

Alex Stryer Diaz

A0707173

4/6/2024

Notas:

Desarrollo de Componentes de Software

Convención de Código:

Para el equipo usar cosas
estándar

- Para variables " _ "
- Clases con uppercase
- nombres simples y coherentes
- Usa PascalCase

Front End:

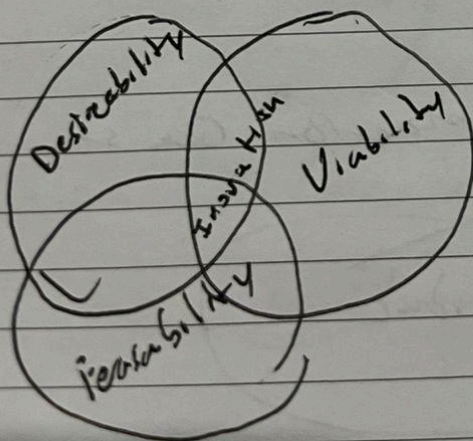
Se utilizan frameworks variados
de estilo junto con Mockups en
Canva.

Metodologías

- SCRUM

Arquitectura:

- Cliente - Servidor
- División de responsabilidades
por modelos



Convención del código:

En general, la mayoría del código que utilizamos está escrito en Django. El código está escrito de esta manera porque ya teníamos algo de experiencia utilizando django por un curso que tomamos juntos. Para nuestra base de datos utilizamos MariaDB. No tenemos mucha experiencia con MariaDB, pero como lo fuimos aprendiendo durante los labs, no se nos hizo muy complicado para implementarlo en nuestro proyecto. El frontend está creado con una mezcla de Tailwind y HTMLX. Personalmente no tuve mucha experiencia previa con HTMLX pero sí con Tailwind y también incluso hice un laboratorio con tailwind.

En nuestro proyecto utilizamos una estructura de templates de la siguiente manera:

- templates/OBD/ para vistas generales
- templates/account/ para autenticación
- partials/ y includes/ para componentes reutilizables

Aparte de esto, encapsulamos nuestras funcionalidades en una app: OBD_users.

Contribuciones:

Para la mayoría de la creación del frontend y backend hasta el momento ha sido liderado por Cesar, quien tomó rol de SCRUM master dentro del equipo. Hasta el momento, hemos implementado mucho del backend, menos ciertas funcionalidades más niñas dentro del proyecto como el mejoramiento de las tareas(Gestión de tareas en un equipo). Pero hemos ya creado:

- El dashboard
- Inventario de campañas
- archivos recientes de campañas
- Sección de campañas
- Creacion y gestion de usuarios, cuentas, roles y vistas

Para la implementación del login y el sistema de autenticación utilizamos Allauth. Allauth nos permitió la creación de esta parte de una forma más eficiente y rápida e incluso pudimos encontrar mucha información útil en el internet sobre como utilizarlo.

Dentro del proyecto, implemente el modelo de usuario:

```
models.py
1  from django.db import models
2  from django.contrib.auth.models import User
3  from django.template.tags import static
4
5  # Create your models here.
6
7  class Profile(models.Model):
8      ROLE_CHOICES = (
9          ('cliente', 'Cliente'),
10         ('vendedor', 'Vendedor'),
11     )
12     role = models.CharField(max_length=10, choices=ROLE_CHOICES)
13     user = models.OneToOneField(User, on_delete=models.CASCADE)
14     image = models.ImageField(upload_to='avatars/', null=True, blank=True)
15     displayname = models.CharField(max_length=20, null=True, blank=True)
16     info = models.TextField(null=True, blank=True)
17
18     def __str__(self):
19         return str(self.user)
20
21     @property
22     def name(self):
23         if self.displayname:
24             name = self.displayname
25         else:
26             name = self.user.username
27         return name
```

Lo más notable y significativo de este código es que se está extendiendo el modelo de usuario en Django mediante una clase Profile, lo cual permite agregar más información personalizada al usuario sin modificar el modelo original de User.

Aparte del modelo de Users, también trabajé con Aksel para la creación del modelo de campañas que es en donde se centra el app. La idea de campañas es que es el lugar donde el cliente verá todo lo que tiene que ver con el desplegado:

```
form_archivolive.html > ...
349 </script>
326 window.addEventListener('pageshow', function(event) {
328     // Página cargada desde cache o botón "atrás"
329     document.getElementById('formSubir').reset();
330
331     // Limpiar preview de imagen
332     const dropZoneContent = document.getElementById('drop-zone-content');
333     const filePreview = document.getElementById('file-preview');
334     if (dropZoneContent && filePreview) {
335         dropZoneContent.classList.remove('hidden');
336         filePreview.classList.add('hidden');
337     }
338 }
339 });
340
341 // Limpiar formulario antes de salir de la página
342 window.addEventListener('beforeunload', function() {
343     const form = document.getElementById('campana-form');
344     if (form) {
345         form.reset();
346     }
347 });
348
349 </script>
350
351 [% endblock %]
```

Aksel y yo también trabajamos en la creación de ciertas partes del css de varias áreas de la aplicación ya que diseñamos muchas vistas de las vistas:

