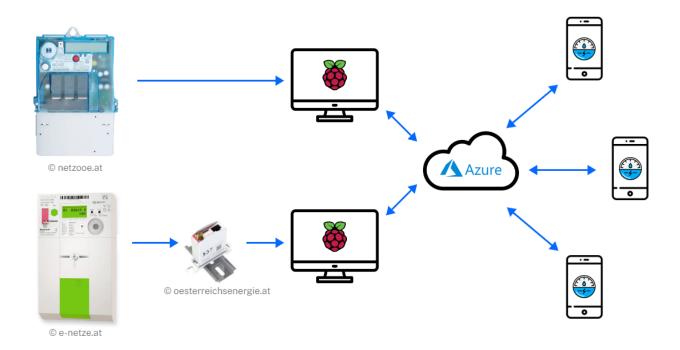
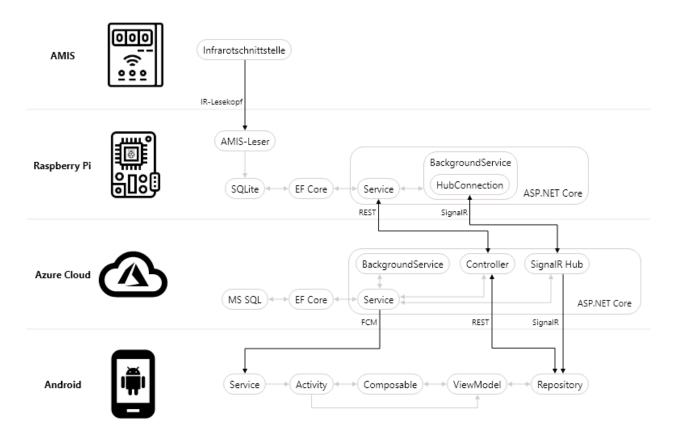
#### Architektur

## Allgemein

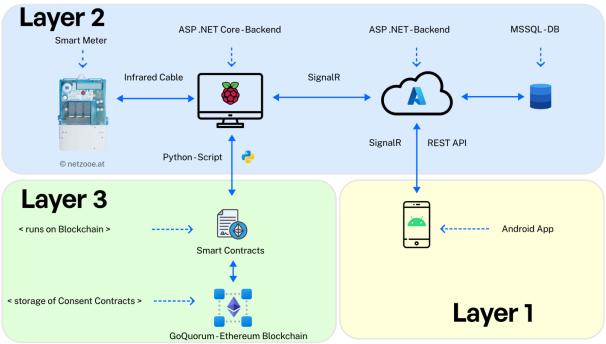
- Smart Meter
  - o digitales Messgerät für den Energieverbrauch/-erzeugung
  - Schnittstelle zwischen Haushalt und Netzbetreiber
    - weiters auch zwischen Haushalt und unserer Plattform
  - o Kundenschnittstelle AMIS Zähler der Energie AG
    - Infrarotschnittstelle
    - liefert jede Sekunde einen Messwert
    - Verschlüsselung der Daten mittels AES-Key
    - künftig einheitliche Schnittstelle in Österreich
- Raspberry
  - liest die Energiedaten mittels der Kundenschnittstelle vom Smart Meter aus
    - C++ Service (programmiert von Energie AG und abgeändert durch uns)
      - persistiert historische Daten alle 15 min
      - speichert Sekundendaten für die letzten 90 Sekunden
  - .NET Core Backend
    - bearbeitet die Energiedaten entsprechend für die Plattform
    - Datenaustausch von/zur Cloud mittels SignalR
      - vor allem Echtzeitdaten
    - Datenaustausch zur Cloud teilweise mit REST (ohne Echtzeitrelevanz)
  - o SQLite3 Datenbank
  - o GoQuorum Blockchain Node
    - Jeder Raspberry befindet sich in einem privaten Ethereum Netzwerk
    - Dient zur dezentralen Speicherung von Einwilligungserklärungen
- Cloud (Azure)
  - .NET Core Backend
    - zentrale Logik und Zugangspunkt
    - Handler für alle Energiegemeinschaften und deren Mitglieder inklusive Smart
       Meter
  - MS-SQL Datenbank
- Endgeräte (Android)
  - Schnittstelle zum Mitglied
  - Echtzeitdaten werden mittels SignalR empfangen
  - andere Daten werden über REST-Requests abgefragt
  - o erhalten Push-Nachrichten mittels Firebase Cloud Messaging (FCM)



## Optimierung



# **Consent Management**

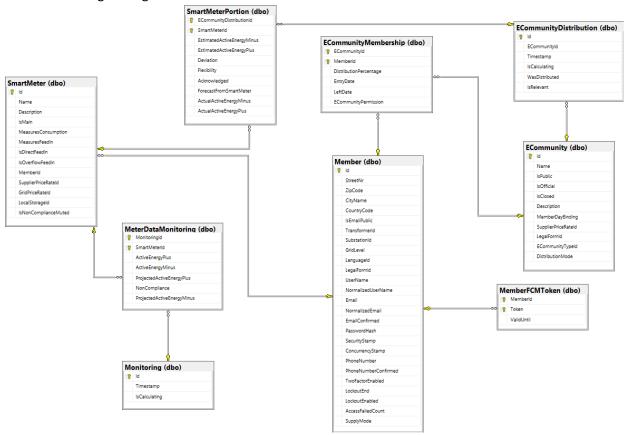


# Datenmodelle

Beschreibung der Attribute kann man sich im Source Code ansehen.

### Cloud (Azure)

Anm.: Nicht verwendete Tabellen sind in der folgenden Abbildung nicht dargestellt, können aber im Quellcode nachgeschlagen werden.



## Lokal (Raspberry Pi)



GridPriceRate

Id TEXT

BaseRate REAL

WorkingPricePlus REAL

TaxRate REAL

GridLevel INTEGER

PVSystem

Id TEXT

PeakWP REAL

 Id
 TEXT

 IsPowerFeedAllowed
 INTEGER

 Id
 TEXT

 BaseRate
 REAL

 WorkingPricePlus
 REAL

 TaxRate
 REAL

 ApplyToECommunity
 INTEGER

 Id
 TEXT

 StreetNr
 TEXT

 ZipCode
 TEXT

 CityName
 TEXT

 CountryCode
 TEXT

 GridLevel
 INTEGER

 SupplyMode
 INTEGER

SmartMeter TEXT AESKev TEXT APIKey TEXT TEXT Name TEXT Description INTEGER MeasuresConsumption INTEGER MeasuresFeedIn INTEGER INTEGER IsOverflowFeedIn INTEGER LocalStorageId

MeterDataHistory INTEGER INTEGER Active Energy MinusActiveEnergyPlus INTEGER ActivePowerMinus INTEGER ActivePowerPlus INTEGER REAL Cloudiness EventCaseId TEXT PrepaymentCounter INTEGER RainVolume REAL ReactiveEnergyMinus INTEGER ReactiveEnergyPlus INTEGER ReactivePowerMinus INTEGER ReactivePowerPlus INTEGER SnowVolume REAL Temperature REAL Visability REAL WorkingPriceMinus REAL WorkingPricePlus REAL

MeterDataProfile INTEGER WorkingPricePlus REAL WorkingPriceMinus RFAI Timestamp TEXT ActiveEnergyPlus INTEGER  ${\sf ActiveEnergyMinus}$ INTEGER ReactiveEnergyPlus INTEGER INTEGER ReactiveEnergyMinus ActivePowerPlus INTEGER ActivePowerMinus INTEGER ReactivePowerPlus INTEGER ReactivePowerMinus INTEGER PrepaymentCounter INTEGER

 SupplierPriceRate

 Id
 TEXT

 BaseRate
 REAL

 WorkingPricePlus
 REAL

 WorkingPriceMinus
 REAL

 PricePerPeak
 REAL

 TaxRate
 REAL

 Id
 TEXT

 Name
 TEXT

 Priority
 INTEGER

 MeterDataRealTime

INTEGER TEXT INTEGER ActiveEnergyPlus INTEGER ActiveEnergyMinus INTEGER INTEGER ReactiveEnergyMinus ActivePowerPlus INTEGER ActivePowerMinus INTEGER ReactivePowerPlus INTEGER ReactivePowerMinus INTEGER PrepaymentCounter INTEGER

 Id
 INTEGER

 AccessToken
 TEXT

 RefreshToken
 TEXT



### App (Android)

```
@Entity(tableName = "tile")
@data class Tile(
    @PrimaryKey
    var tileId: String,
    var region: String,
    var action: String,
    var value: String,
    var isVisible: Boolean,
    var isLargeTile: Boolean = false,
    var colorId: Int,
```

Entity(tableName = "contract")

```
@Entity(tableName = "smart_meter")
∂data class SmartMeter(
    @PrimaryKey var <u>ID</u>: String,
    var <u>name</u>: String,
    var <u>description</u>: String
```

```
@Entity (tableName = "member")
@data class Member(
    @PrimaryKey var memberId: String,
    var username: String,
    var email: String,
    var password: String,
    var accessToken: String,
    var refreshToken: String,
    var languageName: String)

@Entity(tableName = "blockchain_balance")
data class BlockchainBalance(

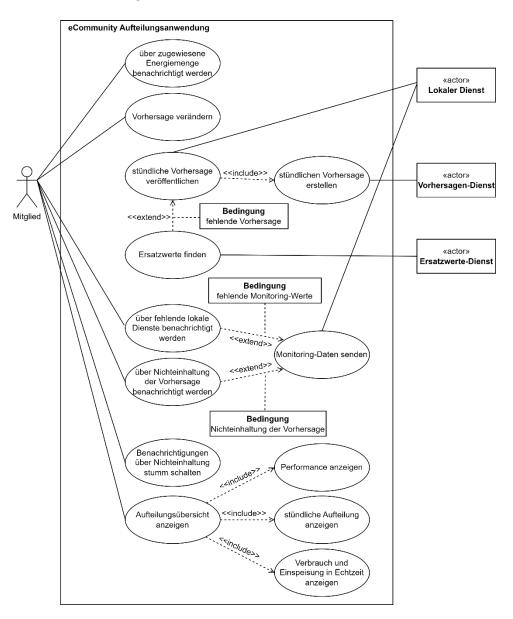
@PrimaryKey(autoGenerate = true)
val id: Int? = null,
val received: String? = null,
val sent: String? = null,
var balance: String? = null
)
```

```
data class ConsentContract (
    var contractId: kotlin.String,
    @SerializedName("State")
                                                   @Entity(tableName = "meter_data_hist_contract")
    var state: kotlin.Int? = null,
                                                   data class MeterDataHistContract(
    @SerializedName("AddressContract")
    val addressContract: kotlin.String? = null,
                                                       @PrimaryKey(autoGenerate = true)
    @SerializedName("AddressProposer")
                                                       @SerializedName("EnergyId")
    val addressProposer: kotlin.String? = null,
                                                       val id: Int? = null,
    @SerializedName("AddressConsenter")
                                                       var contractId: String? = null,
    val addressConsenter: kotlin.String? = null,
                                                       @SerializedName("id")
    @SerializedName("StartEnergyData")
    val startEnergyData: kotlin.String? = null,
                                                       val dataId: Int? = null,
                                                       @SerializedName("timestamp")
    val endEnergyData: kotlin.String? = null,
                                                       val timeStamp: String? = null,
    @SerializedName("ValidityOfContract")
                                                       val activeEnergyPlus: Int? = null,
    val validityOfContract: kotlin.String? = null,
                                                       val activeEnergyMinus: Int? = null,
    @SerializedName("PricePerHour")
                                                       val reactiveEnergyPlus: Int? = null,
    val pricePerHour: kotlin.String? = null,
                                                       val reactiveEnergyMinus: Int? = null,
    @SerializedName("TotalPrice")
                                                       val activePowerPlus: Int? = null,
    val totalPrice: kotlin.String? = null,
                                                       val activePowerMinus: Int? = null,
    @SerializedName("DataUsage")
                                                       val reactivePowerPlus: Int? = null,
    val dataUsage: kotlin.Int? = null,
                                                       val reactivePowerMinus: Int? = null,
    @SerializedName("TimeResolution")
    val timeResolution: kotlin.Int? = null
                                                       val prepaymentCounter: Int? = null
```

## **Use Cases**

## **Optimierung**

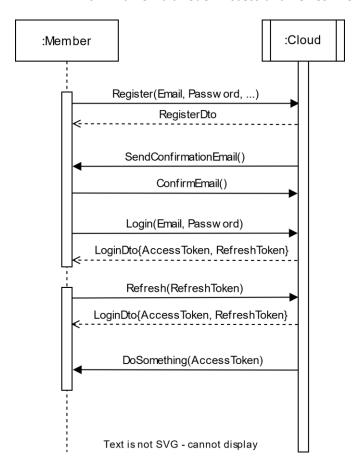
→ Beschreibung siehe Bachelorarbeit Fischer



## Abläufe

### Authentifizierung

- Login
  - o Member muss E-Mail bestätigt haben
  - o E-Mail und Passwort müssen natürlich übereinstimmen
  - o bei 3 fehlgeschlagenen Versuchen
    - Member für 5 Minuten gesperrt
- Access Token
  - o gültig für 10 Minuten
  - o Zugriff auf Ressourcen (REST und SignalR)
- Refresh Token
  - o gültig für 90 Tage
  - o man erhält neuen Access und Refresh Token

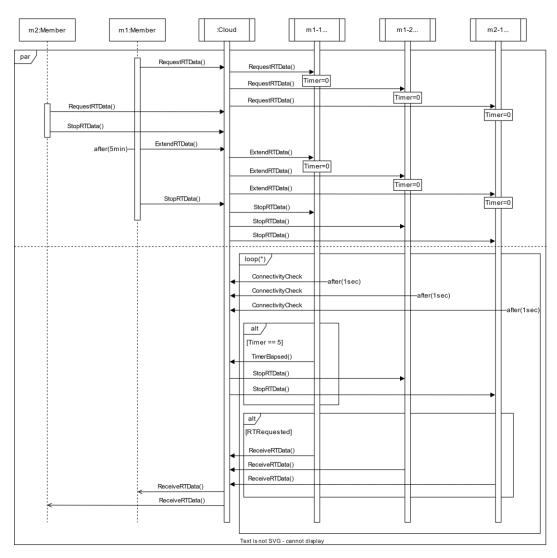


# **Pairing** :RaspberryPi :Member :Cloud discoverDevices() sendlpAndPort() requestStatus() sendStatus() alt [internet connection] connectToCloud(memberld) cloudConnect() getSmartMeterId() ende fullyConfigured plugInEthernet requestNearbyWifi() sendAvailableWifis() selectAndEnterPW() connectionStatus enter device settings cloudConnect() ende fullyConfigured getSmartMeterId()

## Real Time

- Abwicklung des Echtzeit-Traffics mittels SignalR
- RequestRTData
  - o Endgerät-Session starten
  - Raspberry-Sessions starten
    - falls nicht bereits gestartet
    - bei ersten Gerät von Energiegemeinschaft
- ExtendRTData
  - o Endgeräte müssen alle 5 Minuten ihre Session verlängern

- Verhinderung von unnötigen Traffic
- StopRTData
  - o Endgerät-Session stoppen
  - Raspberry-Sessions stoppen
    - falls alle Endgerät-Sessions geschlossen
- jede Sekunde durchläuft ein Raspberry die Schleife
- ConnectivityCheck
  - überprüft ob SignalR-Verbindung noch offen ist
  - o falls geschlossen: neu verbinden
- TimerElapsed
  - o Timer ist abelaufen
  - Echtzeit-Session stoppen
  - o ExtendRTData wurde nicht aufgerufen
- ReceiveRTData
  - o aktueller Messwert an Cloud senden



## **Optimierung**

- Stündliche Vorhersage
  - o 10 Minuten vor vollen Stunde (e.g. 11:50 Uhr)
  - o Vorhersage wird vom Server angefragt und anschließend vom lokalen Dienst gesendet
- Neue Energiezuteilung
  - o 5 Minuten vor vollen Stunde (e.g. 11:55 Uhr)
  - o wird an Mitglied gesendet
    - dieser kann ggf. die Flexibilität verändern
    - nach Veränderung der Flexibilität wird die Nachricht erneut versendet
- Finale Energiezuteilung
  - o volle Stunde (e.g. 12:00 Uhr)
  - o wird an Mitglied gesendet
- Monitoring
  - o alle vollen 5 Minuten
  - Monitoring-Daten werden vom Server angefragt und anschließend vom lokalen Dienst gesendet
    - Überprüfung auf Nichteinhaltung und auf fehlende Einträge (offline)
- → detaillierte Beschreibung siehe Bachelorarbeit Fischer

