Progetto d’esame: laboratorio di Progettazione e implementazione di sistemi software in rete & Applicazioni Web

Fase di progettazione

Componenti principali dell'architettura

- **Client App**: (C# con Razor Pages) eseguita nel browser: interagisce con un microservizio che funge da punto di accesso al sistema e che quale contatta il backend tramite le API REST.
- **Backend**: (C# .NET) permette agli utenti di accedere per consultare le misure presenti nel DB o inserire / modificare / eliminare i dati relativi alle varie entità gestite, come coltivazioni, offerte e ordini. Espone un'interfaccia REST. Comunica anche con il sottosistema IoT tramite il protocollo MQTT.
- **DB**: (PostgreSQL) database contenente tutte le informazioni necessarie al funzionamento del sistema (es. misure rilevate, coltivazioni, offerte idriche, informazioni degli utenti, …). Accessibile solo tramite il backend.
- **Sottosistema IoT** (Java) - si occupa di rilevare misure tramite i sensori che invia tramite broker MQTT e tramite lo stesso canale di comunicazione può accettare comandi per gli attuatori. Può essere quindi sia publisher che subscriber.

Architettura del sistema e componenti

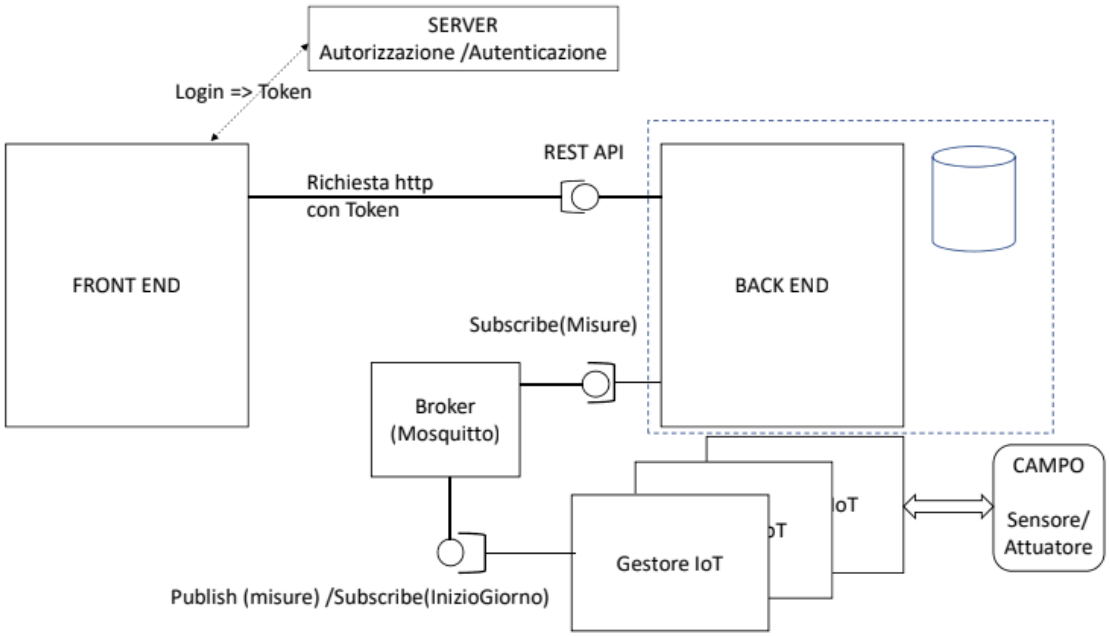
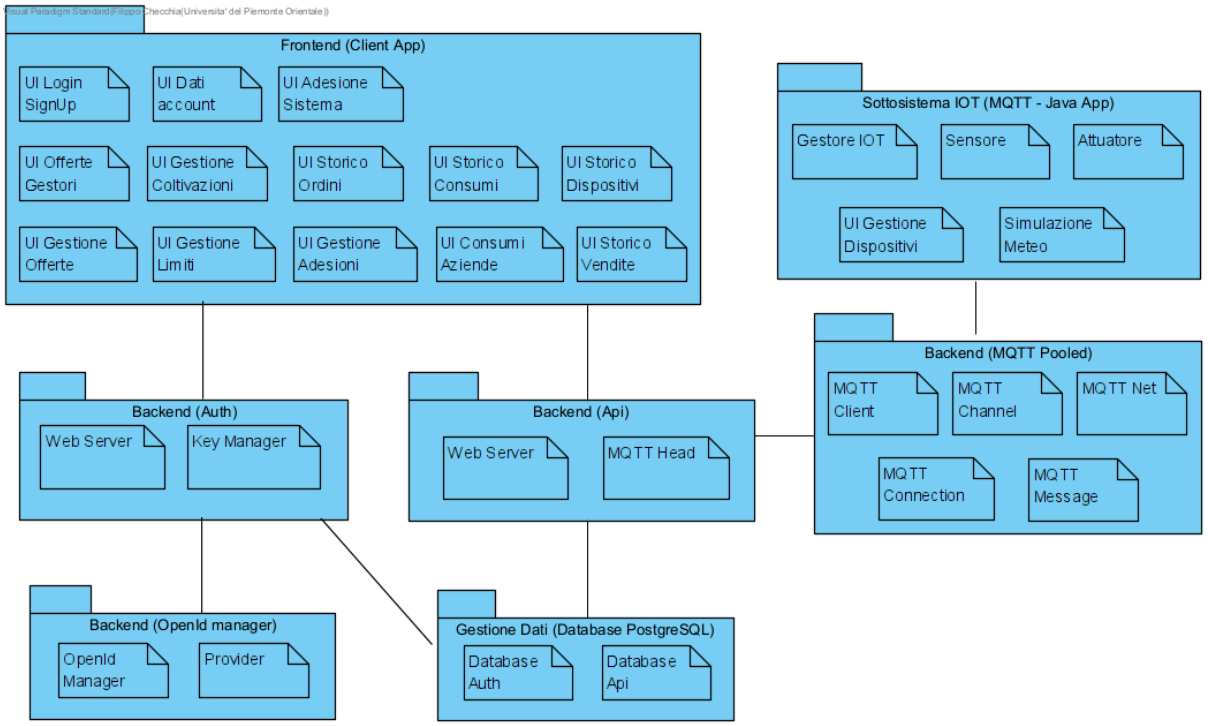


Diagramma dei package



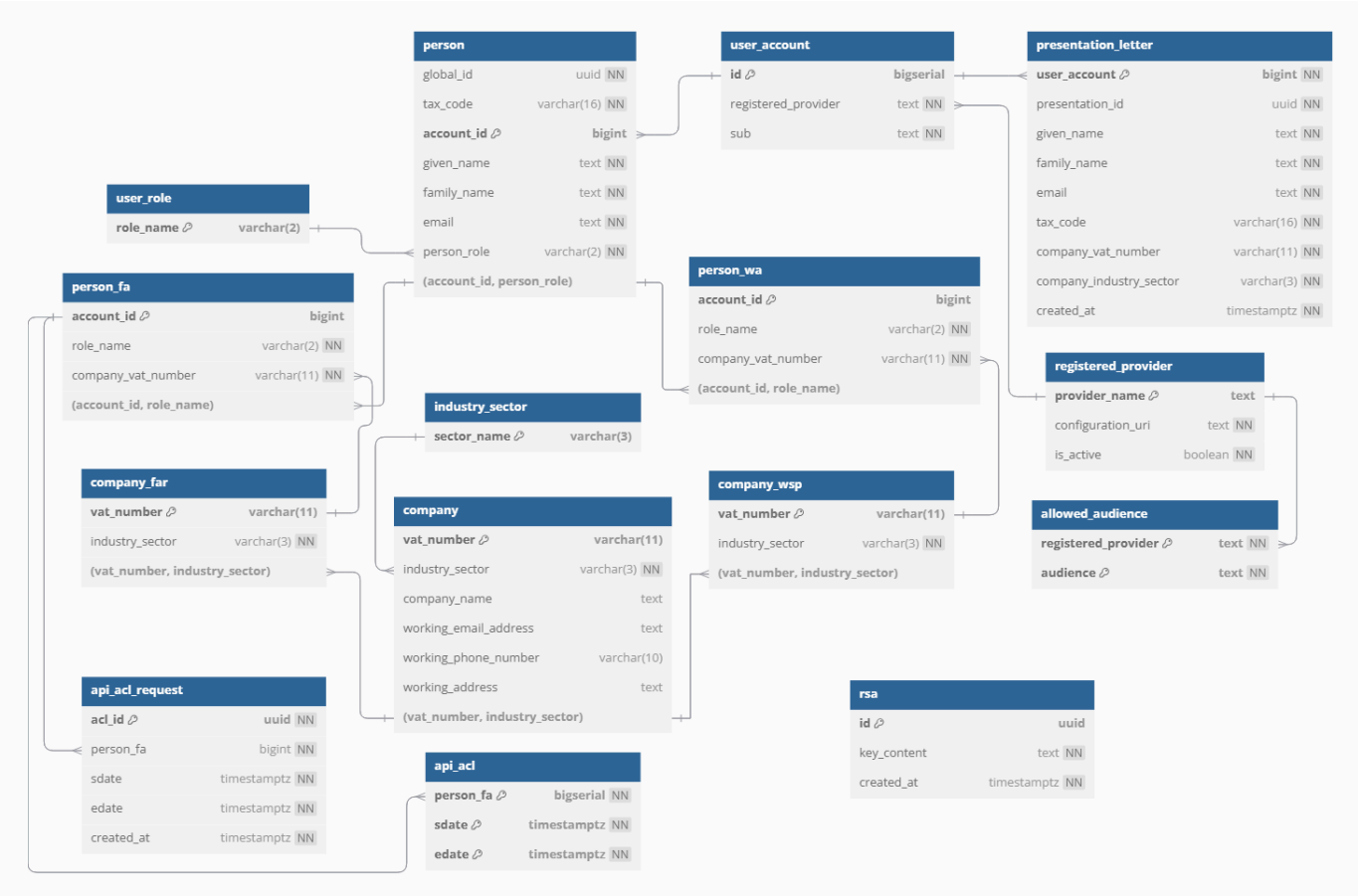
- **Frontend (Client App)**: UI web in C# tramite Razor page del sistema. Interagisce tramite API REST con i Backend Auth e API.
- **Sottosistema IOT (MQTT)**: Applicazione Java per gestire i dispositivi IOT. Comunica con il backend tramite il protocollo MQTT.
- **Backend (Auth)**: Parte del backend che si occupa dell'autenticazione e dell'autorizzazione degli utenti e delle aziende nel sistema mediante l'utilizzo di OpenId e OAuth.
- **Backend (OpenId Manager)**: Parte del backend che si occupa della gestione del protocollo OpenId e delle relative chiavi.
- **Backend (API)**: Parte del backend che si occupa della gestione (previa autenticazione/autorizzazione) degli aspetti e delle funzionalità legate al sistema idrico, come la gestione delle coltivazioni, offerte, storico delle informazioni, ecc…
- **Backend (MQTT Pooled)**: Parte del backend che tramite il protocollo MQTT comunica con il sottosistema IOT.
- **Gestione Dati (Database PostgreSQL)**: Basi di dati del sistema. Auth si occupa di immagazzinare le informazioni degli utenti, aziende e della gestione dell'accesso al sistema; Api si occupa di immagazzinare informazioni sul sistema idrico (coltivazioni, offerte idriche, consumi, ecc…).

Progettazione e implementazione dei sottosistemi

Andrea Barchietto: Backend (Auth, API, MQTT Pooled, OpenId Manager), DB
Filippo Checchia: Frontend, Sottosistema IOT, DB

Diagrammi ER

Database Auth



- **person**: contiene informazioni personali di un utente, tra cui nome, cognome, codice fiscale e ruolo.
- **person_fa**: contiene informazioni degli utenti che lavorano in un'azienda agricola.
- **person_wa**: contiene informazioni degli utenti che lavorano in un'azienda idrica.

- **company**: contiene i dettagli di un’azienda.
- **industry_sector**: contiene i settori industriali (WSP idrico, FAR agricolo).
- **company_far**: elenca le aziende del settore FAR.
- **company_wsp**: elenca le aziende del settore WSP.

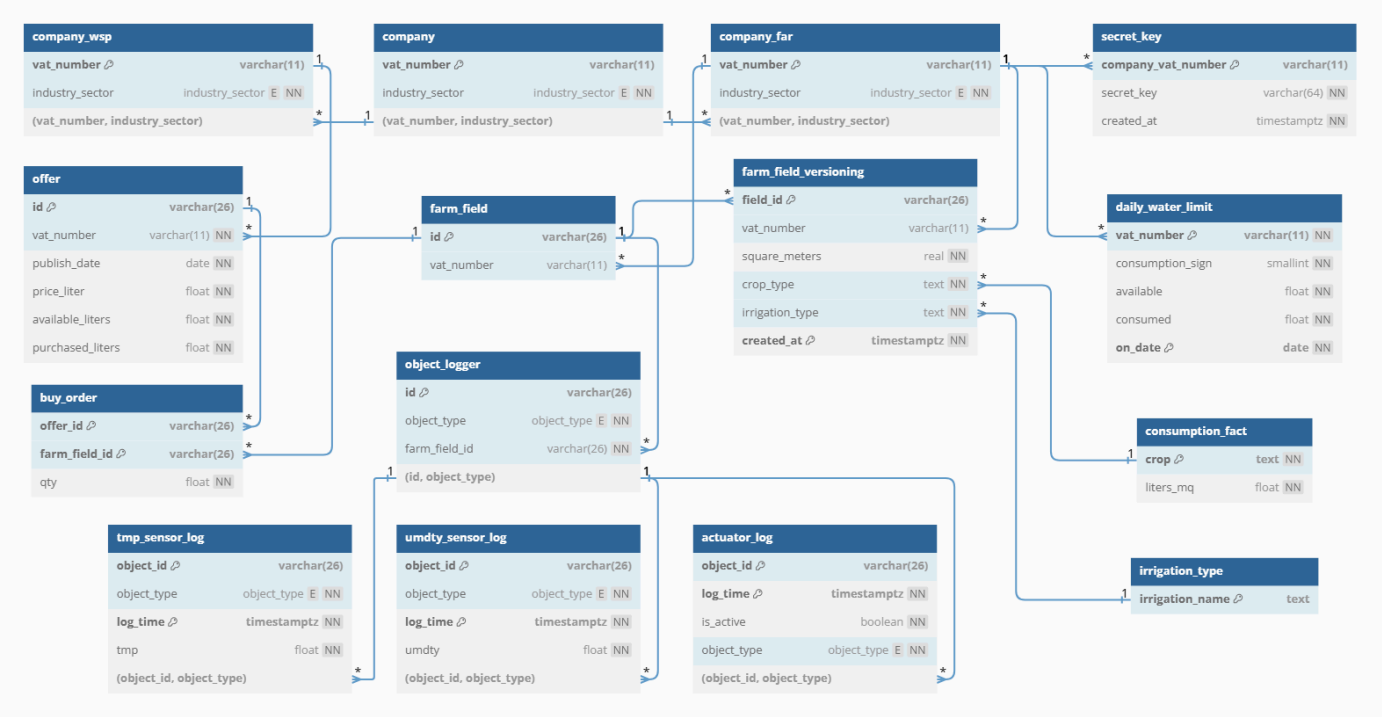
- **allowed_audience**: elenca gli identificativi delle applicazioni accettate per l’accesso al servizio. Tali identificativi vengono generati dai provider OpenId e sono presenti all’interno del campo <aud> di un “id token” generato da un provider OpenId.
- **registered_provider**: elenca i provider OpenId che possono essere usati dagli utenti per effettuare la registrazione al servizio.

- **user_account**: associa gli utenti con un provider OpenId usando come identificativo la stringa presente nel campo <sub> di un “id token”.
- **user_role**: elenca i ruoli utente (WA utente idrico, FA utente agricolo).

- **presentation_letter**: contiene le presentation letter per ogni account utente che si vuole iscrivere al sistema.
- **api_acl_request**: registra le richieste di accesso alle API per un utente agricolo.
- **api_acl**: contiene gli utenti che hanno accesso alle API in un dato periodo (da/a).

- **rsa**: memorizza chiavi RSA generate con data di creazione per l'autenticazione. Tali chiavi servono per firmare gli access token generati dal backend auth verso il backend api. Il backend api userà poi, le chiavi pubbliche per verificare la firma dei token.

Database API

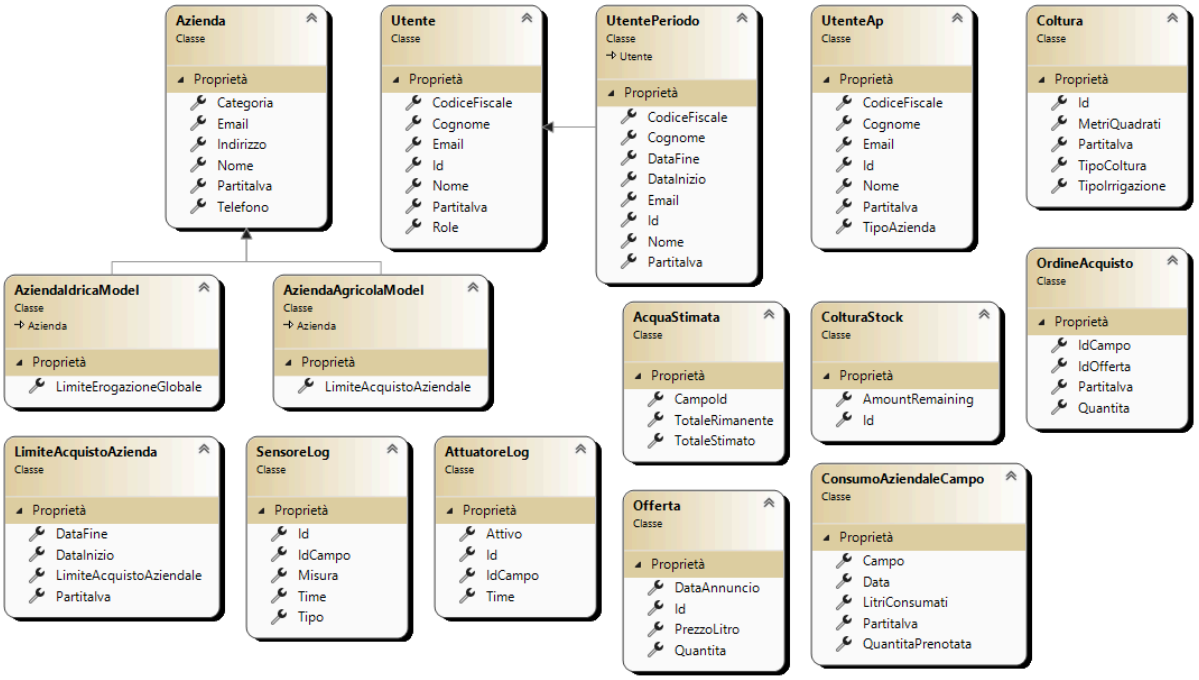


- **company**: rappresenta un'azienda, identificata dal numero di partita IVA e dal settore industriale.
- **industry_sector**: elenco di settori industriali che include WSP (azienda idrica) e FAR (azienda agricola).
- **company_far**: identifica le aziende agricole (settore FAR).
- **company_wsp**: identifica le aziende idriche (settore WSP).
- **daily_water_limit**: memorizza il limite giornaliero di acqua acquistabile per ogni azienda agricola, per una data specifica.
- **secret_key**: memorizza una chiave segreta per le aziende FAR, associata al numero di partita IVA, per firmare e autenticare i messaggi scambiati tra backend e gestore IoT via broker MQTT.
- **offer**: rappresenta un'offerta di vendita per l'acqua, con dettagli come data di pubblicazione, prezzo al litro e quantità disponibili.
- **buy_order**: registra gli ordini di acquisto associati a un’offerta specifica e a un campo agricolo.

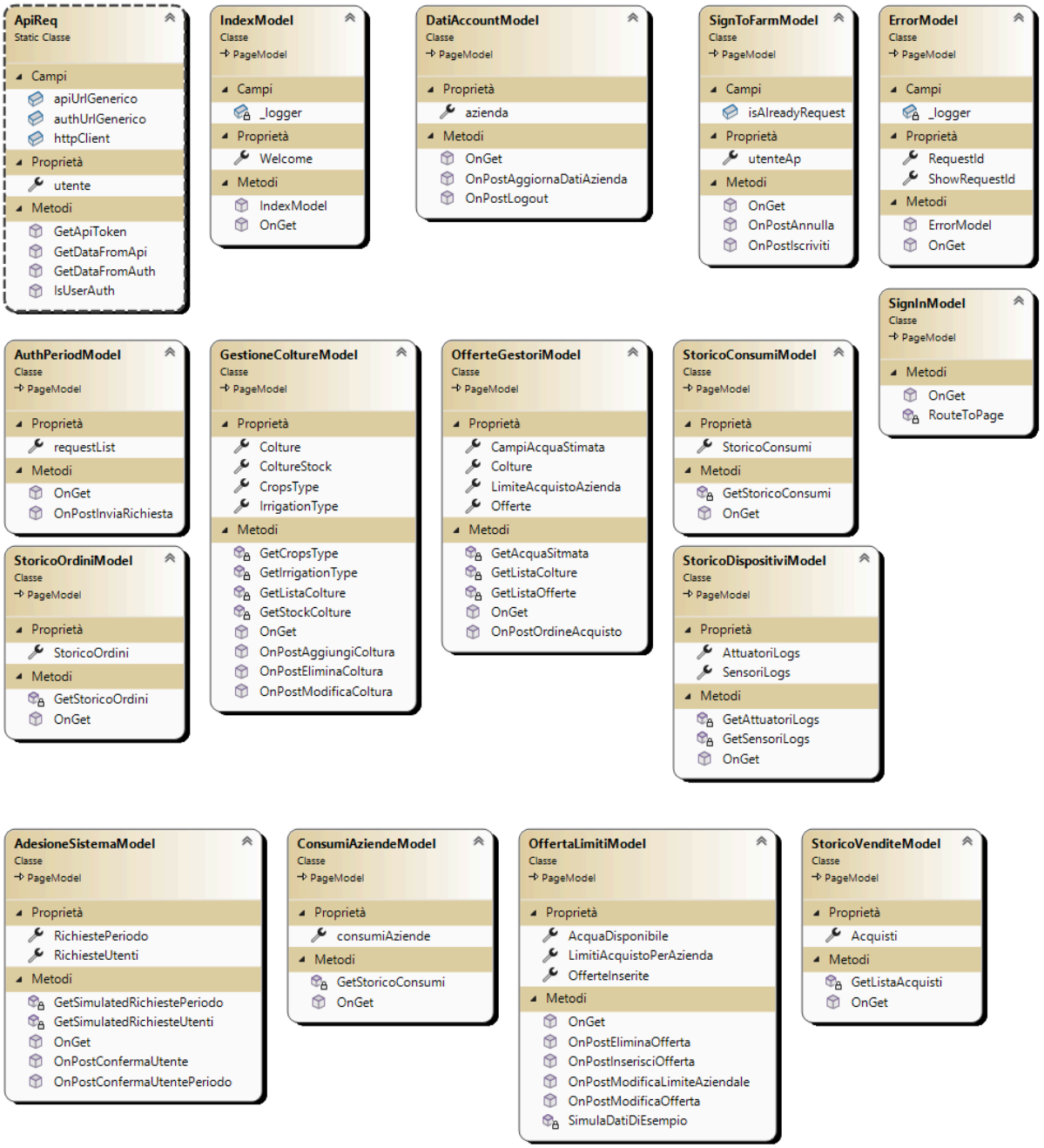
- **farm_field**: rappresenta un campo agricolo appartenente a un'azienda, identificato tramite ID e partita IVA dell'azienda FAR.
- **farm_field_versioning**: memorizza versioni storiche di un campo agricolo, con dettagli come dimensione, tipo di coltura e tipo di irrigazione.
- **irrigation_type**: elenca i tipi di irrigazione disponibili per i campi agricoli.
- **consumption_fact**: memorizza il consumo d'acqua per metro quadro, specifico per ciascun tipo di coltura.
- **object_logger**: tiene traccia delle azioni degli oggetti nel sistema agricolo, inclusi tipo e ID del campo associato.
- **object_type**: è un elenco di tipi di oggetti che include UMDTY, TMP e ACTUATOR.
- **umdty_sensor_log**: registra i log dei sensori UMDTY, includendo ID dell'oggetto, tipo di oggetto, tempo di registrazione e valori misurati.
- **tmp_sensor_log**: registra i log dei sensori TMP, con ID dell'oggetto, tipo di oggetto, tempo di registrazione e valori di temperatura.
- **actuator_log**: tiene traccia dell'attivazione degli attuatori, registrando ID dell'oggetto, tipo di oggetto, tempo e stato attivo/inattivo.

Diagrammi delle classi

Frontend



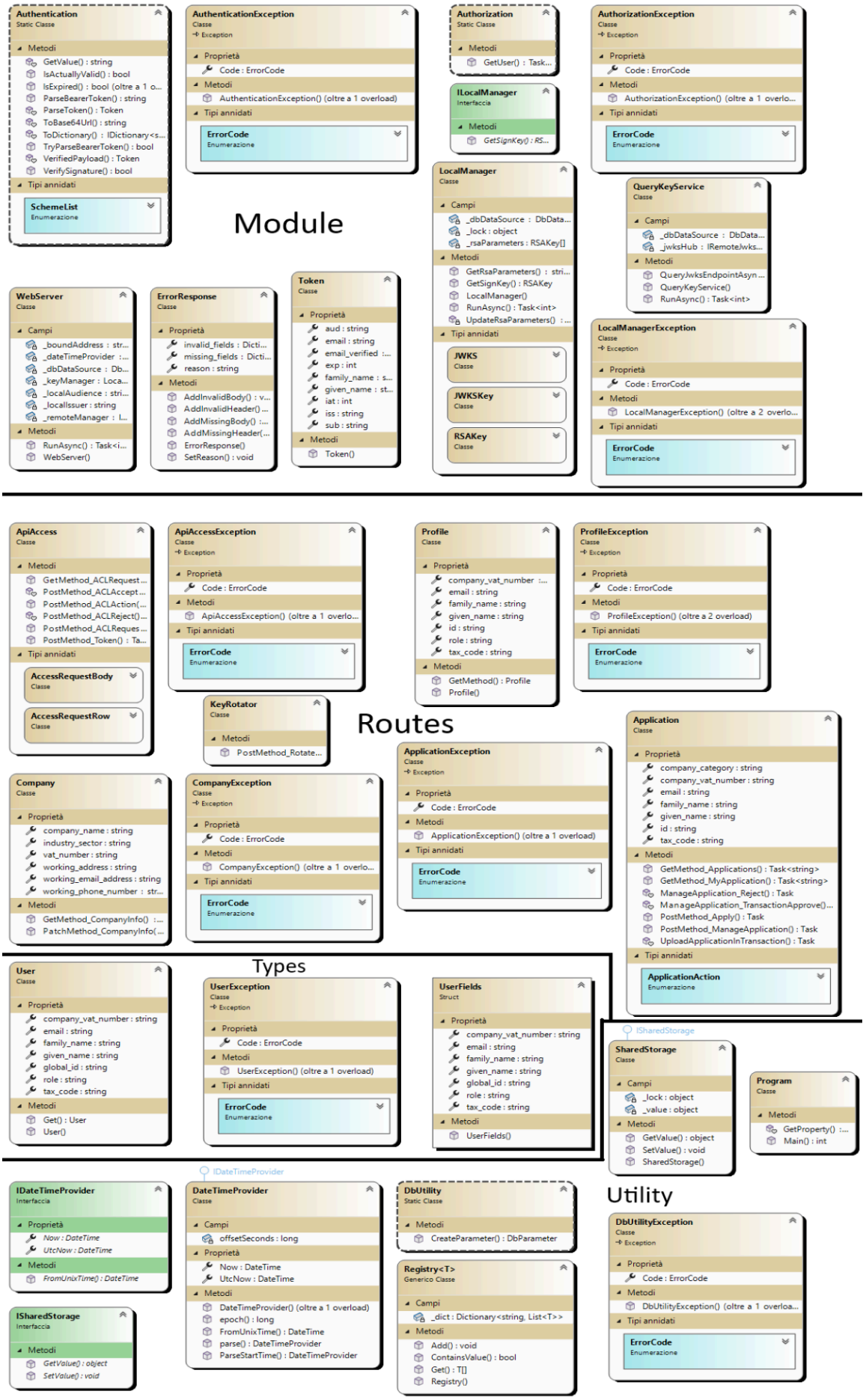
Frontend Models: Classi rappresentanti le entità del sistema con i relativi attributi, utilizzate solamente dal frontend dopo aver richiesto le informazioni al backend per ricostruirne la struttura.



Frontend Pages: Classi rappresentanti le pagine web del sistema idrico, ogni pagina ha un elemento OnGet per ottenere le informazioni necessarie dal backend, e dei metodi per gestire metodi come POST, PATCH, DELETE per gestire i dati del sistema nella relativa pagina del model. I metodi OnGet hanno come riferimento la classe ApiReq che permette di effettuare le richieste al backend Auth o API tramite i token.

Backend

Backend Auth



Il backend di autenticazione svolge due ruoli: **autentica** gli utenti e ne gestisce le **autorizzazioni**. L'autenticazione avviene tramite provider OpenId esterni. Questo backend effettua un compito molto preciso: oltre all'autenticazione, permette la mappatura degli utenti alle rispettive aziende agricole o idriche per cui lavorano. Con questi due dati, vengono poi generati gli access token necessari per permettere agli utenti di effettuare chiamate al backend api. Ogni nuovo utente, a prescindere che faccia richiesta per far parte di un'azienda idrica o agricola, deve essere approvato da un utente con ruolo "utente azienda idrica" già registrato nel sistema.

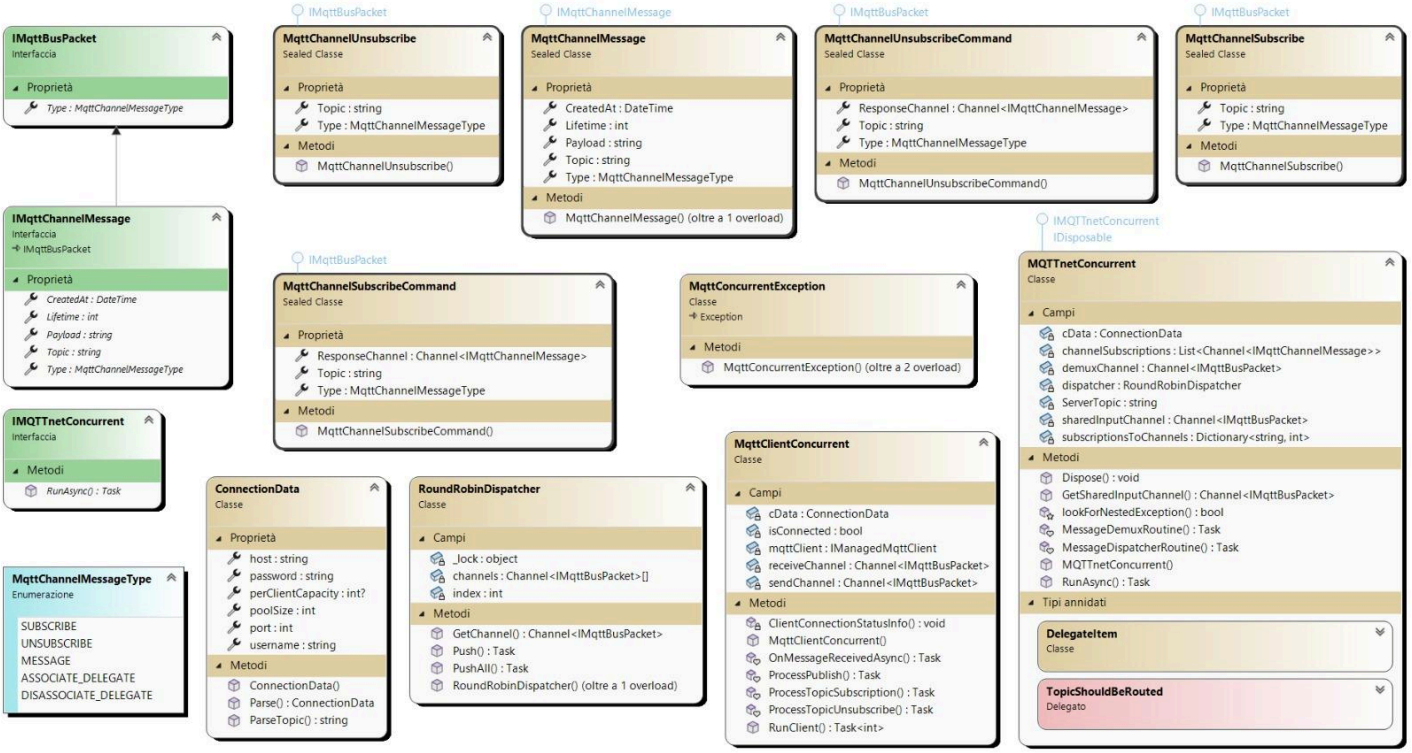
Gli utenti che lavorano per un'azienda idrica possono accedere al backend api in qualsiasi momento, mentre quelli che lavorano per un'azienda agricola necessitano di essere approvati per il periodo di tempo per cui fanno richiesta.

L'access token che genera questo backend contiene 3 informazioni essenziali: l'id univoco che identifica l'utente nel sistema, la partita iva che identifica l'azienda, e il ruolo dell'utente. Il token ha una validità di 10 minuti ed è un token JWT firmato con RS256. Questo backend, come i provider oauth, ha un endpoint che contiene la lista delle chiavi RSA utili per verificare i token da esso generati. Sia gli utenti che lavorano per un'azienda agricola, sia quelli che lavorano per un'azienda idrica necessitano di un access token per interagire con il backend api, la differenza sta nel fatto che agli utenti di un'azienda idrica, la generazione del token non prevede preventive richieste di accesso da far approvare.

Backend API

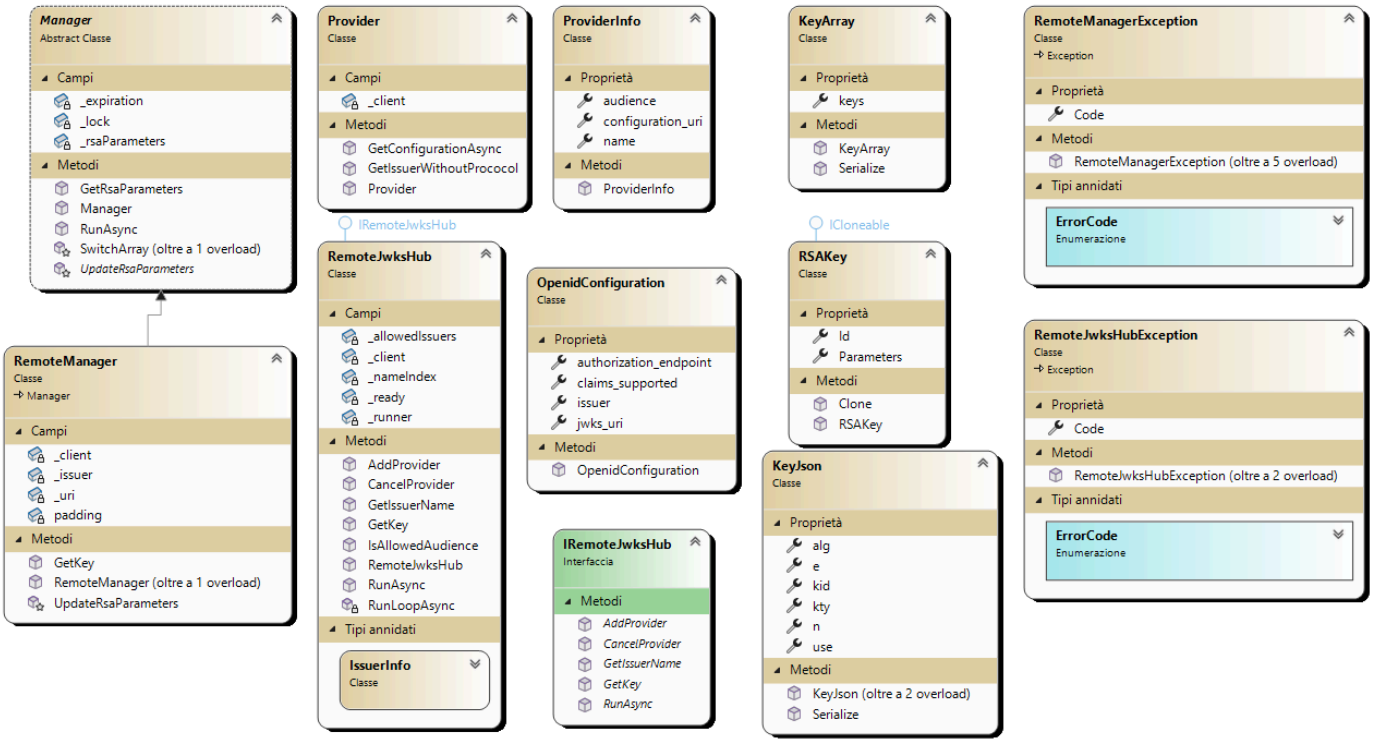
Il backend API ha due compiti fondamentali: dare l'**accesso** alle informazioni per la gestione delle aziende agricole e idriche, e gestire l'interazione con l'applicazione Java dei **sottosistema IoT** tramite il protocollo **MQTT**.
Le azioni effettuabili in questo backend (tramite chiamate REST) sono definite nel capitolo Definizione API rest dopo l'ottenimento dell'access token, spiegato nel paragrafo precedente. Ogni classe creata fa riferimento ad un aspetto del sistema idrico (coltivazioni, offerte idriche, storico consumi, gli oggetti del sottosistema IoT come attuatori e sensori, ecc..).
L'utente dell'azienda idrica ad esempio potrà tramite le pagine web del frontend, dopo essersi autenticato e ottenuto l'autorizzazione necessaria dal backend auth, effettuare le operazioni di creazione, modifica ed eliminazione di un'offerta idrica contattando l'endpoint "/water/offer" per chiedere al backend API di aggiornare le informazioni nel database API; allo stesso modo l'utente dell'azienda agricola può gestire le informazioni relative ai campi dell'azienda stessa. In generale tutte le classi e le relative operazioni fanno riferimento alle funzionalità del sistema definite nella specifica per i diversi tipi di utente.

MQTT Pooled Library



MQTT Pooled è una libreria il cui scopo è quello di creare un certo numero di connessioni MQTT verso uno stesso server (es mosquitto) con lo scopo di aumentare il throughput nell'invio e nella ricezione dei messaggi. Questa libreria per funzionare nel modo corretto necessita di un server MQTT che supporti la versione 5 del protocollo. La versione 5 permette di creare gruppi all'interno dei singoli canali in cui i messaggi vengono inviati con un rapporto 1:1 per ogni client, a differenza delle altre versioni in cui ogni messaggio viene inviato a tutti i client. In questo modo, tutti i client presenti nel pool possono iscriversi agli stessi canali MQTT. Questa soluzione, idealmente, permette di poter eseguire N istanze del backend senza rischio di ricevere messaggi duplicati.

OpenId Manager

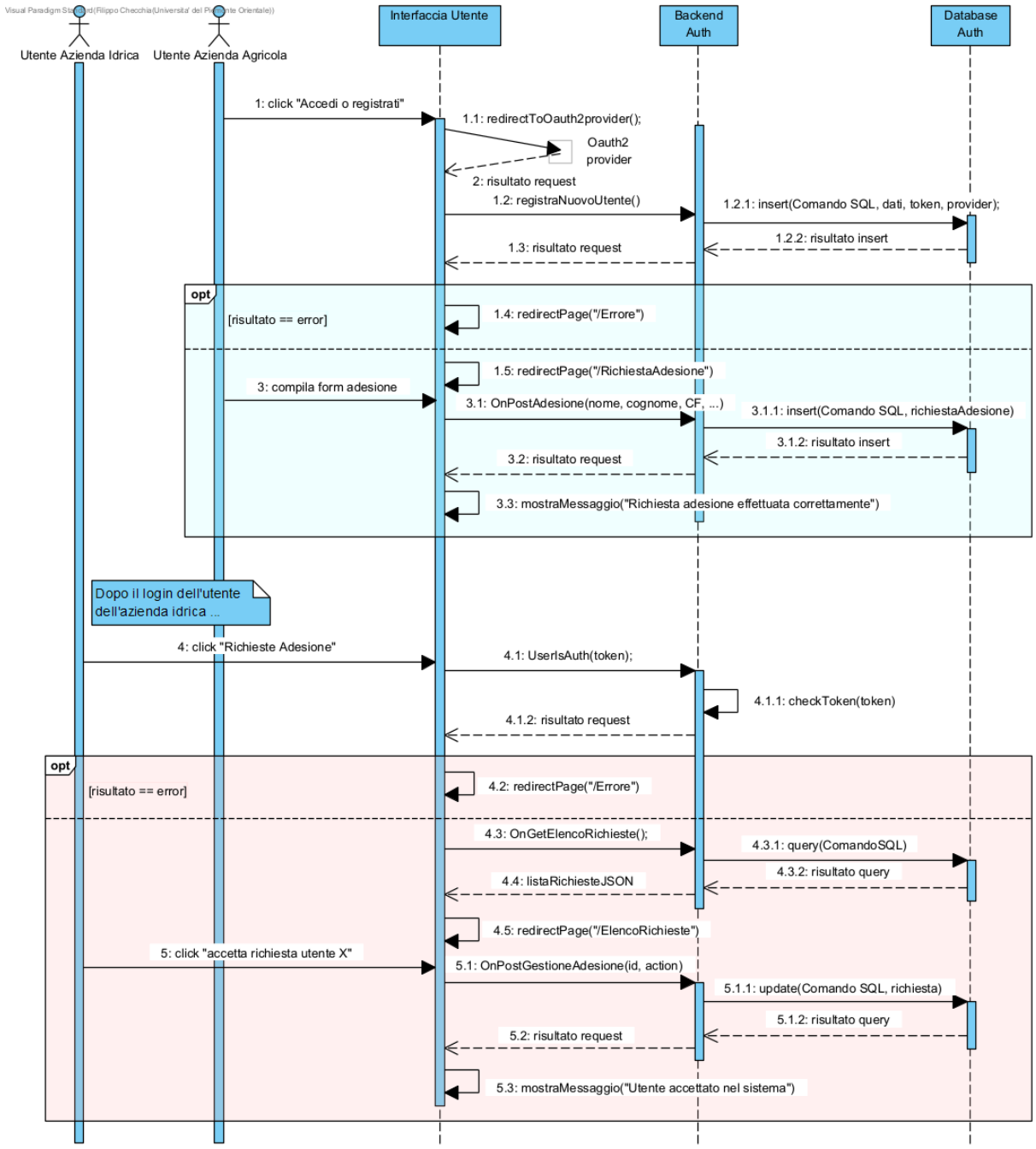


OpenId Manager è una libreria il cui scopo è quello di gestire una lista di provider oauth e memorizzare per ogni provider le chiavi RSA per verificare la firma dei token JWT.

- **RemoteJwksHub**: il suo scopo è quello di tenere una lista di provider oauth identificati dall'url del provider che restituisce le informazioni su come verificare i token da essi generati.
- **RemoteManager**: il suo scopo è quello di effettuare le operazioni di base sul singolo provider oauth, quali: leggere le chiavi fornite dal provider per verificare la firma dei token, effettuare query periodiche a tale url per verificare se le chiavi siano cambiate, esporre un metodo per ottenere una chiave identificata dal campo kid presente nel token jwt.
- **ProviderInfo**: è un oggetto, in genere configurato con dati presi dal database, che contiene informazioni cui: l'url alla pagina di configurazione openid del provider, il nome usato internamente per identificare il provider, e la lista di audience accettati per verificare che il token presentato sia stato effettivamente generato per l'applicazione di questo progetto e non per servizi terzi.
- **OpenidConfiguration**: classe contenente i nomi di alcuni dei campi presenti nel documento json presente all'url con le informazioni di configurazione di un provider OpenId.
  URL di esempio: https://www.facebook.com/.well-known/openid-configuration
  Non tutti i campi forniti dal provider vengono utilizzati, quindi sono quelli essenziali sono presenti nella classe.

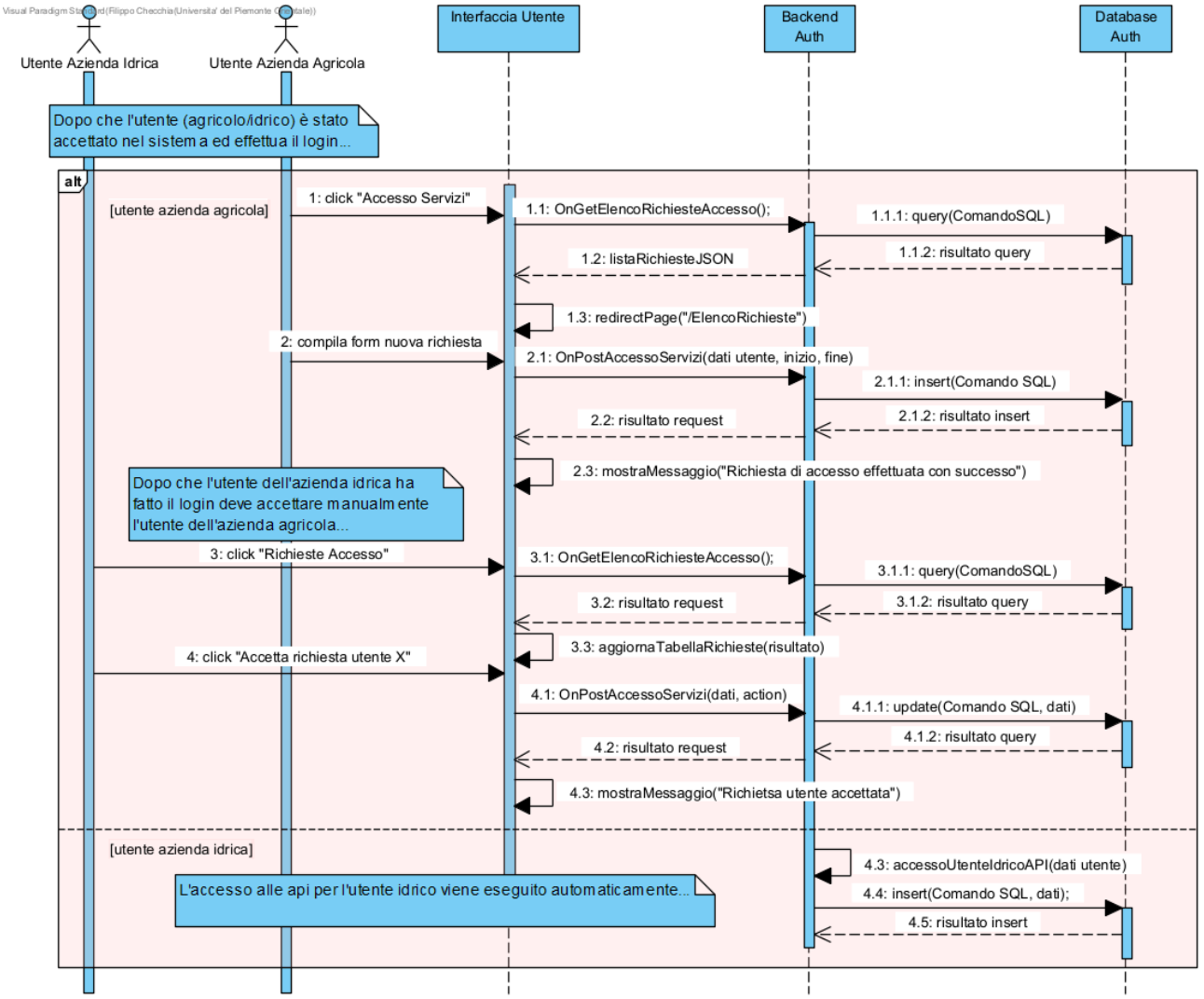
Diagrammi di sequenza

Adesione al sistema idrico - Tutti i tipi di utente



La richiesta di adesione al sistema può essere fatta dopo l'autenticazione dell'utente mediante Oauth2. L'utente autenticato dovrà inserire i dati mancanti quali nome, cognome, codice fiscale, partita iva dell'azienda e il tipo, per fare richiesta di adesione al sistema.
SOLO gli utenti delle aziende idriche potranno accettare o rifiutare le richieste di adesione al sistema.
Una volta approvata la richiesta, al fine di accedere ai servizi di gestione dell'azienda, l'utente dovrà far richiesta anche del periodo di accesso al sistema (il periodo lavorativo, da quando a quando l'utente avrà l'accesso alle api) con il medesimo procedimento.

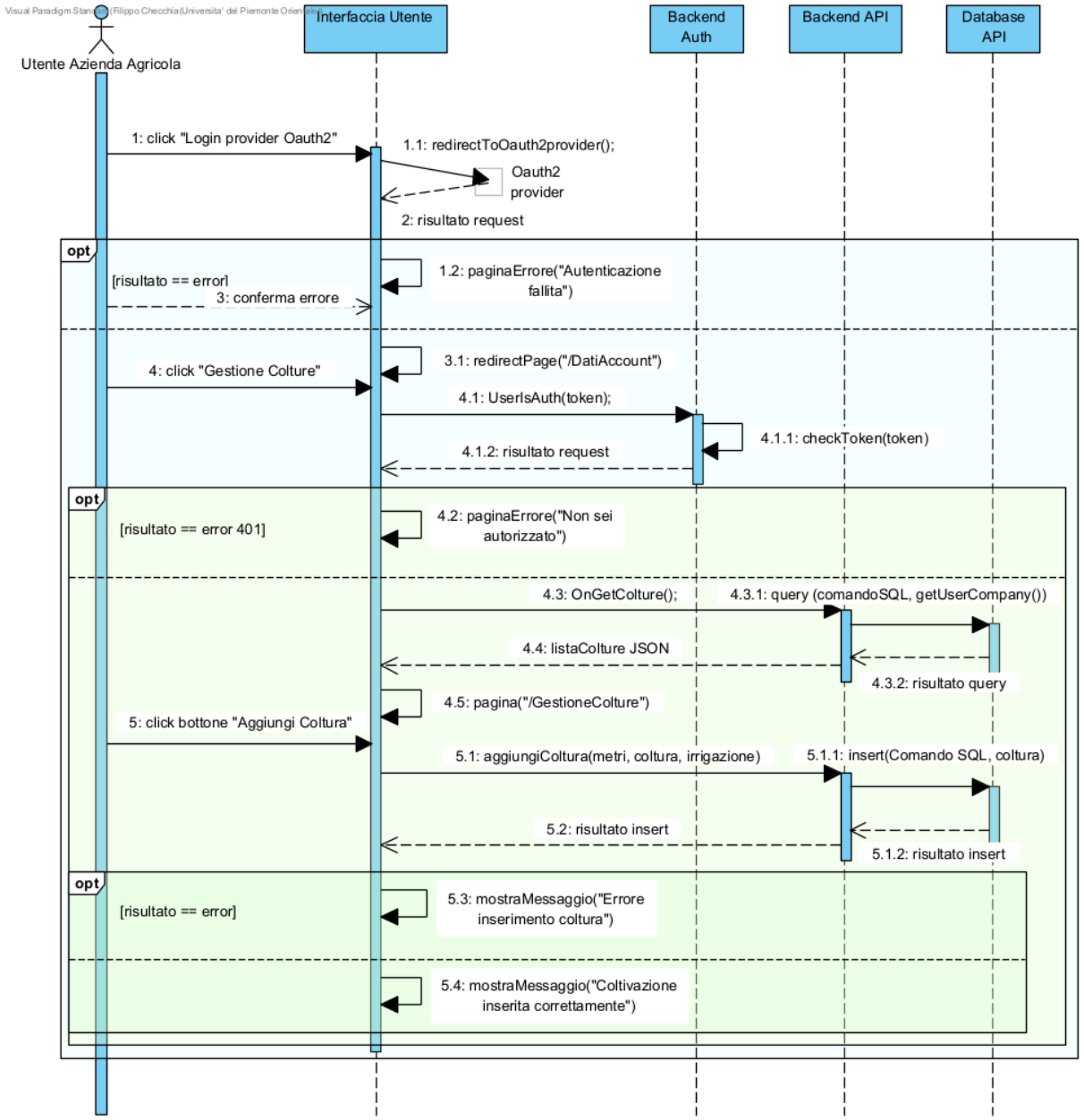
Accesso al sistema idrico - Tutti i tipi di utente



L'accesso al sistema idrico tramite le API (quindi per poter effettuare le operazioni sui servizi come gestione delle offerte, delle colture, dello storico delle misurazioni, ecc…) avviene:

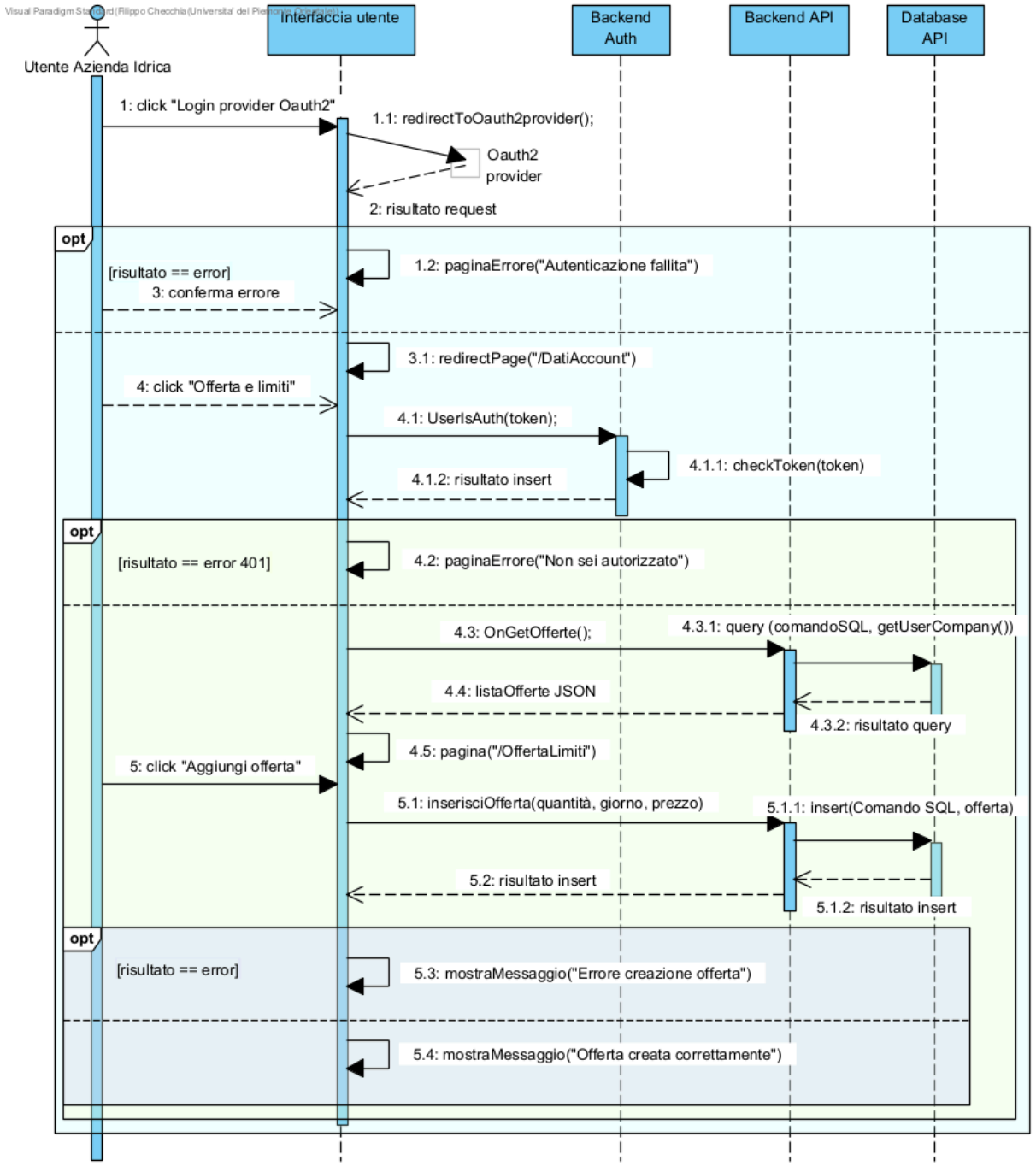
- per **l'utente dell'azienda agricola**: dopo l'accesso e l'accettazione nel sistema, l'utente deve effettuare la richiesta di accesso alle API per effettuare le operazioni. La richiesta deve essere accettata dall'utente dell'azienda idrica, in questo modo si potrà generare l'access token per effettuare le operazioni.
- per **l'utente dell'azienda idrica:** dopo l'accesso e l'accettazione nel sistema, l'utente avrà subito l'accesso alle API.

Aggiunta coltivazione - Azienda Agricola



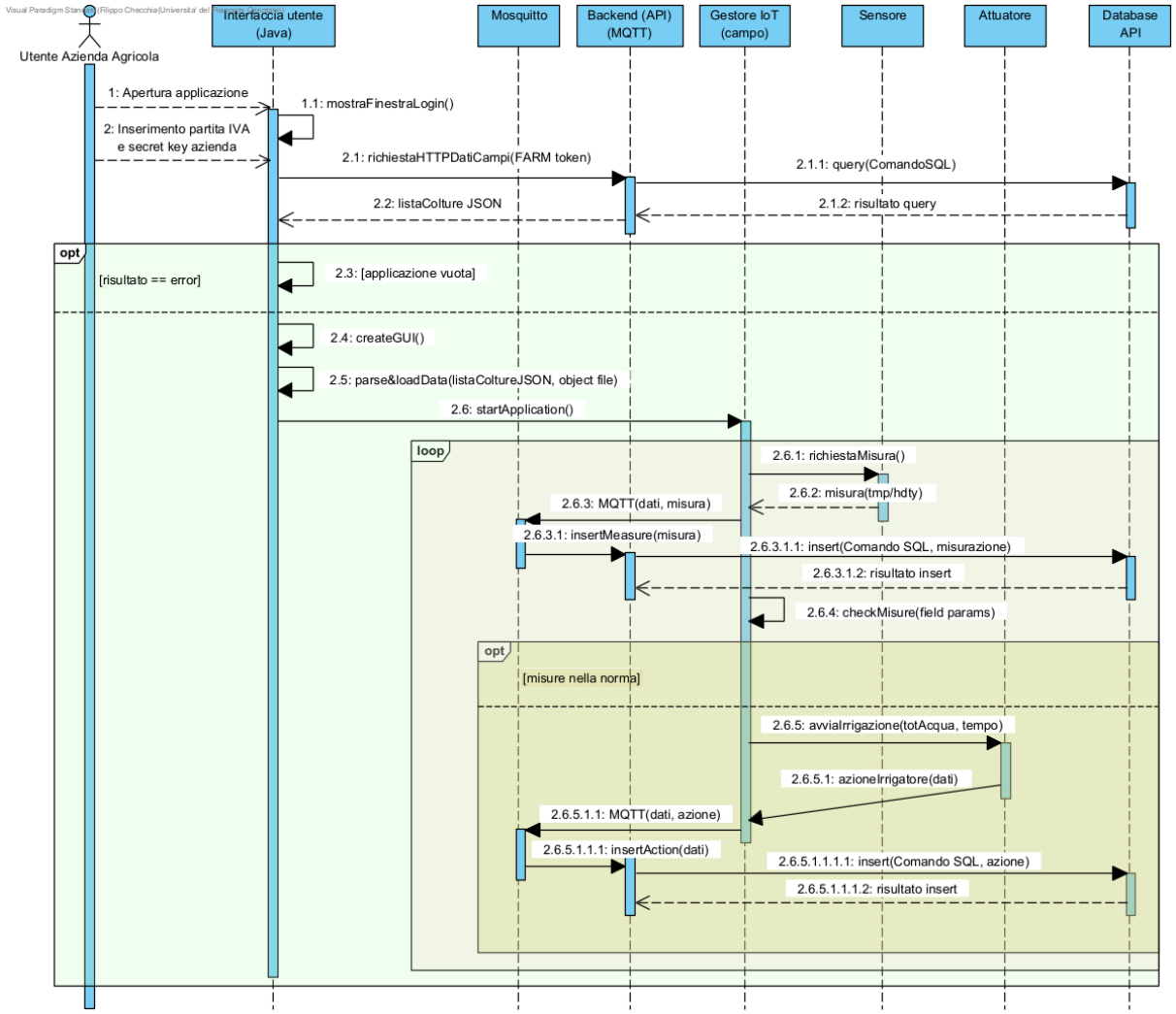
L'aggiunta di una coltivazione viene effettuata dall'utente dell'azienda agricola previa autenticazione e autorizzazione tramite token fornito dal provider Oauth2 (google o facebook). La coltura verrà aggiunta nella tabella specifica del database API per quella determinata azienda per il quale l'utente lavora.

Inserisci offerta idrica - Azienda Idrica



L'inserimento di una nuova offerta idrica viene effettuato dall'utente dell'azienda idrica previa autenticazione e autorizzazione tramite token (formato OpenId) fornito dal provider Oauth2 (google o facebook). L'utente inserirà i dati della nuova offerta nell'apposito form e verrà creato un nuovo record nel database API.

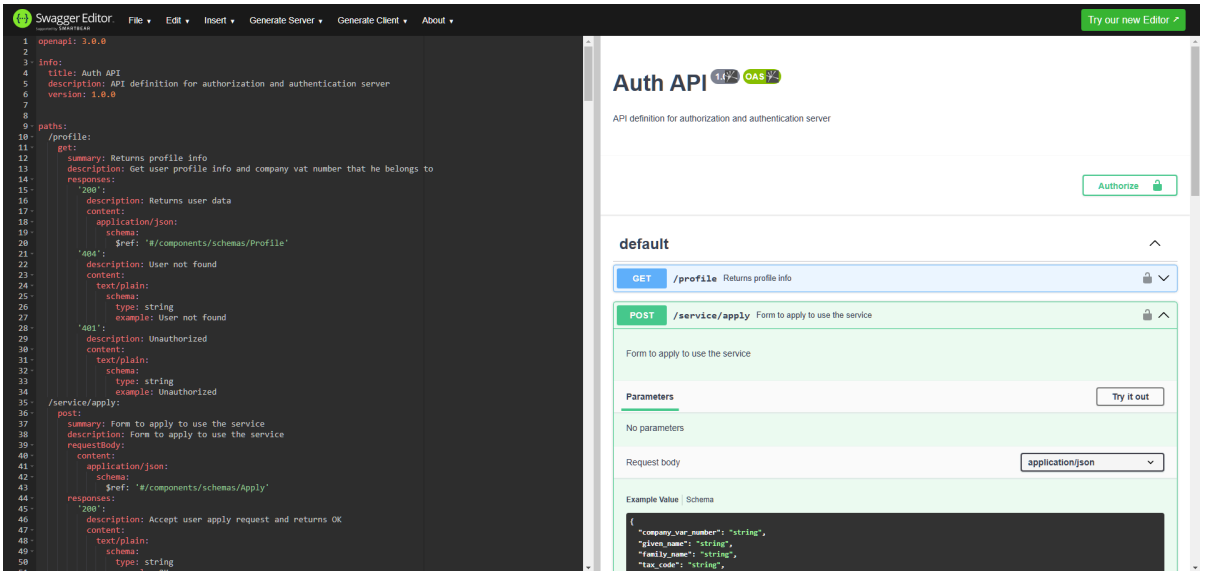
Gestione dispositivi IOT e misure - Azienda Agricola



L'utente "accede" all'applicazione Java per la gestione dei dispositivi quali sensori e attuatori tramite partita Iva e secret key dell'azienda agricola. Effettuato l'accesso vengono prelevati i dati dei campi; per ognuno di essi si andrà a creare un gestore IOT e caricare da file la lista di sensori e attuatori già esistenti per i campi già presenti nel database. Una volta caricato i dispositivi da file e associati ai campi partiranno in loop ogni tot tempo le misurazioni dei sensori, e la valutazione da parte del gestore IOT della gestione degli attuatori. Ogni dato da gestore IOT del campo a backend è passato tramite MQTT (cifrato con secret key dell'azienda, inviato tramite formato JSON); i dati verranno salvati nel rispettivo Database. (si veda il capitolo Topic MQTT per i dettagli dei messaggi scambiati).

Definizione API rest

Per definire le **API REST** si sono utilizzati dei file in formato .yaml presenti su GitLab nella cartella “api-definition”. Per visualizzare le api è necessario copiare e incollare il contenuto del file nell’editor online Swagger al link https://editor.swagger.io/ . All’interno della cartella sono presenti il file auth.yaml, che contiene le informazioni riguardanti le API di autenticazione/autorizzazione, e il file api.yaml contenente le API delle azioni principali della piattaforma idrica.
Di seguito un’immagine della visualizzazione dell’editor:



Sottosistema IOT

Il sottosistema IOT (Applicazione Java) è formato da più componenti:

- **Gestore IOT** : Componente che monitora e controlla sensori e attuatori per una determinata coltivazione. Si connette a un broker MQTT e invia messaggi per aggiornare lo stato, inviare misurazioni o attivare dispositivi, pubblicando sui topic definiti per comunicare con il backend (vedere i topic sotto). Il gestore IoT ha il compito di richiedere le misure ai sensori (per poi fare una media della temperatura/umidità) e in base a questo accendere gli attuatori per l’irrigazione della coltivazione, comunicando al backend MQTT misurazioni e azioni dei dispositivi IoT.

- **Sensori** : Simula un dispositivo IoT (sensore) che rileva valori di temperatura e umidità quando il gestore le richiede. Le misurazioni vengono influenzate anche dalle condizioni meteo.

- **Attuatori** : Simula un dispositivo IoT (irrigatore) che passa tra stati *ON* e *OFF*, controllato tramite il gestore IOT del relativo campo. Passa dati come durata e quantità d'acqua erogata. Un timer interno lo spegne automaticamente dopo un certo tempo, aggiornando lo stato.

- **Simulazione meteo** : Rappresentata dalla classe “Weather” simula le condizioni meteorologiche aggiornando periodicamente lo stato del meteo tra "sole", "nuvoloso", "pioggia" e "temporale" in base a probabilità predefinite. Utilizza un timer per cambiare automaticamente il meteo a intervalli regolari.

- **GUI** : Interfaccia grafica per il sottosistema IoT, nella quale è possibile gestire i sensori e gli attuatori, vedere il riepilogo dei dispositivi nei campi, l'acqua rimasta, lo stato del meteo e i messaggi scambiati.

- **Main** : Classe che gestisce l'intero sistema IoT simulato. Ha il compito di richiedere i dati sui gestori e i relativi dispositivi IoT comunicando tramite HTTP il backend per ricostruire l'intero sistema di dispositivi avviando l'interfaccia grafica e i componenti della simulazione.

Topic MQTT

Ogni campo dell'azienda agricola ha un suo gestore IOT, con i relativi sensori e attuatori.
Il gestore IOT del campo invia i dati con MQTT (misurazioni dei sensori e azioni degli attuatori) sui **topic**:

- backend/measure/umdty
- backend/measure/tmp
- backend/measure/actuator

Il formato del messaggio MQTT scambiato è JSON, con all'interno i seguenti campi:
{type: <sensore/attuatore>, vat_number: <>, obj_id: <>, data: <>, log_timestamp: <>}

Dove:

- *type*: tipo dell'oggetto
- *vat_number*: partita IVA dell'azienda del quale fa parte il campo, il gestore IOT e i relativi sensori/attuatori.
- *obj_id*: id del dispositivo IOT
- *data*: dati dell'oggetto preso in considerazione
    - sensore umidità: {value: <float>}
        - value: valore di umidità rilevato (0-100%)
    - sensore temperatura: {value: <float>}
        - value: valore di temperatura rilevato
    - attuatore irrigatore: {state: ON, period: <int>, water_used: <float>}
        - state: attivazione dell'attuatore
        - period: periodo di tempo in ms
        - water_used: quantità di acqua erogata
- *log_timestamp*: data e ora della misura/azione

Il **messaggio MQTT** da inviare sarà poi **cifrato** nel seguente modo:

- *msg_cifrato* = HMACSHA256(stringa JSON) con chiave privata dell'azienda
- Base64Url(*msg_cifrato*)