

Unlayer Email Builder Integration Guide

Complete Implementation for MakePlus Email Templates

Date: 27 January 2026

Version: 1.0

Status: ☒ Fully Implemented

Overview

The MakePlus platform now uses **Unlayer Email Editor** - a professional drag-and-drop email builder that allows creating beautiful, responsive emails without coding.

Key Benefits

- ☒ **No Coding Required** - Visual drag-and-drop interface
 - ☒ **Professional Templates** - Built-in components and layouts
 - ☒ **Responsive Design** - Mobile-friendly emails automatically
 - ☒ **Merge Tags** - Easy personalization with variables
 - ☒ **Re-editable** - Save designs and edit them later
 - ☒ **Preview Mode** - See exactly what recipients will see
 - ☒ **Clean HTML** - Email-client-compatible output
-

Architecture

How It Works

```
graph TD
    A[User Creates Email Template] --> B[Opens Unlayer Editor]
    B --> C[Designs Email Visually (drag & drop)]
    C --> D[Clicks Save]
    D --> E[Unlayer Exports:]
    E --> F["- HTML (body_html) → Used for sending emails<br>- JSON (builder_config) → Used for re-editing"]
    F --> G[Saved to Database]
    G --> H[Can Re-open and Edit Later]
```

Data Flow

```
# When Creating/Editing Template:
1. Unlayer editor.exportHtml() → Returns HTML + Design JSON
2. HTML saved to: template.body_html (and template.body for compatibility)
3. JSON saved to: template.builder_config
4. Metadata saved: name, subject, type, is_active

# When Sending Email:
1. Load template.body_html
2. Replace merge tags: {{event_name}} → "Tech Conference 2026"
3. Send HTML email via Django send_mail()

# When Re-editing:
1. Load template.builder_config
2. editor.loadDesign(config) → Restores exact design state
3. User makes changes
4. Export and save again
```

Features Implemented

1. Global Email Templates

URL: </dashboard/email-templates/create/>

Features:

- Full-screen Unlayer editor
- Settings modal for metadata
- Preview modal
- Merge tags panel
- Auto-save design configuration

2. Event-Specific Email Templates

URL: /dashboard/events/{event_id}/email-templates/create/

Features:

- Same Unlayer editor
- Event context display
- Can duplicate from global templates
- Event-specific merge tags

3. Editor Components

Available in Unlayer:

- Text blocks
- Images

- Buttons (CTA)
- Dividers
- Social media icons
- Columns (multi-column layouts)
- Headers/Footers
- HTML blocks (for custom code)

4. Merge Tags

Available Variables:

- `{{event_name}}` - Event name
- `{{event_location}}` - Event location
- `{{event_start_date}}` - Start date
- `{{event_end_date}}` - End date
- `{{participant_name}}` - Full name
- `{{first_name}}` - First name only
- `{{last_name}}` - Last name only
- `{{email}}` - Email address
- `{{telephone}}` - Phone number
- `{{etablissement}}` - Institution
- `{{badge_id}}` - Badge ID
- `{{qr_code_url}}` - QR code URL

Usage in Unlayer:

1. Add text block
2. Click "Merge Tags" in toolbar
3. Select variable from dropdown
4. Variable inserted automatically

Technical Implementation

Frontend (Templates)

email_template_form.html

```
<!-- Unlayer Container -->
<div id="email-editor-container"></div>

<!-- Hidden Form Fields -->
<input type="hidden" name="body_html" id="body_html">
<input type="hidden" name="builder_config" id="builder_config">

<!-- Load Unlayer -->
<script src="https://editor.unlayer.com/embed.js"></script>
```

```

<!-- Initialize -->
<script>
emailEditor = unlayer.createEditor({
  id: 'email-editor-container',
  projectId: 1234, // Optional: Your Unlayer project ID
  displayMode: 'email',
  mergeTags: [
    { name: 'Event Name', value: '{{event_name}}', sample: 'Tech Conference
2026' },
    // ... more tags
  ]
});

// On Save
emailEditor.exportHtml(function(data) {
  document.getElementById('body_html').value = data.html;
  document.getElementById('builder_config').value =
JSON.stringify(data.design);
  form.submit();
});

// On Load (for editing)
emailEditor.loadDesign(existingDesignJSON);
</script>

```

Backend (Views)

views_email.py

```

@login_required
def email_template_create(request):
    if request.method == 'POST':
        name = request.POST.get('name')
        subject = request.POST.get('subject')
        body_html = request.POST.get('body_html') # Unlayer HTML
        builder_config = request.POST.get('builder_config') # Unlayer JSON

        template = EmailTemplate.objects.create(
            name=name,
            subject=subject,
            body=body_html, # Backward compatibility
            body_html=body_html, # New field
            builder_config=builder_config, # New field
            template_type=request.POST.get('template_type', 'custom'),
            is_active=True,
            created_by=request.user
        )

        return redirect('dashboard:email_template_list')

```

```
return render(request, 'dashboard/email_template_form.html', {
    'template_variables': [...],
})
```

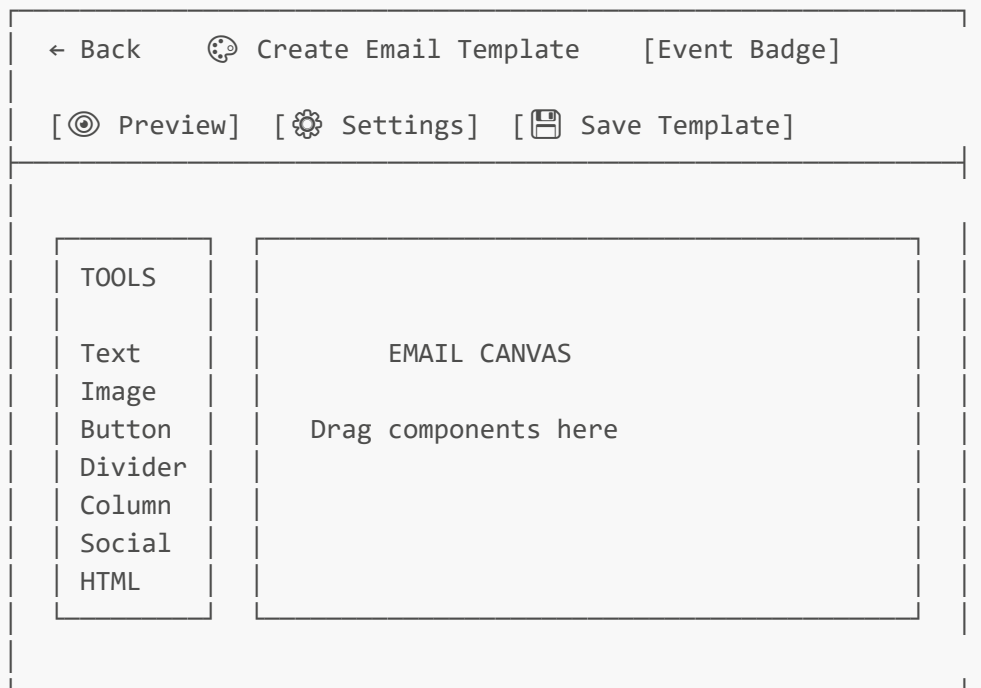
Database Schema

```
class EmailTemplate(models.Model):
    # ... existing fields ...

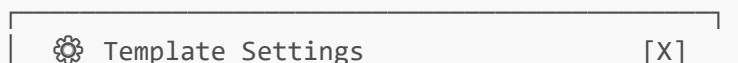
    # NEW: Unlayer fields
    body_html = models.TextField(blank=True) # Unlayer HTML output
    builder_config = models.JSONField(default=dict, blank=True) # Unlayer
design JSON
    template_type = models.CharField(max_length=30, default='custom')
    is_active = models.BooleanField(default=True)
```

🧠 User Interface

Main Editor Screen



Settings Modal



Template Name: [_____]

Email Subject: [_____]

Template Type: [Invitation ▼]

[✓] Template Active

i Available Merge Tags:

{{event_name}} {{first_name}}
{{email}} {{badge_id}} ...

[Close] [Save Settings]

Preview Modal

👁 Email Preview

[X]

Subject: Welcome to Tech Conference 2026

[Email content renders here in iframe]

Exactly as recipients will see it

🔧 Configuration

Unlayer Project ID (Optional)

If you have an Unlayer account with a project ID:

```
emailEditor = unlayer.createEditor({  
  id: 'email-editor-container',  
  projectId: YOUR_PROJECT_ID, // Replace with actual ID
```

```
// ... other options
});
```

Benefits of using Unlayer project ID:

- Save templates in Unlayer cloud
- Access premium features (if subscribed)
- Use custom fonts
- Custom colors palette
- Team collaboration

Without project ID:

- Still fully functional
- All designs stored in your database
- Free forever

Customizing Unlayer

```
emailEditor = unlayer.createEditor({
  id: 'email-editor-container',
  projectId: 1234,
  displayMode: 'email', // or 'web' for landing pages

  // Appearance
  appearance: {
    theme: 'light', // or 'dark'
    panels: {
      tools: {
        dock: 'left' // or 'right'
      }
    }
  },

  // Features
  features: {
    preview: true,
    imageEditor: true,
    textEditor: {
      tables: true,
      emojis: true
    }
  },

  // Disable specific tools
  tools: {
    form: {
      enabled: false // Disable form components
    }
  },
});
```

```
// Merge tags
mergeTags: [
  {
    name: 'Event Name',
    value: '{{event_name}}',
    sample: 'Tech Conference 2026'
  }
]
});
```

Usage Workflows

Creating a New Email Template

1. Navigate to Templates

- Dashboard → Email Templates → Create
- OR Events → [Event] → Email Templates → Create

2. Wait for Editor to Load

- Loading overlay appears
- Unlayer editor initializes (~2-3 seconds)
- Editor ready!

3. Design Your Email

- Drag components from left panel
- Drop on canvas
- Click to edit text, images, buttons
- Use merge tags from toolbar
- Style with visual controls (colors, fonts, spacing)

4. Configure Settings

- Click "Settings" button
- Enter template name (internal use)
- Enter email subject (what recipients see)
- Select template type (invitation, confirmation, etc.)
- Set active status
- Click "Save Settings"

5. Preview

- Click "Preview" button
- See exactly how email will look
- Check subject line
- Verify merge tags are present

6. Save Template

- Click "Save Template"
- HTML and design JSON exported automatically
- Redirects to template list
- Success message shown

Editing an Existing Template

1. Open Template

- Dashboard → Email Templates
- Click "Edit" on desired template

2. Editor Loads with Saved Design

- Unlayer initializes
- Loads builder_config JSON
- All elements restored exactly as saved

3. Make Changes

- Edit any element
- Add/remove components
- Update styles

4. Update Settings (if needed)

- Click "Settings"
- Modify name, subject, type, etc.

5. Save Changes

- Click "Save Template"
- New HTML and JSON exported
- Database updated

Duplicating a Template

For Event-Specific Templates:

1. Go to Event → Email Templates
2. Click "Use as Base" on global template
3. Unlayer loads with global template design
4. Customize for event
5. Save as new event template

Sending Emails

Process:

1. User creates/edits template in Unlayer
2. Template saved with HTML in `body_html`
3. When sending email:
 - Load `body_html` from database
 - Replace merge tags with actual values
 - Send via Django `send_mail()`

Example:

```
# views_email.py - send_event_email()

template_html = template.body_html # Unlayer HTML

# Replace merge tags
context = {
    'event_name': event.name,
    'event_location': event.location,
    'participant_name': participant.user.get_full_name(),
    # ... more variables
}

email_body = template_html
for key, value in context.items():
    email_body = email_body.replace(f"{{{key}}}", str(value))

# Send email
send_mail(
    subject=template.subject,
    message=strip_tags(email_body), # Plain text version
    from_email=settings.DEFAULT_FROM_EMAIL,
    recipient_list=[participant.user.email],
    html_message=email_body, # HTML version
)
```

Troubleshooting

Editor Not Loading

Problem: White screen, editor doesn't appear

Solutions:

1. Check browser console for errors
2. Verify Unlayer script loaded: <https://editor.unlayer.com/embed.js>
3. Check internet connection (Unlayer loads from CDN)
4. Try different browser
5. Disable browser extensions (AdBlock, etc.)

Design Not Saving

Problem: Click Save but design not persisted

Solutions:

1. Check browser console for JavaScript errors
2. Verify `builder_config` field in database accepts JSON
3. Check Django CSRF token is present
4. Verify form submission completes (check Network tab)

Merge Tags Not Replacing

Problem: Emails show `{{event_name}}` instead of actual value

Solutions:

1. Verify merge tags use correct format: `{{variable_name}}`
2. Check `replace_template_variables()` function
3. Verify variable exists in context dictionary
4. Check for typos in variable names

Preview Not Working

Problem: Preview modal blank or shows error

Solutions:

1. Check browser console
2. Verify `exportHtml()` returns valid HTML
3. Check iframe srcdoc support
4. Try different browser

Existing Templates Not Editable

Problem: Old templates created before Unlayer don't load in editor

Solutions:

1. Check if template has `builder_config` field populated
2. If empty, can't re-edit (design not saved as JSON)
3. Options:
 - Recreate template in Unlayer
 - Convert HTML to Unlayer JSON (manual process)
 - Keep using old template as-is (HTML still works for sending)

Security Considerations

XSS Protection

Unlayer HTML Output:

- Unlayer generates clean, safe HTML
- But still sanitize before sending
- Use Django's template escaping for variables

Best Practice:

```
from django.utils.html import escape

# When replacing merge tags
safe_value = escape(str(value))
email_body = email_body.replace(f"{{{key}}}", safe_value)
```

CSRF Protection

Forms include CSRF token:

```
<form method="post">
    {% csrf_token %}
    <!-- hidden fields -->
</form>
```

Input Validation

Backend validation:

```
if not all([name, subject, body_html]):
    messages.error(request, 'Please fill in all required fields.')
    return redirect(...)
```

Database Storage

Storage Size Considerations

Typical Sizes:

- `body_html`: 10-100 KB (depends on email complexity)
- `builder_config`: 5-50 KB (Unlayer JSON)
- Total: ~15-150 KB per template

Database Impact:

- 100 templates \approx 1.5-15 MB

- Minimal impact on performance
- Consider archiving old unused templates

JSON Structure Example

builder_config (Unlayer JSON):

```
{
  "counters": {
    "u_column": 2,
    "u_row": 1,
    "u_content_text": 2,
    "u_content_button": 1
  },
  "body": {
    "rows": [
      {
        "cells": [1],
        "columns": [
          {
            "contents": [
              {
                "type": "text",
                "values": {
                  "text": "<p>Hello {{participant_name}},</p>"
                }
              }
            ]
          }
        ]
      }
    ]
  }
}
```

Best Practices

Email Design

1. Keep It Simple

- Don't overload with too many elements
- Clear hierarchy and spacing
- Focus on one primary CTA

2. Mobile-First

- Unlayer is responsive by default
- Test preview on mobile size

- Use large buttons (min 44x44px)

3. Brand Consistency

- Use consistent colors
- Use your logo
- Maintain typography standards

4. Merge Tags

- Always include personalization
- Test with sample data
- Provide fallback values

Template Organization

1. Naming Convention

- [Type] - [Purpose] - [Version]
- Example: Invitation - Tech Conference - v2

2. Template Types

- Use correct type for filtering
- Makes finding templates easier

3. Active Status

- Deactivate old versions
- Keep only current templates active

4. Documentation

- Add description field
- Note what event/purpose it's for



Performance Optimization

Loading Speed

Unlayer loads from CDN:

```
<script src="https://editor.unlayer.com/embed.js"></script>
```

Tips:

- CDN is fast (global edge network)
- Script cached after first load
- Subsequent loads instant

Editor Initialization

Show loading overlay:

```
document.getElementById('loadingOverlay').style.display = 'flex';

emailEditor.addEventListener('editor:ready', function() {
    document.getElementById('loadingOverlay').style.display = 'none';
});
```

Large Designs

If design gets very large:

- Consider splitting into multiple templates
- Remove unused components
- Optimize images before upload
- Use image CDN for external images

Migration from Old System

For Existing Templates

Old System (Plain Textarea):

```
body = models.TextField() # Plain HTML/text
```

New System (Unlayer):

```
body = models.TextField() # Kept for compatibility
body_html = models.TextField() # Unlayer HTML
builder_config = models.JSONField() # Unlayer design
```

Migration Strategy

Option 1: Keep Old Templates

- Old templates still work for sending
- Can't re-edit in Unlayer
- Gradually replace with new ones

Option 2: Recreate in Unlayer

- Create new template in Unlayer

- Copy text/content from old
- Design visually
- Save and replace old

Option 3: Hybrid

- New templates use Unlayer
- Old templates remain as-is
- Mark old as "legacy"

Code Compatibility

Sending emails works with both:

```
# Works with old and new templates
email_html = template.body_html or template.body
send_mail(html_message=email_html, ...)
```

Support & Resources

Unlayer Documentation

- Official Docs: <https://docs.unlayer.com/>
- API Reference: <https://docs.unlayer.com/docs/javascript-api>
- Examples: <https://github.com/unlayer/react-email-editor>

MakePlus Resources

- [EVENT_REGISTRATION_SYSTEM.md](#) - Full system guide
- [BACKEND_DOCUMENTATION.md](#) - Backend architecture
- Dashboard → Help - In-app documentation

Common Questions

Q: Is Unlayer free?

A: Yes, basic version is free forever. Pro features require subscription.

Q: Do I need an Unlayer account?

A: No, editor works without account. Optional project ID for cloud features.

Q: Can I export templates?

A: Yes, [builder_config](#) JSON can be exported and imported.

Q: Does it work offline?

A: No, Unlayer requires internet connection (loads from CDN).

Q: Can I use custom fonts?

A: With Unlayer project ID and custom configuration, yes.

☒ Checklist for Going Live

- ☐ Test creating new template
- ☐ Test editing existing template
- ☐ Test preview functionality
- ☐ Test settings modal
- ☐ Test merge tags in editor
- ☐ Test saving and reloading design
- ☐ Send test email and verify HTML renders correctly
- ☐ Test on mobile devices
- ☐ Test in multiple email clients (Gmail, Outlook, Apple Mail)
- ☐ Verify merge tags replace correctly
- ☐ Configure Unlayer project ID (optional)
- ☐ Train team on using Unlayer
- ☐ Create template examples/library
- ☐ Document custom workflows

Document End

Unlayer integration successfully brings professional drag-and-drop email design to MakePlus!