COMMpla
Communication Platforms and Online Solutions

Project Description

The objective of this project is to design and implement a **POC of a "CRM" system** capable of managing the following minimal core entities:

- Users
- Person
- Company
- WebformSubmission

The CRM will expose a set of RESTful APIs required to seamlessly integrate with a custom Drupal module developed by COMMpla and provided for the task.

This module will be responsible for periodically sending contact data and webform submissions to the CRM via a cron-based process, ensuring reliable and asynchronous data synchronization.

System Architecture

The entire solution will be **containerized using Docker** and will include at least the following components:

1. **Drupal Instance**
    - Configured with the custom module installed (provided)
    - Responsible for collecting contact data and webform submissions
    - Periodically sends data to the CRM APIs via cron jobs
2. **Frontend Application**
    - Provides a user interface for visualizing and managing Users, Persons, and Webform Submissions
    - Includes search and filtering capabilities for all supported entities

3. **Backend API**
    - Exposes REST APIs to receive, store, and retrieve data from Drupal
    - Implements the business logic of the CRM
4. **Database Layer**
    - Based on MySQL
      Stores all CRM entities and related metadata

Technology Stack

The project will leverage the following technologies:

- **Backend / API:** PHP or Django
- **Frontend:** React or Angular
- **CMS Integration:** Drupal (with a custom module)
- **Database:** MySQL
- **Infrastructure:** Docker (multi-container setup)

Expected Outcome

The final outcome will be a fully containerized POC demonstrating:

- End-to-end data flow from Drupal to the CRM
- Proper API-based integration between systems
- A functional frontend for data visualization and search
- A modular and extensible architecture suitable for future production development

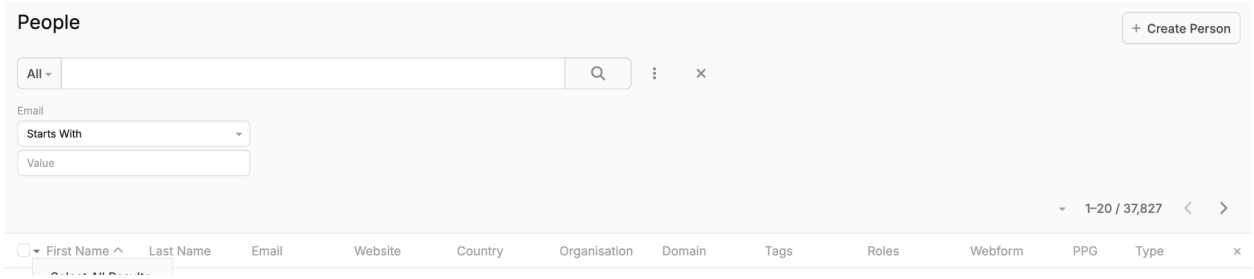
The frontend must support the following user stories:

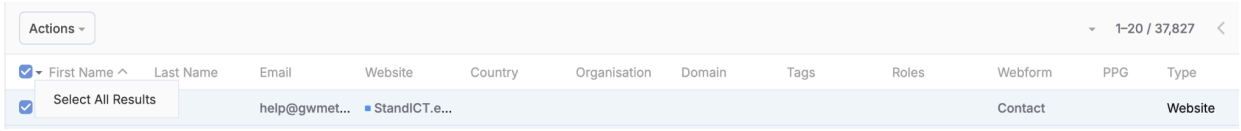
- Enable full CRUD operations (Create, Read, Update, Delete) for the Person entity.
- Implement a mechanism to receive webform submission data, where the email address is always mandatory.
- Automatically associate each webform submission with an existing Person in the CRM, based on the email address.
- If the related Person does not exist, automatically create a new Person record and link it to the incoming webform submission.

COMMpla
Communication Platforms and Online Solutions

- Ensure that webform submission data is displayed in a flexible and dynamic way, considering that the structure and fields of webforms may vary.
- Include the source website as one of the tags associated with the Person and/or the webform submission when data is pushed to the CRM.
- Provide the ability to import Person data via CSV, through a guided import wizard that allows users to map CSV fields to CRM attributes.

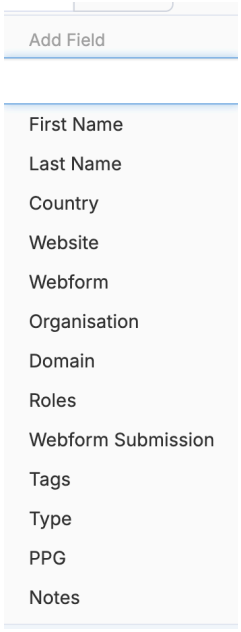
Frontend Interface (desired)





COMMpla
Communication Platforms and Online Solutions

Search selecting fields:



COMMpla
Communication Platforms and Online Solutions

CRUD API Operations

- List
- Read
- Create
- Update
- Delete

List

`GET {entityType}`

Returns a list of records of a specific entity type.

GET parameters:

- **`maxSize`**
- **`offset`**
- **`orderBy`** – *string*
- **`order`** – *asc*|*desc*.
- **`select`** – *string* – Specifies which fields should be included in the response. If omitted, the response will contain all available fields.
- **`where`** – *array* – filters.

Example:

`GET Account?offset=0&maxSize=20`

Returns:

```
{
  "list": [... array of records...],
  "total": {total Count Of Records}
}
```

Read

`GET {entityType}/{id}`

Returns attributes of a specific record.

Example:

`GET Account/5564764442a6d024c`

Create

`POST {entityType}`

Creates a new record of a specific entity type.

Payload: Object of entity attribute

Returns attributes of the created record.

Headers:

- **`Content-Type: application/json`**

Example:

`POST Account`

Payload:

```
{
  "name": "Test",
  "assignedUserId": "1"
}
```

Update

`PUT {entityType}/{id}`

Updates an existing record.

Payload: Object of entity attribute needed to be changed.

Returns attributes of the updated record.

Headers:

- **`Content-Type: application/json`**

Example:

`PUT Account/5564764442a6d024c`

Payload:

```
{
  "assignedUserId": "1"
}
```

Delete

`DELETE {entityType}/{id}`

Deletes an existing record.

Returns **`true`**.

Example:

`DELETE Account/5564764442a6d024c`

Entity Type

An **entity type** represents a type of data, or object.

Examples:

Account, Webform, Person

General Naming Convention

- Entity types use **UpperCamelCase**
- They always start with a **capital letter**

Example:

Account, Webform, Person

Entity Structure

Each entity type definition includes:

- **Fields**
- **Links**

Field

A **field** is a unit of data.

Each entity type defines its own set of fields.

Examples:

name, status, createdAt, assignedUser

COMMpla
Communication Platforms and Online Solutions

Field Attributes

Each field can have one or more attributes, depending on the field type.

The list of available field types is available in the official documentation.

Attribute

An attribute represents the actual stored value of a field.

Key Concepts

- Attributes usually correspond to database columns, if they are storable
- In the REST API, JSON object keys correspond to attributes

Single vs Multiple Attributes

Most fields have one attribute with the same name as the field.

Example:

- Field: createdAt
- Attribute: createdAt

Some field types expose multiple attributes.

COMMpla
Communication Platforms and Online Solutions

Field Types with Multiple Attributes

Field Type	Attributes
Link	fieldId, fieldName
Link-Multiple	fieldIds, fieldNames (IDs array, names map ID → name)
Link-Parent	fieldId, fieldType, fieldName (if needed)
Currency	field, fieldCurrency
Person Name	firstName, lastName, middleName, salutationName
Address	fieldStreet, fieldCity, fieldPostalCode, fieldCountry, fieldState

In the table above, field refers to the name of the field.

Link

A link represents a relationship with another entity type.

Relationships

Each relationship between two entity types is defined by a pair of links, one on each side of the relationship.

Link Types

The available link types should be:

- **belongsTo**
- **hasMany**

- **hasOne**
- **belongsToParent**
- **hasChildren**

Relationship Types Mapping

Relationship Type	Link Types
One-to-Many	hasMany + belongsTo
Many-to-One	belongsTo + hasMany
Many-to-Many	hasMany + hasMany
One-to-One (Right)	belongsTo + hasOne
One-to-One (Left)	hasOne + belongsTo
Parent-to-Children	hasChildren + belongsToParent

Users Management

User Types

The system supports the following types of users:

- **Admin**
- **Regular**

Admin

An **Admin** user has full access to the system.

Key capabilities:

- Create and remove users
- Manage roles and access permissions
- Manage teams and portals

- Create and manage other admin users

There can be multiple admin users in the system.

Regular

A Regular user has access only to the scopes defined by the Roles assigned to them.

Permissions and limitations:

- Can edit their own user record (if allowed by roles)
- Cannot edit the following fields:
    - User Name
    - Type
    - Emails
    - Roles
    - Is Active

Sending Access Info (optional)

When an admin creates a regular, admin user, they can send an access info email.

Access Info Email Flow

1. The admin fills in the Email field on the user creation form
2. The Send access info checkbox becomes available
3. After user creation, an access email is sent automatically

The access info email includes:

- A link to the system
- The username

User Inactivating

To disable a user without deleting their record:

- The admin unchecks the Is Active field

This immediately prevents the user from accessing the system.

COMMpla
Communication Platforms and Online Solutions

Entity

WebForm:

Label	Name	Type
Assigned User	assignedUser	Link
Contact	contact	Link
Created At	createdAt	Date-Time
Created By	createdBy	Link
Description	description	Text
DupId	dupId	Varchar
Modified At	modifiedAt	Date-Time
Modified By	modifiedBy	Link
Name	name	Varchar
Persons	persons	Link Multiple
Tags	tags	Multi-Enum
Teams	teams	Link Multiple
Webform Submissions	webformSubmissions	Link Multiple
Website	website	Link

COMMpla
Communication Platforms and Online Solutions

WebFormSubmission:

Label	Name	Type
Assigned User	assignedUser	Link
Contact	contact	Link
Created At	createdAt	Date-Time
Created By	createdBy	Link
Data	data	Text
Description	description	Text
DupId	dupId	Varchar
Modified At	modifiedAt	Date-Time
Modified By	modifiedBy	Link
Name	name	Varchar
Person	person	Link
Teams	teams	Link Multiple
Webform	webform	Link

COMMpla
Communication Platforms and Online Solutions

WebSite

Label	Name	Type
Assigned User	assignedUser	Link
Campaigns	campaigns	Link Multiple
Contact	contact	Link
Created At	createdAt	Date-Time
Created By	createdBy	Link
Description	description	Text
Instagram Followers	instagramFollowers	Link Multiple
Linkedin Followers	linkedinFollowers	Link Multiple
Meetings	meetings	Link Multiple
Modified At	modifiedAt	Date-Time
Modified By	modifiedBy	Link
Name	name	Varchar
Persons	persons	Link Multiple
Tags	tags	Multi-Enum
Teams	teams	Link Multiple
Twitter Followers	twitterFollowers	Link Multiple
Url	url	Url
Webforms	webforms	Link Multiple
Website Users	websiteUsers	Link Multiple

COMMpla
Communication Platforms and Online Solutions

WebSiteUser

Label	Name	Type
Assigned User	assignedUser	Link
Contact	contact	Link
Created At	createdAt	Date-Time
Created By	createdBy	Link
Data	data	Text
Description	description	Text
DupId	dupId	Varchar
Last Login	lastLogin	Date-Time
Modified At	modifiedAt	Date-Time
Modified By	modifiedBy	Link
Name	name	Varchar
Organisation	organisation	Text
Person	person	Link
Picture	picture	Image
PPG Accepted	ppgAccepted	Boolean
PPG updated	ppgUpdated	Date-Time
Teams	teams	Link Multiple
Website	website	Link
Created (Website)	websiteCreatedAt	Date-Time
Modified (Website)	websiteModifiedAt	Date-Time

COMMpla
Communication Platforms and Online Solutions

Person

Label	Name	Type
Address	address	Address
City	addressCity	Varchar
Country	addressCountry	Varchar
Map	addressMap	Map
Postal Code	addressPostalCode	Varchar
County	addressState	Varchar
Street	addressStreet	Text
Assigned User	assignedUser	Link
Type	contacttype	Enum
Created At	createdAt	Date-Time
Created By	createdBy	Link
Description	description	Text
DupId	dupId	Varchar
Email	emailAddress	Email
Email Address is Invalid	emailAddressIsInvalid	Boolean
Email Address is Opted-Out	emailAddressIsOptedOut	Boolean
First Name	firstName	Varchar
Tags	genericTags	Multi-Enum
Last Name	lastName	Varchar
Middle Name	middleName	Varchar

COMMpla
Communication Platforms and Online Solutions

Modified At	modifiedAt	Date-Time
Modified By	modifiedBy	Link
Name	name	Person Name
Notes	notes	Wysiwyg
Organisation	organisation	Varchar
Phone	phoneNumber	Phone
Phone Number is Invalid	phoneNumberIsInvalid	Boolean
Phone Number is Opted-Out	phoneNumberIsOptedOut	Boolean
PPG	ppgAccepted	Boolean
PPG Updated	ppgUpdated	Date-Time
Roles	roles	Multi-Enum
Salutation	salutationName	Enum
Domain	tags	Multi-Enum
Teams	teams	Link Multiple
Webform	webform	Link
Webform Submission	webformSubmission	Link One
Website	website	Link
WebsiteUrl	websiteUrl	Foreign
Website User	websiteUser	Link One

COMMpla
Communication Platforms and Online Solutions

API Search Parameters

Search parameters and filters can be used with API endpoints that return a list of records.

Parameters

Search parameters are passed as **query parameters** in a GET request.

offset

Type: Integer

Pagination offset.

maxSize

Type: Integer

Maximum number of records to return.

select

Type: String or Array of strings

List of attributes to return.

- Attributes must be comma-separated
- Whitespaces are not allowed

Example:

```
None
id,name,status,assignedUserId
```

JSON example:

COMMpla
Communication Platforms and Online Solutions

None

```
["id", "name"]
```

where

Type: Array

Search criteria definition.

primaryFilter

Type: String

boolFilterList

Type: Array

Boolean filters, e.g.:

- onlyMy

orderBy

Type: String

Attribute used for sorting.

order

Type: String

Sort direction.

Allowed values:

- asc
- desc

Where Items

The where parameter is an array of filter objects.

Items can be nested using logical operators:

- **equals - notEquals**
- **isNull - isNotNull**
- **isTrue - isFalse**
- **linkedWith - notLinkedWith**
- **For link-multiple fields.**
- **isLinked / isNotLinked For link-multiple fields.**
- **in / notIn**
- **contains - notContains - startsWith - like - notLike**
- **or / and**
- **Date Filters**
- **Date-Time Fields**
- **between**