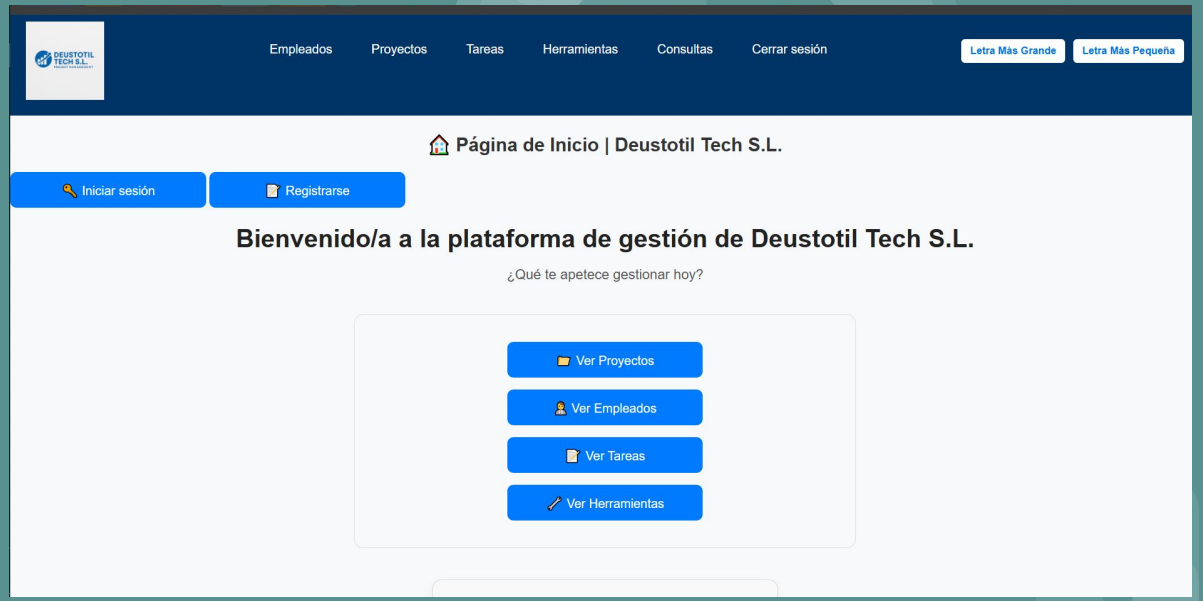
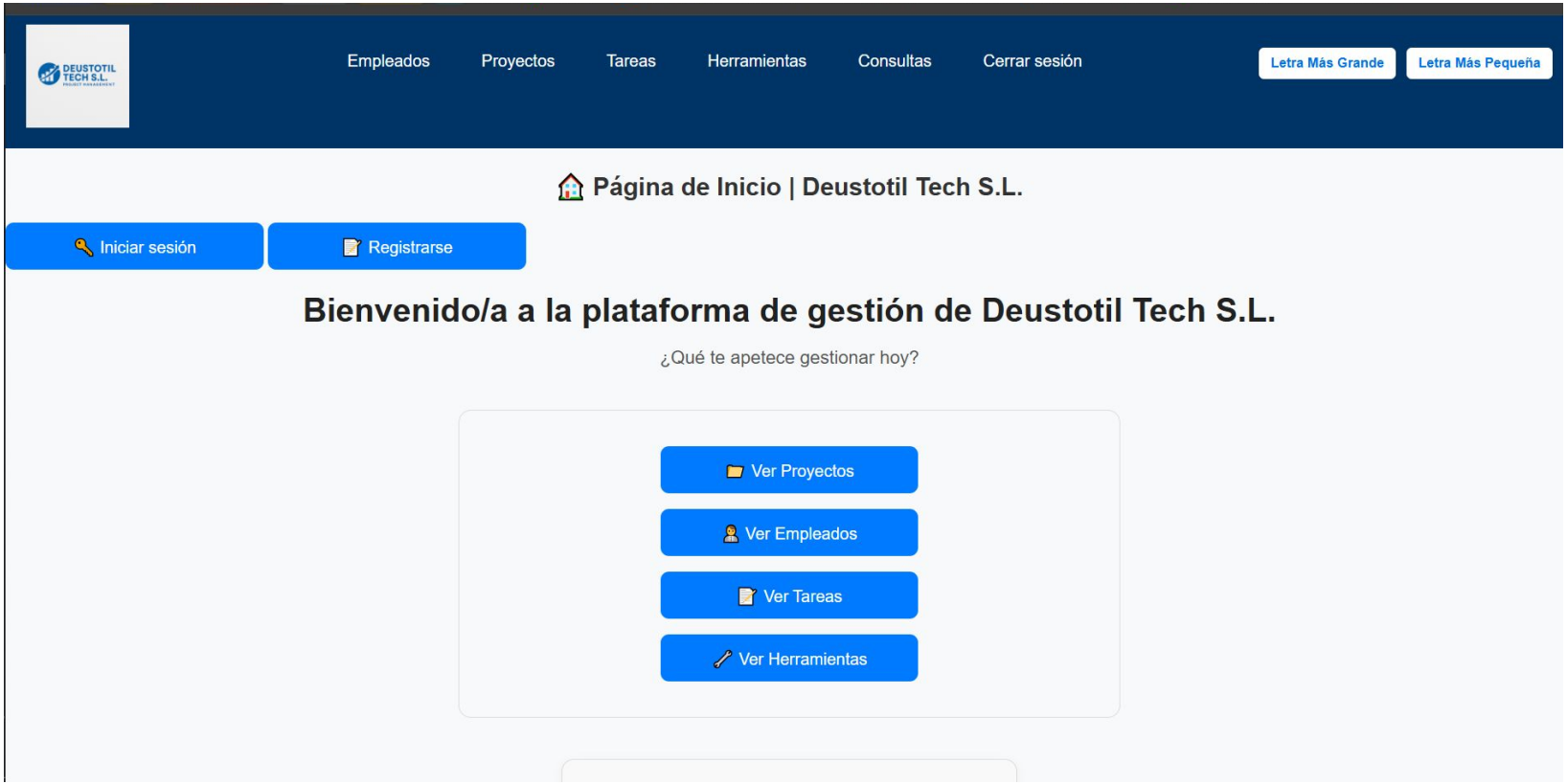
Proyecto Colaborativo

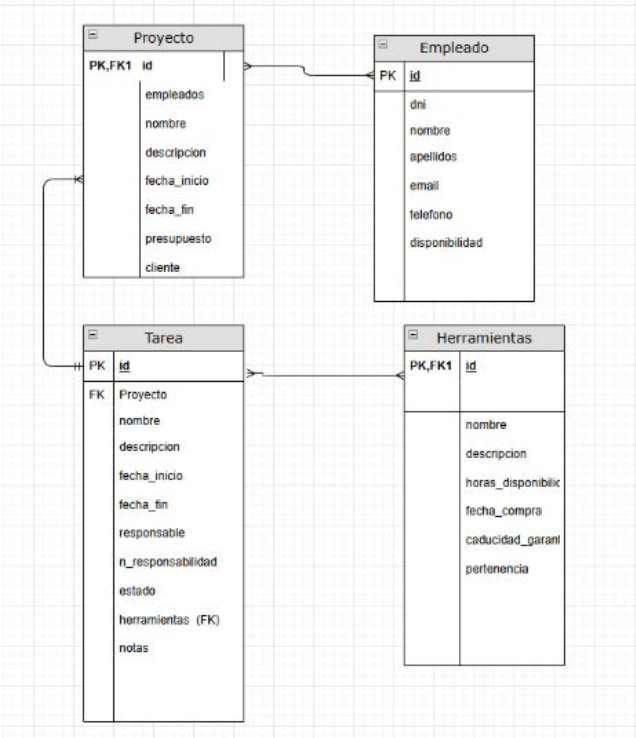


Equipo 3: Jon Tolosa, Daniel Pillado y Gaizka Miranda

1. Presentación y Explicación Breve del Proyecto:



2. Diseño: Funcionalidades Planteadas, Modelo de Datos Diseñado y Descripción de las Interfaces:



3. Planificación y Organización en Equipo:

- GitHub Projects

- Desarrollo Diario

- Comunicación Proactiva

4.- Implementación/Demo:

<code>

4.1 Registro e Inicio de Sesión de Usuarios (Django)

Vista de Registro de Usuario en Django: Validación y Respuesta

```python
def registro_view(request):
    if request.method == 'POST':
        form = RegistroForm(request.POST)
        if form.is_valid():
            user = form.save()
            username = form.cleaned_data.get('username')
            messages.success(request, f'Cuenta creada para {username}')
            return redirect('login')
        else:
            messages.error(request, 'Error al crear la cuenta. Verifica los datos e intenta nuevamente.')
    # si el metodo no es POST se crea una instancia vacia del formulario para que la pagina de registro se muestre
    # correctamente, esto permite que la plantilla registro.html se renderice con un formulario vacio que el usuario
    # pueda rellenar
    else:
        form = RegistroForm()
    return render(request, 'registro.html', {'form': form})

```

4.2 Registro e Inicio de Sesión de Usuarios (Django)

Formulario de registro de usuario

```python
class RegistroForm(UserCreationForm):
    #vamos a hacer el autorrelleno en este campo
    email = forms.EmailField(required=True)
    class Meta:
        fields = ['username', 'email', 'password1']
        model = User # tenemos que usar el model User de Django:

    def save(self, commit=True):
        user = super().save(commit=False)
        user.email = self.cleaned_data['email']
        if commit:
            user.save()
        return user
```

4.3 Registro e Inicio de Sesión de Usuarios (Django)

Plantilla de Registro de Usuario: html

```
<div class="login-container">
    <h2>Crear nueva cuenta</h2>

    <form method="post" class="registration-form">
        {% csrf_token %}
        <!-- enlaces que nos han ayudado -->
        <!-- https://docs.djangoproject.com/en/5.2/topics/forms/#looping-over-the-forms-fields  -->
        <!-- https://simpleisbetterthancomplex.com/article/2017/08/19/how-to-render-django-form-manually.html -->
        {% for field in form %}
        <div class="form-group">
            <label for="{{ field.id_for_label }}">{{ field.label }}</label>
            {{ field }}
            {% if field.help_text %}
            <small class="form-help">{{ field.help_text }}</small>
            {% endif %}
            {% for error in field.errors %}
            <div class="form-error">{{ error }}</div>
            {% endfor %}
        </div>
        {% endfor %}

        <div class="form-actions">
            <button type="submit" class="btn-enviar">Registrarse</button>
        </div>
    </form>

    <div class="texto-cuenta">
        ¿Ya tienes una cuenta? <a href="{% url 'login' %}" class="enlace-inicio">Inicia sesión aquí</a>.
    </div>

    {% if messages %}
    <div class="messages">
        {% for message in messages %}
        <div class="mensaje3">
            {{ message }}
        </div>
        {% endfor %}
    </div>
    {% endif %}
</div>
```

- Renderizado dinámico con bucle for:
  - Si se añaden o quitan campos, el template no necesita modificarse
- Mediante el sistema de mensajes de Django se indica si la cuenta ha sido creada o ha habido un error.
- Se da la opción de iniciar sesión desde la plantilla

4.4 Registro e Inicio de Sesión de Usuarios (Django)

Vista del inicio de sesión

```python
def login_view(request):
    # Si el usuario ya está autenticado, redirigir
    if request.user.is_authenticated:
        return redirect('proyectos')

    if request.method == 'POST':
        username = request.POST.get('username', '').strip()
        password = request.POST.get('password', '').strip()

        # Validación básica
        if not username or not password:
            messages.error(request, 'Todos los campos son obligatorios')
        else:
            user = authenticate(request, username=username, password=password)

            if user is not None:
                if user.is_staff:  # Verifica que el usuario sea admin
                    login(request, user)
                    next_url = request.GET.get('next', 'inicio') #envíamos al usuario a la p
                    return redirect(next_url)
                else:
                    messages.error(request, 'No tienes permisos de administrador.')
            else:
                messages.error(request, 'Nombre de usuario o contraseña incorrectos.')

    # Contexto para el template
    context = {
        'title': 'Inicio de Sesión',
        'messages': messages.get_messages(request)
    }
    return render(request, 'login.html', context)
```

- Redirección de usuario en caso de estar ya autenticado.
- Se obtienen las credenciales desde el “POST”: Se recogen los datos del formulario de login y se eliminan espacios extra.
- Validación de todos los campos completados: evita continuar si faltan datos.
- Se redirige al usuario a la página de inicio con “next”.

4.5 Registro e Inicio de Sesión de Usuarios (Django)

Plantilla de inicio de sesión: html

```
{% block contenido %}
<div class="login-container">
    <h2>Inicio de Sesión</h2>
    <form method="post">
        {% csrf_token %}
        <label for="username">Nombre de usuario:</label>
        <input type="text" id="username" name="username" required><br><br>

        <label for="password">Contraseña:</label>
        <input type="password" id="password" name="password" required><br><br>

        <div class="form-buttons">
            <button type="submit" class="btn-enviar">Iniciar Sesión</button>
            <a href="{% url 'registro' %}" class="link-registro">Registro</a>
        </div>
    </form>

    {% if messages %}
        <div class="messages">
            {% for message in messages %}
                <div class="mensaje2">
                    {{ message }}
                </div>
            {% endfor %}
        </div>
    {% endif %}
</div>
{% endblock %}
```

- Campos obligatorios para username y password
- Uso de {% csrf_token %} para proteger contra ataques CSRF.
- Botón para enviar el formulario y enlace directo a la página de registro para usuarios nuevos.

4.6 Registro e Inicio de Sesión de Usuarios (Django)

Formulario de envío de email: html

```python
# formulario para el envio de emails
class EmailForm(forms.Form):
    mensaje = forms.CharField(
        label='Mensaje',
        widget=forms.Textarea(attrs={'rows': 3, 'cols': 100}),

        required=True
    )

    def clean_mensaje(self):
        mensaje = self.cleaned_data.get('mensaje')
        palabras_prohibidas = ["palabra1", "palabra2", "palabra3"]
        for palabra in palabras_prohibidas:
            mensaje = mensaje.replace(palabra, '*' * len(palabra))
        return mensaje

```

- Se define la estructura del mensaje a enviar:
  - Textarea de 3 líneas y 100 columnas de caracteres.
- Validación personalizada del mensaje mediante “clean_mensaje”.
  - Palabras prohibidas, límite de contenido…
- Se evita rechazar el formulario por lenguaje inapropiado, ya que no se lanza un error, se corrige automáticamente.

4.7 Envío de Emails (Django)

Vista de envío de emails: views.py

```python
# vista de envio de emails
def enviar_email(request):
    if request.method == 'POST':
        form = EmailForm(request.POST)
        if form.is_valid():
            mensaje = form.cleaned_data['mensaje']
            try:
                send_mail(
                    'Mensaje de Soporte',
                    mensaje,
                    request.user.email,
                    ['deustotiltech80@gmail.com']
                )
                messages.success(request, 'Mensaje enviado con éxito.')
                return redirect('consulta_email')
            except Exception as e:
                messages.error(request, f'Error al enviar el mensaje: {e}')
    else:
        form = EmailForm()
    return render(request, 'consulta.html', {'form': form})
```

- Validación del formulario para asegurar que el mensaje está correcto antes de enviar.
- Uso de mensajes (messages.success y messages.error) para retroalimentar al usuario.
- Redirección tras el envío exitoso para mejorar la experiencia de usuario.

4.8 Evento para Cambiar Tamaño de Elementos Tipo Text

función modificarTamanoTexto.js

```js
document.addEventListener("DOMContentLoaded", function () {
    const aumentarTexto = document.getElementById("aumentarTexto");
    const disminuirTexto = document.getElementById("disminuirTexto");
    let fontSize = 16;
    function cambiarTamanoTexto(tamano) {
        const elementos = document.querySelectorAll("h1, h2, h3, h4, h5, h6, p, li, td, th");
        elementos.forEach(el => {
            el.style.fontSize = tamano + "px";
        });
    }
    if (aumentarTexto) {
        aumentarTexto.addEventListener("click", function (e) {
            e.preventDefault();
            fontSize += 1;
            cambiarTamanoTexto(fontSize);
        });
    }
    if (disminuirTexto) {
        disminuirTexto.addEventListener("click", function (e) {
            e.preventDefault();
            fontSize -= 1;
            cambiarTamanoTexto(fontSize);
        });
    }
});
```

Espera a que todo el HTML esté cargado antes de ejecutar el script.

Selecciona los botones por su ID. Uno aumenta el tamaño del texto, el otro lo disminuye.

Esta función aplica el nuevo tamaño a todos los títulos, párrafos, listas y celdas de tablas de la página.

Al hacer clic, se modifica el valor de fontSize y se actualiza el estilo de los elementos. Se evita también que el botón recargue la página (e.preventDefault()).

4.9 Evento para Mostrar Alerta Cuando el Usuario Realiza una Acción Determinada

función alertaCerrarSesion.js

static > js > JS alertaCerrarSesion.js > ...

```
1   // Captura de evento en DOM y mostar alerta si el usuario pulsa "cerrar sesion"
2   // Se trata de una implementacion que trata de asegurarse mediante una doble confirmacion
3   // de que el usuario, efectivamente quiere cerrar sesion
4
5   document.addEventListener("DOMContentLoaded", function () {
6       const cerrarSesion = document.getElementById("cerrarSesion");
7
8       if (cerrarSesion) {
9           cerrarSesion.addEventListener("click", function (e) {
10              const confirmado = confirm("¿Confirmar cierre de sesión?");
11              if (!confirmado) {
12                  e.preventDefault(); // Cancela la redirección si el usuario pulsa "Cancelar"
13              }
14          });
15      }
16  });
```

- Espera a que todo el HTML esté cargado antes de ejecutar el script.

- Selecciona el elemento con ID cerrarSesion, que se trata del enlace de "Cerrar sesión" de la sección Header.

- Añade un listener para capturar el clic del usuario.

- Muestra una ventana emergente con los botones “Aceptar” y “Cancelar”.

- Si el usuario pulsa “Cancelar”, se evita la acción de cierre de sesión.

4.10 Validar campos de Empleados

```
document.addEventListener('DOMContentLoaded', () => {
    const DNI = document.getElementById('id_dni');
    const email = document.getElementById('id_email');
    const telefono = document.getElementById('id_telefono');
    const mensajeDiv = document.getElementById('mensaje-validacion');
    const formulario = document.querySelector('form'); // Selección genérica del form

    const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;

    function showMessage(message) {
        mensajeDiv.innerText = message;
        mensajeDiv.style.display = 'block';
    }

    function clearMessage() {
        mensajeDiv.innerText = '';
        mensajeDiv.style.display = 'none';
    }

    DNI.addEventListener('blur', () => {
        if (DNI.value.length !== 9 && DNI.value.length !== 0) {
            showMessage('La longitud del DNI debe ser igual a 9');
        } else {
            clearMessage();
        }
    });

    telefono.addEventListener('blur', () => {
        if (telefono.value.length !== 9 && telefono.value.length !== 0) {
            showMessage('La longitud del teléfono debe ser igual a 9');
        } else {
            clearMessage();
        }
    });

    email.addEventListener('blur', () => {
        if (email.value.length > 0 && !emailRegex.test(email.value)) {
            showMessage('El formato del correo electrónico no es válido.');
        } else {
            clearMessage();
        }
    });
```

Esperar a que el DOM esté completamente cargado

Selección de elementos del DOM

Definición de la expresión regular para validar el correo electrónico

Funciones para mostrar y limpiar mensajes de validación

Validación del campo DNI al perder el foco

Validación del campo teléfono al perder el foco

Validación del campo email al perder el foco

4.10.2 Validar campos de Empleados

Validación al enviar el formulario

```
formulario.addEventListener('submit', (event) => {
    if (DNI.value.length !== 9) {
        showMessage('La longitud del DNI debe ser igual a 9');
        event.preventDefault();
    } else if (telefono.value.length !== 9) {
        showMessage('La longitud del teléfono debe ser igual a 9');
        event.preventDefault();
    } else if (!emailRegex.test(email.value)) {
        showMessage('El formato del correo electrónico no es válido.');
        event.preventDefault();
    } else {
        clearMessage();
    }
```

4.11 Fetch API

```
const API_URL_DETALLE_EMPLEADO = "http://127.0.0.1:8000/api/empleado/";

// Espera a que el DOM cargue completamente
document.addEventListener("DOMContentLoaded", function () {
    document.querySelectorAll(".detalle-empleado").forEach(item => {
        item.addEventListener("click", function (e) {
            e.preventDefault(); // evita que el enlace recargue la página

            const li = this.closest(".item-lista");

            const nombre = li.dataset.nombre;
            const dni = li.dataset.dni || "No especificado";
            const apellidos = li.dataset.apellidos || "No especificado";
            const disponibilidad = li.dataset.disponibilidad || "No especificada";
            const email = li.dataset.email || "No especificado";
            const telefono = li.dataset.telefono || "No especificado";

            // Actualiza el contenido del contenedor de detalles
            document.getElementById("detalle-nombre").textContent = nombre;
            document.getElementById("detalle-dni").textContent = dni;
            document.getElementById("detalle-apellidos").textContent = apellidos;
            document.getElementById("detalle-email").textContent = email;
            document.getElementById("detalle-disponibilidad").textContent = disponibilidad;
            document.getElementById("detalle-telefono").textContent = telefono;

            // Muestra el contenedor si está oculto
            document.getElementById("detalles-empleado").style.display = "block";
        });
    });
});
```

Definición de la URL base de la API

Esperar a que el DOM esté completamente cargado

Añadir eventos de click a elementos con la clase .detalle-empleado

Manejo del evento de clic para evitar el comportamiento predeterminado

Extracción de datos del elemento utilizando atributos data-*

4.11.2 Fetch API

```
function cargarDetalleEmpleado(id) {
    const apiUrl = `${API_URL_DETALLE_EMPLEADO}${id}/`;

    fetch(apiUrl)
        .then(response => {
            if (!response.ok) {
                throw new Error("Error al cargar los datos del empleado.");
            }
            return response.json();
        })
        .then(data => {
            mostrarDetalleEnDOM(data);
        })
        .catch(error => {
            console.error(error);
            document.getElementById("detalles-empleado").innerHTML = `
                <div class="error">⚠ No se pudo cargar la información del empleado.</div>`;
        });
}

function mostrarDetalleEnDOM(empleado) {
    const contenedor = document.getElementById("detalles-empleado");

    contenedor.innerHTML = `
        <div class="detalle-box">
            <h3>Detalles del empleado</h3>
            <p><strong>Nombre:</strong> ${empleado.nombre}</p>
            <p><strong>Apellido:</strong> ${empleado.apellidos}</p>
            <p><strong>Email:</strong> ${empleado.email}</p>
            <p><strong>Disponibilidad:</strong> ${empleado.disponibilidad}</p>
        </div>
    `;

    contenedor.style.display = "block";
```

Actualización del contenido del DOM con los datos del empleado

Mostrar el contenedor de detalles del empleado

Función para cargar detalles del empleado desde la API utilizando fetch

Función para mostrar los detalles del empleado en el DOM

5. Conclusiones: