



Self Service Company Profile Update System

Author(s)	@Fahmi Roihanul Firdaus
Acknowledgement	

Related Documents

Document Name	Document URL
Reverse Proxy	 NGINX Reverse Proxy
Custom Web Form	 How to Create a Generic Controller in Odoo 16 Website

1. Summary

 TLDR lo mau ngajuin apa, masalah lo apa dan bentuknya apa, pitch paragraph gitu.

Create **self-service company profile update system** with an **approval workflow** before changes take effect.


2. Problem and Motivation

 Latar belakang dan why problem, What's the current status of real condition, refer ke Problem Definition juga bisa.

We need reusable web form for client / customer to update their personal information with rules:

- Customers must only access their update form through a special encoded link
- The link SHOULD NOT reveal our main system's web address
- Each link SHOULD ONLY work for its specific customer
- Links SHOULD NOT be guessable or easily modified to access other customer data
- Links can be used every time, but when a pending request is exist, the system SHOULD NOT create a new request.

3. Detailed Design

 Bisa pasang sketching, C4 architecture, pseudocode, flow/algorithm, rancangan kode modul, cara penggunaan, constraint, ide teknis, dll yang spesifik teknis.

Design Flow

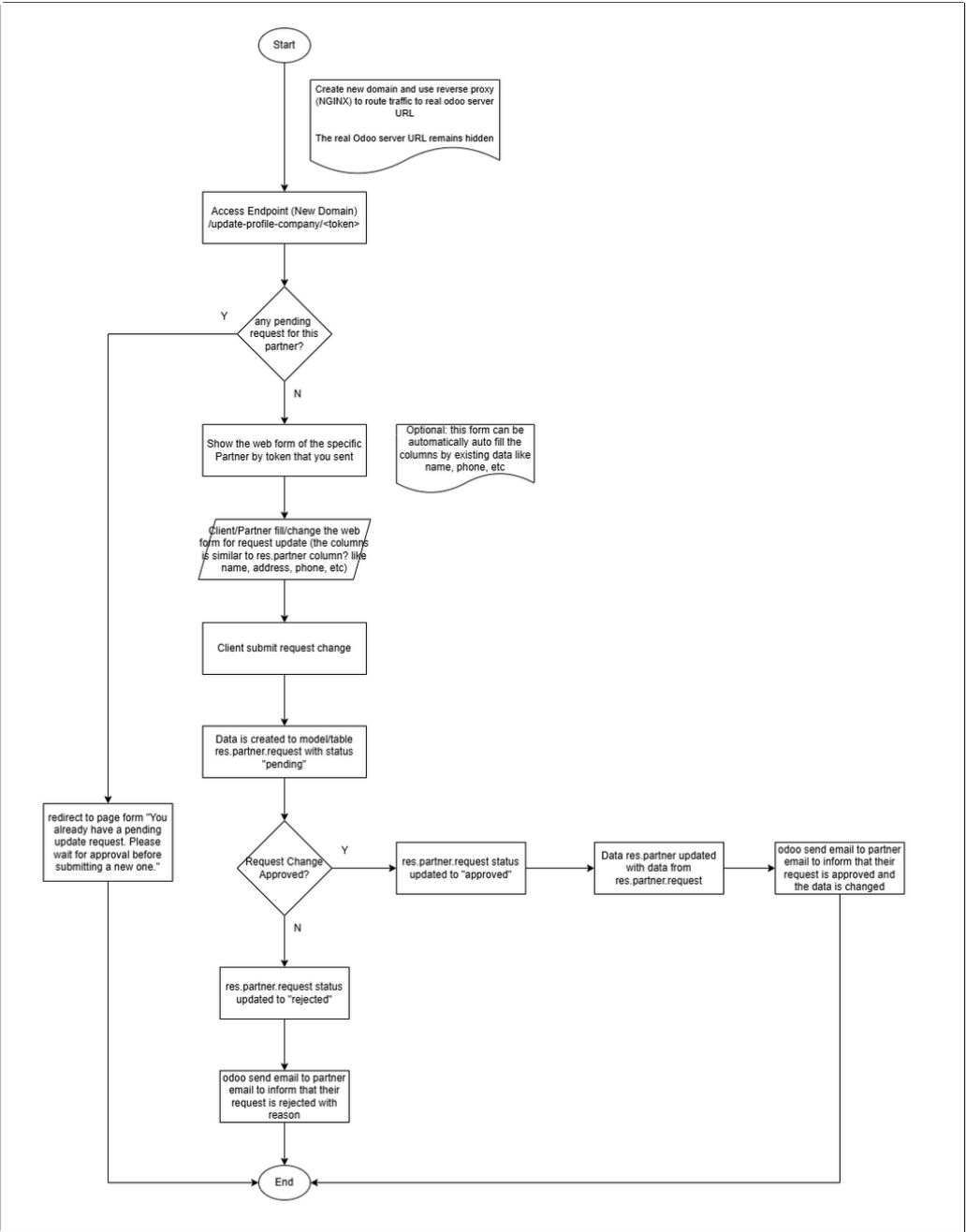


Table & Column

on this section, we define what column should be add on table res.partner.request, this column also reflect what is in the web form update request.

table / model	column	attribute
res.partner.request	state	<div>type: Selection</div> <div>options before:</div> <div><pre>1 [2 ("pending", "Waiting Approval"), 3 ("approved", "Approved"), 4 ("rejected", "Rejected"),</pre></div>

		5]
res.partner.request	token	type: Char, for store static token (we can use JWT or custom token encoded into base64)
res.partner.request	reject_reason	type Char, Reason why request is rejected
res.partner.request	name	type: Char, later this field will used to replace name on res.partner
res.partner.request	address	type: Char, later this field will used to replace street and street2 on res.partner
res.partner.request	city	type: Char, later this field will used to replace city on res.partner
res.partner.request	province	type: Char, later this field will used to replace state_id on res.partner
res.partner.request	zip	type: Char, later this field will used to replace zip on res.partner
res.partner.request	country	type: Char, later this field will used to replace country_id on res.partner
res.partner.request	phone	type: Char, later this field will used to replace phone on res.partner
res.partner.request	email	type: Char, later this field will used to replace email on res.partner
res.partner.request	website	type: Char, later this field will used to replace website on res.partner

Groups (Optional)

Add Independent group/access for res.partner.request, if you want to make specific user odoo to do this approval, we can create new group access to be able CRUD on table res.partner.request, also later this group can be improve if want approval layer (officer to manager) on res.partner.request

Views

On res.partner.request we hide button create/delete (optional) on model level (optional) so only data created from **self-service company profile update system** is exist, it's also prevent any human error from odoo user.

we only provide button approve and reject on form view for odoo user approve and reject the change request.

Views XML

```

1 <odoo>
2   <record id="view_partner_request_tree" model="ir.ui.view">
3     <field name="name">res.partner.request.tree</field>
4     <field name="model">res.partner.request</field>
5     <field name="arch" type="xml">
6       <tree create="false" delete="false">
7         <field name="partner_id"/>
8         <field name="name"/>
9         <field name="state"/>

```

```

10         </tree>
11     </field>
12 </record>
13
14 <record id="view_partner_request_form" model="ir.ui.view">
15     <field name="name">res.partner.request.form</field>
16     <field name="model">res.partner.request</field>
17     <field name="arch" type="xml">
18         <form string="Company Update Request" create="false" delete="false">
19             <sheet>
20                 <header>
21                     <button name="approve_request" type="object" string="Approve" class="oe_highlight"/>
22                     <button name="reject_request" type="object" string="Reject" class="btn-danger"/>
23                     <field name="state" widget="statusbar"
24                 </header>
25                 <group>
26                     <group>
27                         <field name="partner_id" readonly="1"/>
28                         <field name="name" readonly="1"/>
29                         <field name="address" readonly="1"/>
30                         <field name="zip" readonly="1"/>
31                         <field name="city" readonly="1"/>
32                         <field name="state_id" readonly="1"/>
33                         <field name="country_id" readonly="1"/>
34                     </group>
35                     <group>
36                         <field name="email" readonly="1"/>
37                         <field name="phone" readonly="1"/>
38                         <field name="website" readonly="1"/>
39                         <field name="reject_reason"/>
40                     </group>
41                 </group>
42             </sheet>
43         </form>
44     </field>
45 </record>
46
47 <menuitem id="menu_partner_request" name="Company Update Requests" parent="base.menu_base_partner"
48     action="action_partner_request"/>
49 </odoo>

```

Function Approve & Reject:

```

1 def approve_request(self):
2     """Approves the request and updates the partner record."""
3     for rec in self:
4         if rec.partner_id:
5             rec.partner_id.write({
6                 'name': rec.name,
7                 'email': rec.email,
8                 'phone': rec.phone,
9                 'street': rec.address,
10                'zip': rec.zip,
11                'city': rec.city,
12                'state_id': rec.state_id,
13                'country': rec.country_id,
14                'website': post.get('website'),
15            })
16            rec.state = 'approved'

```

```

17         # send email from mail template if needed
18
19     def reject_request(self):
20         """Rejects the request without updating the partner record."""
21         for rec in self:
22             if not rec.reject_reason:
23                 raise ValidationError(_("You need to fill reject reason first when reject request change"))
24
25             rec.state = 'rejected'
26             # send email from mail template if needed

```

Controllers

this is for public user can access our **self-service company profile update system**, we need 2 new endpoints:

- for client to access form request change
- for client to submit request change

Access Form Request Change

```

1  @http.route('/update-profile-company/<string:token>', type='http', auth='public', website=True)
2  def update_form(self, token, **kwargs):
3      """Render the company update form if the token is valid."""
4      partner = request.env['res.partner'].sudo().search([('token', '=', token)], limit=1)
5
6      if not partner:
7          return request.render('your_module.invalid_token_template')
8
9      # Check if there's a pending request
10     existing_request = request.env['res.partner.request'].sudo().search([
11         ('partner_id', '=', partner.id),
12         ('state', '=', 'pending')
13     ], limit=1)
14
15     if existing_request:
16         return request.render('your_module.pending_request_template')
17
18     return request.render('your_module.update_form_template', {'partner': partner})

```

Submit Request Change

```

1  @http.route('/submit-update', type='http', auth='public', methods=['POST'], csrf=False)
2  def submit_update(self, **post):
3      """Handles submission of the update form."""
4      token = post.get('token')
5      partner = request.env['res.partner'].sudo().search([('update_token', '=', token)], limit=1)
6
7      if not partner:
8          return request.render('your_module.invalid_token_template')
9
10     # Check for pending requests
11     existing_request = request.env['res.partner.request'].sudo().search([
12         ('partner_id', '=', partner.id),
13         ('state', '=', 'pending')
14     ], limit=1)
15
16     if existing_request:
17         return request.render('your_module.pending_request_template')
18

```

```

19 # Create a new request for staff approval
20 request.env['res.partner.request'].sudo().create({
21     'partner_id': partner.id,
22     'name': post.get('company_name'),
23     'email': post.get('email'),
24     'phone': post.get('phone'),
25     'address': post.get('address'),
26     'zip': post.get('zip'),
27     'city': post.get('city'),
28     'state_id': post.get('province'),
29     'country_id': post.get('country'),
30     'website': post.get('website'),
31     'state': 'pending',
32 })
33
34 return request.render('your_module.success_template')

```

Template Web

we create 4 web form for handle client needs:

- Form Access Request Change
- Form Request Pending
- Form Invalid Token
- Form Success Submit Request Change

Form Access Change Request

```

1 <t t-name="your_module.update_form_template">
2     <t t-call="website.layout">
3         <div class="container">
4             <h2>Update Your Company Information</h2>
5             <form action="/submit-update" method="post">
6                 <input type="hidden" name="token" t-att-value="partner.update_token"/>
7                 <label>Company Name:</label>
8                 <input type="text" name="name" t-att-value="partner.name" required/>
9                 <label>Address:</label>
10                <input type="text" name="address" t-att-value="partner.street" required/>
11                <label>Email:</label>
12                <input type="email" name="email" t-att-value="partner.email" required/>
13                <label>Phone:</label>
14                <input type="text" name="phone" t-att-value="partner.phone" required/>
15                <label>Website:</label>
16                <input type="text" name="website" t-att-value="partner.website" required/>
17                <label>City:</label>
18                <input type="text" name="city" t-att-value="partner.city" required/>
19                <label>Province:</label>
20                <input type="text" name="province" t-att-value="partner.state_id.name or ''" required/>
21                <label>Country:</label>
22                <input type="text" name="country" t-att-value="partner.country_id.name or ''" required/>
23                <label>ZIP Code:</label>
24                <input type="text" name="zip" t-att-value="partner.zip" required/>
25                <button type="submit">Submit</button>
26            </form>
27        </div>
28    </t>
29 </t>

```

Form Request Pending

```
1 <t t-name="your_module.pending_request_template">
2   <t t-call="website.layout">
3     <div class="container">
4       <h2>Update Request Pending</h2>
5       <p>You already have a pending update request. Please wait for approval before submitting a new one.
6     </div>
7   </t>
8 </t>
```

Form Invalid Token

```
1 <t t-name="your_module.invalid_token_template">
2   <t t-call="website.layout">
3     <div class="container">
4       <h2>Invalid Link</h2>
5       <p>The update link you used is invalid or expired.</p>
6     </div>
7   </t>
8 </t>
```

Form Success Submit Request Change

```
1 <t t-name="your_module.success_template">
2   <t t-call="website.layout">
3     <div class="container">
4       <h2>Success!</h2>
5       <p>Your update request has been submitted. Our team will review it soon.</p>
6     </div>
7   </t>
8 </t>
```

4. Dependencies [↗](#)

i Kalau ada kemungkinan dependensi ke sistem apa atau bikin dependensi untuk sistem apa.

- Odoo
- Nginx

5. Milestone/Deployment Strategy [↗](#)

i Ajuan cara adopsi, milestone development risetnya atau pemasangannya di production level atau saat implementasi riilnya.

...

6. Data Result [↗](#)

i Silakan isi dengan hasil data production-nya. Bisa diisi dengan Data Flow Diagram, Entity and Relationship Diagram, dan atau Data Model Documentation-nya.

...

7. Drawbacks/Risks/Possible Failures [↗](#)

i Kalau ada kemungkinan cons atau kekurangan dengan design ini, apa saja. Kemungkinan gagal dll-nya.

...

8. Alternatives [↗](#)

i Diluar ide ajuan ini ada apa lagi? Bisa masukkin RFC lain atau link services lain atau link artikel lain.

...

9. Unresolved/Future Possibilities [↗](#)

i Listing sesuatu yang masih abu-abu dan ga jelas, perlu riset lanjut, kemungkinan ke depannya.

...

10. Discussions [↗](#)

...