

Universidad de Sevilla
Escuela Técnica Superior de Ingeniería Informática
decide-single-maroto



Grado en Ingeniería Informática – Ingeniería del Software
Evolución y gestión de la configuración

Curso 2023 – 2024

Miembro	Implicación
Bustamante Lucena, Eduardo	10
Carrera Bernal, Álvaro	10
Martínez Cano, Juan	10
Rodríguez Corder, Javier	10
Pacheco Rodrigues, Guillermo Alonso	10

Documento del proyecto.	4
Indicadores del proyecto.	4
Integración con otros equipos.	6
Resumen ejecutivo.	7
Descripción del sistema.	8
Visión global del proceso de desarrollo.	43
Entorno de desarrollo.	47
Ejercicio de propuesta de cambio.	50
Conclusiones y trabajo futuro.	53

Documento del proyecto.

Indicadores del proyecto.

Los indicadores del proyecto se especificarán a través de la siguiente tabla.

Miembro del equipo	Horas	Commits	LoC	Test	Issues	Incremento
Bustamante Lucena, Eduardo	35	26	820	16	18	<ul style="list-style-type: none">- Postprocesado de tipo D'hondt.- FrontEnd módulo Voting (creación, edición de la votación, operaciones start, stop y tally)- FrontEnd Auth
Carrera Bernal, Álvaro	33	12	405	7	6	<ul style="list-style-type: none">- Traducción.- Visualización de resultados mediante gráficas.- Información en tiempo real.- Integrar estos cambios en el FrontEnd.
Martínez Cano, Juan	35	36	1090	12	6	<ul style="list-style-type: none">- Creación de votaciones tipo sí/no- FrontEnd Creación de preguntas.- FrontEnd Creación de censo.
Rodríguez Cordero, Javier	35	21	659	8	7	<ul style="list-style-type: none">- OAuth Google.- Registro de usuarios.- Funcionalidad de Logout de usuarios con sesión activa.- Integrar estos cambios en el FrontEnd.
Pacheco Rodrigues, Guillermo Alonso	35	30	1101	16	6	<ul style="list-style-type: none">- Importación/Exportación de censos.- FrontEnd Listado de Censos

Leyenda utilizada en la tabla:

- Horas: número de horas empleadas en el proyecto.
- Commits: solo contar los commits hechos por miembro del equipo, no los commits previos.
- LoC(Líneas de código): solo contar las líneas producidas por el equipo y no las que ya existían o las que se producen al incluir código de terceros.
- Test: solo contar los test realizados por el equipo.
- Issues: solo contar las issues gestionadas dentro del proyecto y que hayan sido gestionadas por el equipo.
- Incremento: principal incremento funcional del que se ha hecho cargo el miembro del proyecto.

Integración con otros equipos.

En nuestro caso al tratarse de un proyecto single este apartado no aplica.

Resumen ejecutivo.

Inicialmente el equipo de trabajo acordó en una reunión de equipo que funcionalidad iba a realizar cada uno de los desarrolladores.

En esta reunión se obtuvieron los siguientes puntos.

- El postprocesado utilizando la ley de D'hondt correría a cargo de Eduardo Bustamante Lucena.
- La creación de preguntas del tipo Sí/No, correría a cargo de Juan Martínez Cano.
- El registro de usuarios y la implementación de la autenticación con Google OAuth, sería implementado por Javier Rodríguez Cordero.
- Traducción usando la API de Google Translate, el encargado de la implementación fue Álvaro Carrera Bernal
- Las gráficas de visualización de resultados de votaciones y la información en tiempo real de las mismas correrían a cargo de Álvaro Carrera Bernal.
- La funcionalidad de importación y exportación de censos sería realizada por Guillermo Alonso Pacheco Rodrigues.

Una vez se realizó esta primera reunión el esfuerzo del equipo se centró en completar las funcionalidades acordadas en esta reunión. Ahora que tenemos una vista global del proyecto podemos decir que no ha sido la organización más correcta puesto que cada desarrollador se especializó exclusivamente en el desarrollo de su funcionalidad perdiendo la visión total del proyecto.

Esto mismo fue la conclusión que obtuvimos en el M2 junto con nuestra profesora encargada del seguimiento Belén Ramos Gutiérrez, la cual denominó nuestro proyecto como “frankenstein” ya que los cambios realizados eran muy dispares, modificando una gran cantidad de módulos pero teniendo una conexión muy pobre entre los cambios.

La solución a la cual llegó el equipo de trabajo junto a Belén fue la creación de un FrontEnd con el objetivo de obtener un producto más completo y que todas los incrementos realizados se integrarán de una manera más correcta. Aquí es donde ha centrado la mayoría del esfuerzo el equipo de trabajo puesto que ahora teníamos un reto mayor ya que debíamos integrar toda la funcionalidad realizada hasta ahora.

A día de hoy, antes de la realización del M3 podemos decir de manera satisfactoria que el equipo de trabajo ha conseguido desarrollar el FrontEnd de manera completa y a nuestro criterio tener un producto de calidad tal y como se nos pidió en el M2.

Descripción del sistema.

A continuación se detalla de manera individual el incremento de cada desarrollador.

Ley D'hondt

El desarrollo de la ley D'hondt ha supuesto que el desarrollador Eduardo Bustamante realice cambios en los siguientes módulos:

- **Módulo Voting.** Los cambios en el módulo voting realizados han sido los siguientes:

El primer cambio realizado fue en el models.py, incluyendo una primera opción para permitir la selección entre el postprocesado mediante “Identity” y mediante “D'hondt”, además se incluyó un atributo “seats” que permitirá agregar el número de escaños a contemplar en la votación.

```
class Voting(models.Model):
    name = models.CharField(max_length=200)
    desc = models.TextField(blank=True, null=True)
    question = models.ForeignKey(Question, related_name='voting', on_delete=models.CASCADE)

    start_date = models.DateTimeField(blank=True, null=True)
    end_date = models.DateTimeField(blank=True, null=True)

    pub_key = models.OneToOneField(Key, related_name='voting', blank=True, null=True, on_delete=models.SET_NULL)
    auths = models.ManyToManyField(Auth, related_name='votings')

    model = models.CharField(max_length=8, choices=[('IDENTITY','Identity'), ('DHONDT', "D'hondt")], default= 'IDENTITY')
    seats = models.PositiveIntegerField(blank=True, null=True)

    tally = JSONField(blank=True, null=True)
    postproc = JSONField(blank=True, null=True)
```

- **Módulo Postproc.** Los cambios en el módulo postproc han permitido desarrollar lo que es la propia función de d'hondt.

En primer lugar se desarrolló el propio código de la ley de D'hondt.

```
def dhondt(self, options):
    nSeats = int(self.request.data.get('seats'))
    seats={}

    results={}
    print(options)
    for opt in options:
        if opt['option'] not in results:
            results[opt['option']] = opt['votes']
        else:
            results[opt['option']].append(opt['votes'])

    seats={}
    t_votes = results.copy()

    for key in results: seats[key]=0
    while sum(seats.values()) < nSeats:
        max_v= max(t_votes.values())
        next_seat=list(t_votes.keys())[list(t_votes.values()).index(max_v)]
        if next_seat in seats:
            seats[next_seat]+=1
        else:
            seats[next_seat]=1

    t_votes[next_seat]=results[next_seat]/(seats[next_seat]+1)
    return Response(seats)
```

Seguidamente se agregó la opción de que se pudiera realizar este cálculo si el tipo seleccionado era igual a “DHONDT”

```
def post(self, request):
    """
    * type: IDENTITY | EQUALITY | WEIGHT
    * options: [
        {
            option: str,
            number: int,
            votes: int,
            ...extraparams
        }
    ]
    """

    t = request.data.get('type', 'IDENTITY')
    opts = request.data.get('options', [])

    if t == 'IDENTITY':
        return self.identity(opts)
    if t == 'DHONDT':
        return self.dhondt(opts)

    return Response({})
```

Como podemos ver el desarrollo de esta funcionalidad ha requerido que los módulos de voting y postproc se integren de manera correcta para el funcionamiento de este nuevo sistema de postprocesado.

Creación de preguntas tipo sí/no

La creación de preguntas de tipo sí/no ha requerido que el desarrollador Juan Martínez Cano realice cambios en los siguientes módulos:

- **Módulo voting:** Se añadió un nuevo atributo al modelo Question, este atributo en cuestión fue el atributo category, que indicaba el tipo de pregunta a elegir, con las opciones entre Option y Yes/No.

```
class Question(models.Model):
    desc = models.TextField()

    cattegory = models.CharField(max_length=8, choices=[('YES/NO', 'yes/no'), ('OPTIONS', 'options')], default="")
    def __str__(self):
        return self.desc
```

Además, se creó una nueva función para la creación automática de las opciones sí y no. En ella se comprueba si la pregunta es de tipo sí/no. En caso de que sea de este tipo borra las posibles opciones de respuesta y crea las opciones de sí y no.

```
def create_options_yes_no(self):

    if self.question.cattegory == "YES/NO":
        if self.question.options.all().count() != 0:
            for opt in self.question.options.all():
                opt.delete()

        option1 = QuestionOption(question = self.question, option = "Yes", number = 1)
        option1.save()
        option2 = QuestionOption(question = self.question, option = "No", number = 2)
        option2.save()
```

Frontend - Authentication - Registro de usuarios

La implementación de la funcionalidad referente al registro de usuarios del sistema ha sido llevada a cabo por el miembro del grupo Javier Rodríguez Cordero, quien ha realizado los cambios que se detallan a continuación.

El módulo donde se ha llevado a cabo la funcionalidad ha sido el de **authentication**. Los cambios realizados son:

1. Se añadió al fichero urls.py la dirección donde encontrar la funcionalidad:



```
1 urlpatterns = [
2     path('login/', obtain_auth_token),
3     path('logout/', LogoutView.as_view(), name='logout'),
4     path('getuser/', GetUserView.as_view()),
5     path('register/', RegisterView.as_view(), name='register'),
6     path('signup/', RegisterUserView.as_view(), name='register_user'),
7     path('', include('allauth.urls')),
8 ]
```

2. A continuación, se desarrolló la función en el fichero views.py que daría respuesta al get y al post, así como las restricciones y campos del formulario:



```
1 class RegisterUserView(APIView):
2     def get(self, request):
3         return render(request, 'register.html')
4
5     def post(self, request):
6         email = request.data.get('email', '')
7
8         username = request.data.get('username', '')
9         pwd = request.data.get('password', '')
10        pwd_confirm = request.data.get('password_confirm', '')
11        email = request.data.get('email', '')
12
13        User = get_user_model()
14        errors = []
15
16        if not username or not pwd or not pwd_confirm or not email:
17            errors.append('All fields are required.')
18
19        if User.objects.filter(username=username).exists():
20            errors.append('The username is already in use.')
21
22        if pwd != pwd_confirm:
23            errors.append('The passwords do not match.')
24
25        if len(pwd) < 8:
26            errors.append('The password must have 8 characters at least.')
27
28        if errors:
29            return render(request, 'register.html', {'errors': errors})
30
31        try:
32            user = User(username=username)
33            user.set_password(pwd)
34            user.save()
35            message = 'Account successfully created.'
36        except IntegrityError:
37            return Response({}, status=HTTP_400_BAD_REQUEST)
38        return render(request, 'login.html', {'message': message})
```

3. Para finalizar se desarrolló el fichero hmtl:

```

● ● ●
1  <body>
2    <div class="login-container">
3      <h1>Register</h1>
4
5      {% if errors %}
6        <ul>
7          {% for error in errors %}
8            <li>{{ error }}</li>
9          {% endfor %}
10       </ul>
11     {% endif %}
12
13   <form action="{% url 'register_user' %}" method="post">
14     {% csrf_token %}
15     <label for="email">Email:</label>
16     <input type="email" name="email" id="email" placeholder="Enter your email">
17
18     <label for="username">Username:</label>
19     <input type="text" name="username" id="username" placeholder="Choose a username">
20
21     <label for="password">Password:</label>
22     <input type="password" name="password" id="password" placeholder="Enter your password (8 characters at least)">
23
24     <label for="password_confirm">Repeat Password:</label>
25     <input type="password" name="password_confirm" id="password_confirm" placeholder="Repeat your password">
26     <input type="submit" value="Register">
27   </form>
28   <p>Already have an account? <a href="{% url 'signin' %}">Log in</a></p>
29 </div>
30 </body>

```

A continuación, se muestran las vistas de la funcionalidad:

1. En la pantalla de Login se muestra un enlace para que en caso de que el usuario quiera registrarse, pueda hacerlo mediante este:

The screenshot shows a login form with the following elements:

- A text input field labeled "Username".
- A text input field labeled "Password".
- A large green rectangular button labeled "Login".
- A blue rectangular button labeled "Login with Google".

Don't have an account? [Register](#)

2. Tras acceder al enlace, el usuario encuentra un formulario con los distintos campos necesarios para la creación de la cuenta:

The screenshot shows a registration form with the following elements:

- A text input field labeled "Email:" with placeholder "Enter your email".
- A text input field labeled "Username:" with placeholder "Choose a username".
- A text input field labeled "Password:" with placeholder "Enter your password (8 characters at least)".
- A text input field labeled "Repeat Password:" with placeholder "Repeat your password".
- A large green rectangular button labeled "Register".

Already have an account? [Log in](#)

3. Una vez llenado los campos de forma correcta, el usuario es redirigido a la página de Login, donde podrá iniciar sesión con la cuenta creada:

Register

Email:

Username:

Password:

Repeat Password:

Already have an account? [Log in](#)

Oauth de Google

La implementación de la funcionalidad referente a la autenticación de usuarios vía “Google OAuth” ha sido llevada a cabo por el miembro del grupo Javier Rodríguez Cordero, quien ha realizado los cambios que se detallan a continuación.

Antes de toda implementación, se comenzó por la configuración de la API en [“https://console.cloud.google.com/apis/dashboard”](https://console.cloud.google.com/apis/dashboard), donde se siguieron los siguientes pasos:

1. Creación del proyecto bajo el nombre “decide-single-maroto”.
2. Creación de una credencial para el uso de OAuth 2.0.
3. En ella se agregaron los orígenes autorizados, las URIs de redireccionamiento autorizados, y se acceden a ID y el secreto del cliente.

Additional information

ID de cliente	558970792437-0rlcb3qqajt40khtrd2goilcja5s6jh5.apps.googleusercontent.com
Fecha de creación	12 de noviembre de 2023, 12:07:37 GMT+1

Secretos del cliente

Si estás en proceso de cambiar los secretos del cliente, puedes rotarlos de forma manual sin tiempo de inactividad. [Más información](#)

Secreto del cliente	GOCSNX-We8GOqqE33w7e6mGWSVyb2Uihh4X	 
Fecha de creación	12 de noviembre de 2023, 12:07:37 GMT+1	
Estado	 Habilitado	

[+ ADD SECRET](#)

Tras la configuración de la API, se realizaron los cambios en la configuración del proyecto (settings.py):

1. Adición de allauth a las “Installed Apps”:

```
'allauth',
'allauth.account',
'allauth.socialaccount',
'allauth.socialaccount.providers.google',
```

2. Instanciación de la id del site:

```
SITE_ID = 3
```

A continuación, se realizaron los cambios en el módulo **authentication** para agregar la funcionalidad a la vista de inicio de sesión.

Se realizó un cambio en la urls.py de decide para añadir la dirección del signin:

```
● ● ●
1 urlpatterns = [
2     path('admin/', admin.site.urls),
3     path('doc/', schema_view),
4     path('gateway/', include('gateway.urls')),
5     path('', SigninView.as_view(), name='signin')
6 ]
7
```

Además se agregó una línea a la urls.py del authentication (la línea 7 en la imagen), necesaria para el correcto funcionamiento de la funcionalidad:



```
1 urlpatterns = [
2     path('login/', obtain_auth_token),
3     path('logout/', LogoutView.as_view(), name='logout'),
4     path('getuser/', GetUserView.as_view()),
5     path('register/', RegisterView.as_view(), name='register'),
6     path('signup/', RegisterUserView.as_view(), name='register_user'),
7     path('', include('allauth.urls')),
8 ]
```

Se modificó el views.py:



```
1 class SigninView(TemplateView):
2     def post(self, request):
3         form_class = LoginForm(request.POST)
4
5         if form_class.is_valid():
6             username = form_class.cleaned_data.get('username')
7             password = form_class.cleaned_data.get('password')
8
9             user = authenticate(request, username=username, password=password)
10            if user is not None:
11                login(request, user)
12                return redirect("menuSignin")
13            else:
14                error_message="Usuario y/o contraseña incorrecto/a"
15            else:
16                error_message = "Error al cargar el formulario"
17
18            return render(request, 'login.html', {'form': form_class, 'msg': error_message})
19
20    def get(self, request):
21        form_class = LoginForm(None)
22        if request.user.is_authenticated:
23            return render(request, 'base.html')
24        return render(request, 'login.html', {'form': form_class, 'msg': None})
```

Tras esto, toda la funcionalidad referente a la autenticación con Google OAuth estaría realizada. A continuación se verá cómo se implementó esta funcionalidad en el frontend de la aplicación.

Frontend - Authentication - OAuth de Google

Debido a que toda la funcionalidad la proporciona la propia API de Google, lo que se debía realizar es la integración en la aplicación. Por ello se creó la vista de inicio de sesión donde los usuarios tendrán tres opciones: iniciar sesión con una cuenta existente, registrar una nueva cuenta o iniciar sesión con la autenticación de Google OAuth:

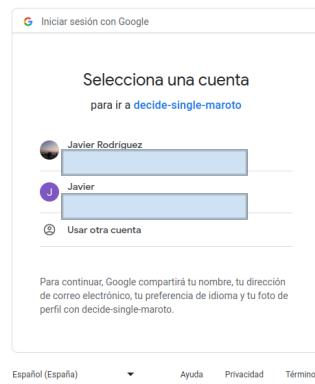
```
● ● ●
1 <body>
2   <div class="login-container">
3     <h1>Login</h1>
4     <form action="{% url 'signin' %}" method="post">
5       {% csrf_token %}
6       <input type="text" name="username" placeholder="Username">
7       <input type="password" name="password" placeholder="Password">
8       {% if msg %}
9         <p style="color: red;">{{ msg }}</p>
10        {% endif %}
11        <input type="submit" value="Login">
12    </form>
13    <div class="google-login">
14      <button class="google-button">
15        <a href="{% provider_login_url 'google'%}?next=http://127.0.0.1:8000/base/">
16          Login with Google
17        </a>
18      </button>
19    </div>
20    <p>Don't have an account? <a href="{% url 'register_user' %}">Register</a></p>
21  </div>
22 </body>
```

Se muestran a continuación las vistas pertinentes:

- Pantalla de Login, donde se accede a la autenticación a través del botón de “Login with Google”:



- Autenticación con OAuth



Frontend - Authentication - Log Out

Tras realizar las funcionalidades referentes al inicio de sesión, autenticación con Google OAuth y registro de usuarios, se decidió que era necesario desarrollar la funcionalidad que permitiese al usuario hacer logout para así, ofrecer un servicio completo de sesión a estos.

Para ello, se hicieron los cambios necesarios en el módulo **authentication**:

1. Se añadió en el fichero urls.py la dirección para encontrar la funcionalidad (línea 3):



```
1 urlpatterns = [
2     path('login/', obtain_auth_token),
3     path('logout/', LogoutView.as_view(), name='logout'),
4     path('getuser/', GetUserView.as_view()),
5     path('register/', RegisterView.as_view(), name='register'),
6     path('signup/', RegisterUserView.as_view(), name='register_user'),
7     path('', include('allauth.urls')),
8 ]
```

2. Tras esto, se desarrolló la funcionalidad pertinente en el views.py:



```
1 class LogoutView(View):
2     def get(self, request):
3         if request.user.is_authenticated:
4             logout(request)
5             return redirect('/')
6
7     def post(self, request):
8         if request.user.is_authenticated:
9             logout(request)
10            return redirect('/')
```

3. Para finalizar, se implementó la representación en el html, en concreto en el header, para que en cualquier momento se pueda hacer logout:



```
1 <li class="nav-item active">
2     <a class="nav-link" href="/authentication/logout">Log out <span class="sr-only">(current)</span></a>
3 </li>
```

A continuación se muestra la funcionalidad en una vista, donde si el usuario pulsa en el botón de “Log out” del header, se cerrará su sesión:

Decide Votación · Preguntas · Censo · Log out · admin

¡Bienvenido admin!

Correo electrónico: admin@us.es

Google Seleccionar idioma

Traducción mediante la API de Google Translate

La funcionalidad de traducción de la página mediante la API de Google ha sido implementada en el sistema. Los detalles de los cambios realizados están detallados a continuación y fueron llevados a cabo por el miembro del equipo Álvaro Carrera Bernal.

El módulo donde se ha llevado a cabo la funcionalidad ha sido el de **base**. Los cambios realizados son los siguientes:

- Se desarrolló el fichero html, en el que por un lado tenemos la parte de la cabecera donde lo más destacado es el script incluido. El cual es un script externo de Google Translate que carga la funcionalidad de traducción.

```
● ● ●

<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <title>{% block title %}Decide!{% endblock %}</title>

    <script type="text/javascript"
src="https://translate.google.com/translate_a/element.js?cb=googleTranslateElementInit">
</script>

<style>
    #google_translate_element {
        text-align: center;
        margin-top: 50px;
    }

    #language-selector {
        display: block;
        margin: 0 auto;
        text-align: center;
    }
</style>
    {% block extrahead %}{% endblock %}
</head>
    return go(f, seed, [])
}
```

Por otro lado tenemos el cuerpo de la página web, donde podemos ver cómo se inserta el widget de traducción de Google en la página y, además, un script que incluye dos funciones. La primera inicializa el widget mencionado anteriormente cuando la página se carga, a la vez que define el idioma y el diseño. La segunda función se usa para cambiar el idioma al que deseemos.

```


    <body>
        {% block content %}
        {% endblock %}
        <div id="google_translate_element" class="google"></div>

        <script type="text/javascript">
            function googleTranslateElementInit() {
                new google.translate.TranslateElement({
                    pageLanguage: 'es',
                    layout: google.translate.TranslateElement.InlineLayout.SIMPLE
                }, 'google_translate_element');
            }

            function changeLanguage(languageCode) {
                const selectElement = document.getElementById("google_translate_element");
                const event = new Event('change');
                selectElement.value = languageCode;
                selectElement.dispatchEvent(event);
            }
        </script>
        {% block extrabody %}
        {% endblock %}
    </body>


```

Tras realizar esto, la traducción de la interfaz de la web estaría realizada.

The screenshot shows a top navigation bar with tabs: 'Decide', 'Votación', 'Preguntas', 'Censo', 'Log out', and 'admin'. Below this, a welcome message '¡Bienvenido admin!' is displayed, followed by the email 'Correo electrónico: admin@us.es'. A red box highlights a dropdown menu labeled 'Seleccionar idioma' with a small Google logo.

Visualización de resultados mediante gráficas e información en tiempo real

La funcionalidad de visualización de resultados mediante gráficas e información en tiempo real ha sido implementada en el sistema. Los detalles de los cambios realizados están detallados a continuación y fueron llevados a cabo por el miembro del equipo Álvaro Carrera Bernal.

El módulo donde se ha llevado a cabo la funcionalidad ha sido el de visualizer. Los cambios realizados son los siguientes:

1. Se desarrolló la función en el fichero views.py para recopilar información específica sobre una votación (si está disponible), así como calcular la participación.

```
● ● ●

class VisualizerView(TemplateView):
    template_name = 'visualizer/visualizer.html'

    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        voting_id = kwargs.get('voting_id', 0)

        try:
            voting = mods.get('voting', params={'id': voting_id})
            context['voting'] = json.dumps(voting[0])
            num_census = 0
            num_votos = 0
            participation = "-"

            if voting[0].get('start_date'):
                num_census = Census.objects.filter(voting_id=voting_id).count()
                num_votos = Vote.objects.filter(voting_id=voting_id).count()

            if num_census != 0:
                participation = str(round((num_votos*100)/num_census, 2)) + '%'

            realTimeData = {'num_census': num_census, 'num_votos': num_votos, 'participation': participation}
            context['realTimeData'] = realTimeData

        except:
            raise Http404

        return context
```

2. Para finalizar se desarrolló el fichero html:

En el caso de que una votación no haya finalizado, los datos a mostrar en la página web serán número de votantes, número de votos y porcentaje de participación.



```
{% block content %}  
  <div id="app-visualizer">  
    <!-- Navbar -->  
    <nav class="navbar navbar-dark bg-dark">  
      <div class="container-fluid">  
        <h1 class="navbar-brand">Decide</h1>  
      </div>  
    </nav>  
  
    <div class="voting container">  
      <h1>[[ voting.id ]] - [[ voting.name ]]</h1>  
  
      <h2 v-if="!voting.start_date">Votación no comenzada</h2>  
      <div v-else-if="!voting.end_date">  
        <h2>Votación en curso</h2>  
        <p>Información en tiempo real:</p>  
  
        <table class="table table-bordered table-striped">  
          <thead>  
            <tr>  
              <th>Número de votantes</th>  
              <th>Número de votos</th>  
              <th>Porcentaje de participación</th>  
            </tr>  
          </thead>  
          <tbody>  
            <tr>  
              <td id="census"> [[realTimeData.num_census]]</td>  
              <td id="votes"> [[realTimeData.num_votos]]</td>  
              <td id="participation"> [[realTimeData.participation]]  
            </td>  
          </tr>  
        </tbody>  
      </table>  
    </div>
```

Está diseñado para mostrar visualizaciones dinámicas de datos de votación, permitiendo al usuario interactuar con gráficos de barras y polares relacionados con las opciones de votación. Eso en el caso de que una votación haya finalizado.

```

● ● ●

<div v-else>

    <h2 class="heading">Resultados:</h2>

    <table class="table table-bordered table-striped">
        <thead>
            <tr>
                <th>Opción</th>
                <th>Puntuación</th>
                <th>Votos</th>
            </tr>
        </thead>
        <tbody>
            <tr v-for="opt in voting.postproc" :key="opt.number">
                <th>[[opt.option]]</th>
                <td>[[opt.postproc]]</td>
                <td class="text-muted">[[opt.votes]]</td>
            </tr>
        </tbody>
    </table>
    <select id="chart-select" onchange="showChart()">
        <option value="bar-chart">Bar Chart</option>
        <option value="polar-chart-post">Polar Chart (Puntuación)</option>
        <option value="polar-chart-votes">Polar Chart (Votos)</option>
    </select>

    <canvas id="bar-chart"></canvas>
    <canvas id="polar-chart-post" style="display:none"></canvas>
    <canvas id="polar-chart-votes" style="display:none"></canvas>
</div>

</div>
<% endblock %>

```

Además, este fichero html incluye un bloque de código HTML con scripts integrados, que se encuentra dentro de un bloque de una plantilla web. Sus funciones incluyen cargar bibliotecas esenciales, configurar la interactividad con Vue.js y generar gráficos utilizando Chart.js. Es decir, definir funciones para la interacción del usuario con los gráficos, y crear gráficos de barras y polares basados en los datos de la votación para mostrarlos en la página web.

El resultado sería lo siguiente:

The screenshot shows a web application interface for an open election. At the top, there is a navigation bar with the following items: "Decide", "Votación ▾", "Preguntas ▾", "Censo ▾", "Log out•", and "admin". Below the navigation bar, the main content area has a title "2 - Votación abierta" and a subtitle "Votación en curso". A sub-header "Información en tiempo real:" is followed by a table with the following data:

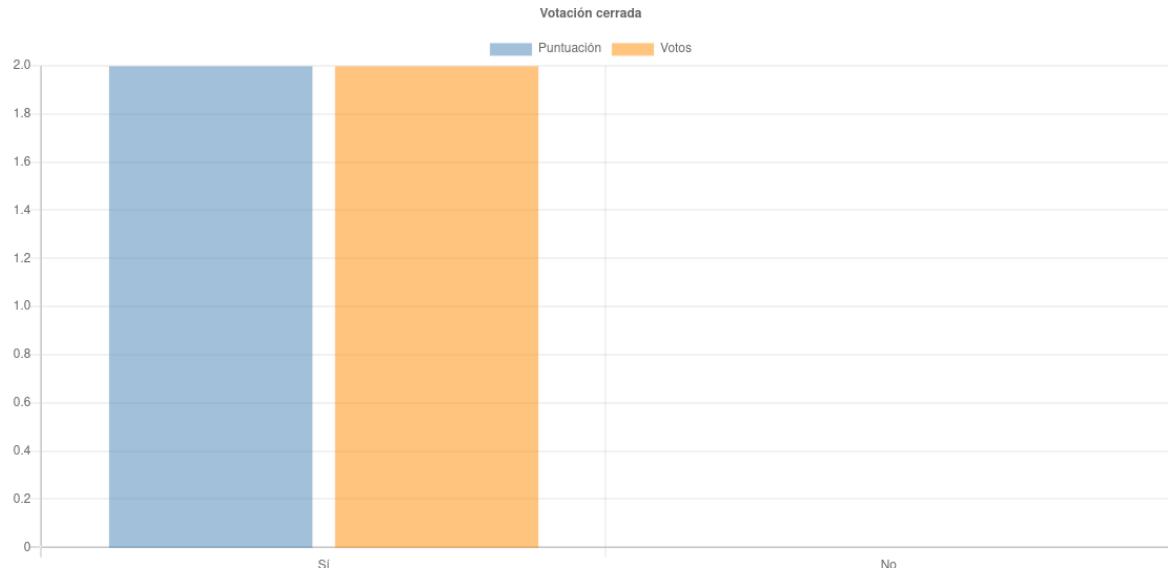
Número de votantes	Número de votos	Porcentaje de participación
3	1	33.33%

1 - Votación cerrada

Resultados:

Opción	Puntuación	Votos
Si	2	2
No	0	0

Bar Chart ▾



FrontEnd - Visualización gráficas e información en tiempo real

Una vez realizado las funcionalidades asociadas a la visualización de gráficas e información en tiempo real de las votaciones, era necesario integrarlo en la aplicación. Para ello, se hicieron los cambios necesarios en el módulo **visualizer**:

1. Se añadió en el fichero urls.py la dirección para encontrar la funcionalidad:

```

    ● ● ●
    urlpatterns = [
        path('<int:voting_id>', VisualizerView.as_view(), name='visualizer'),
    ]

```

También fue necesario realizar cambios en el módulo **voting**, más concretamente en el fichero html:

2. Se implementó un botón que en función de la variable 'start_date' o 'end_date', te conducía a una vista u otra.



```
<td>
    {% if voting.start_date != None or voting.end_date != None %}
        <form action="{% url 'visualizer' voting_id=voting.id %}" method="get">
            <button type="submit" class="btn btn-primary">Ver datos</button>
        </form>
    {% endif %}
</td>
```

A continuación se muestra la funcionalidad en una vista, donde si el usuario pulsa en el botón “Ver Datos”, le llevará a la vista para ver la información acerca de la votación:

Votación creada	None	None	IDENTITY	<button>Start</button>
Votación abierta	Dec. 5, 2022, 5:38 p.m.	None	IDENTITY	<button>Stop</button> <button>Ver datos</button>
Votación cerrada	Dec. 5, 2022, 4:50 p.m.	Dec. 5, 2022, 4:54 p.m.	IDENTITY	No more actions <button>Ver datos</button>

Importación y exportación de census

La implementación de las funcionalidades relacionadas con la exportación e importación de censos ha sido realizada por Guillermo Alonso Pacheco Rodrigues, miembro del grupo, quien ha introducido los siguientes cambios detallados a continuación.

En primer lugar, se inició la implementación de la funcionalidad de exportar censos. Para llevar a cabo esta tarea, se utilizó la biblioteca Django import/export, la cual, como su nombre indica, facilita la importación y exportación de datos desde la vista de administrador.

En el archivo *census/admin.py* se realizaron las siguientes modificaciones,

```
1      from import_export.admin import ImportExportModelAdmin
```

```

26   class CensusAdmin(ImportExportModelAdmin):
27       resource_class = CensusResource
28       list_display = ('voting_id', 'voter_id')
29       list_filter = (VotingIdFilter, )
30       search_fields = ('voter_id', )
31       actions = ["export_selected"]
32
33   @classmethod
34   def export_selected(cls, modeladmin, request, queryset):
35       response = HttpResponseRedirect(content_type='text/csv')
36       response['Content-Disposition'] = 'attachment; filename="census_export.csv"'
37
38       writer = csv.writer(response)
39       writer.writerow(['voting_id', 'voter_id'])
40
41       for census in queryset:
42           voting_id = census.voting_id if census.voting_id is not None else ''
43           voter_id = census.voter_id if census.voter_id is not None else ''
44           writer.writerow([voting_id, voter_id])
45
46       return response
47
48   export_selected.short_description = "Export census"
49
50
51
52   admin.site.register(Census, CensusAdmin)

```

También se creó una función que permite exportar únicamente los censos seleccionados, *export_selected*.

Posteriormente, al comenzar la implementación del frontend, se inició también la implementación de la importación de censos y la modificación del código de exportación para permitir la exportación desde el nuevo frontend.

La función responsable de exportar censos desde el frontend, ubicada en `census/views.py`, es la siguiente

```

58  def export_census(request):
59      selected_ids = request.GET.get('ids', '')
60      try:
61          selected_ids = [int(id) for id in selected_ids.split(',') if id]
62      except ValueError:
63          return HttpResponseBadRequest(json.dumps({'error': 'Invalid IDs provided'}), content_type='application/json')
64
65      if not selected_ids:
66          # If no IDs are selected, export all census data
67          census_list = Census.objects.all()
68      else:
69          try:
70              # Filter the census based on the selected IDs
71              census_list = Census.objects.filter(id__in=selected_ids)
72          except IntegrityError:
73              return HttpResponseBadRequest(json.dumps({'error': 'IDs no válidos'}), content_type='application/json')
74
75      response = HttpResponse(content_type='text/csv')
76      response['Content-Disposition'] = 'attachment; filename="census_export.csv"'
77
78      writer = csv.writer(response)
79      writer.writerow(['voting_id', 'voter_id'])
80
81      for census in census_list:
82          writer.writerow([census.voting_id, census.voter_id])
83
84      return response

```

A diferencia de la función en el archivo admin.py que utiliza la biblioteca django-import-export, esta función ha sido creada exclusivamente con la capacidad de exportar archivos en formato CSV, facilitando así la importación de estos mismos archivos.

La importación de archivos que se almacenarán en la base de datos puede ser complicada, por lo que es crucial que el código que permite esta funcionalidad esté bien validado.

A continuación, se presenta el código que valida los archivos .csv y sus datos,

```

86  def validate_ids(voting_id, voter_id):
87      if not Voting.objects.filter(id=voting_id).exists():
88          return False
89
90      if not User.objects.filter(id=voter_id).exists():
91          return False
92
93      return True

```

```

95 ✓  def validate_and_read_csv(csv_file):
96     try:
97         # Decodificar el archivo CSV y crear un lector de CSV
98         dataset = csv_file.read().decode('utf-8')
99         io_string = io.StringIO(dataset)
100        csv_reader = csv.reader(io_string, delimiter=',')
101
102        # Leer la primera linea del archivo CSV (encabezados)
103        headers = next(csv_reader)
104
105        # Verificar que los encabezados son correctos
106        if len(headers) != 2 or headers[0] != 'voting_id' or headers[1] != 'voter_id':
107            return None, 'El archivo CSV debe tener dos columnas: voting_id y voter_id.'
108
109        new_census_list = []
110        for row in csv_reader:
111            try:
112                voting_id = int(row[0])
113                voter_id = int(row[1])
114            except ValueError:
115                return None, 'Todos los valores deben ser enteros.'
116            if not validate_ids(voting_id, voter_id):
117                return None, 'El archivo contiene datos no existentes en la base de datos.'
118            elif Census.objects.filter(voting_id=voting_id, voter_id=voter_id).exists():
119                return None, 'El archivo contiene censos ya existentes.'
120            else:
121                # Crear un nuevo objeto Census
122                new_census = Census(voting_id=voting_id, voter_id=voter_id)
123                new_census_list.append(new_census)
124
125        return new_census_list, None
126    except Exception as e:
127        return None, f'Error al leer el archivo CSV: {str(e)}'

```

Finalmente, la función encargada de la importación es la siguiente,

```

129 ✓  def import_census(request):
130     ImportCensusForm(request.POST or None, request.FILES or None)
131     if request.method == 'POST':
132         csv_file = request.FILES.get('csv_file')
133         if not csv_file:
134             messages.error(request, 'No se ha subido ningún archivo CSV')
135             return redirect('all_census')
136         elif not csv_file.name.endswith('.csv'):
137             messages.error(request, 'El archivo debe ser un archivo CSV.')
138             return redirect('all_census')
139         else:
140             new_census_list, error = validate_and_read_csv(csv_file)
141             if error:
142                 messages.error(request, error)
143                 return redirect('all_census')
144
145             # Verificar si cada censo ya existe antes de agregarlo a new_census_list
146             unique_census_list = [census for census in new_census_list
147                                   if not Census.objects.filter(voting_id=census.voting_id,
148                                     voter_id=census.voter_id).exists()]
149
150             messages.success(request, 'Censo importado correctamente.')
151             with transaction.atomic():
152                 Census.objects.bulk_create(unique_census_list)
153
154             return redirect('all_census')
155
156     return redirect('all_census')

```

Para la creación de esta vista ha sido necesario realizar un formulario que permita importar un nuevo censo.

```

12 ✓  class ImportCensusForm(forms.Form):
13     csv_file = forms.FileField()
14     class Meta:
15         model = Census
16         fields = ('voting_id', 'voter_id')

```

FrontEnd - Voting.

Con el objetivo de implementar los cambios en el FrontEnd a continuación se detallarán los pasos seguidos para integrar el módulo Voting en nuestro sistema final.

Modificaciones realizadas en el módulo voting, concretamente en el views.py

1. Creación de la vista para mostrar todas las votaciones que existan hasta ese momento.

```
def all_votings(request):  
    if not request.user.is_staff:  
        template = loader.get_template('403.html')  
        return HttpResponseRedirect(template.render({}, request))  
    else:  
        votings = Voting.objects.all()  
        return render(request, 'allVotings.html', {'votings': votings, 'title': 'Votaciones'})
```

2. Creación de la vista que permite acceder a los detalles de cada votación, para acceder a cada una se hará mediante la Primary Key de la votación.

```
def votings_detail(request, voting_id):  
    if not request.user.is_staff:  
        template = loader.get_template('403.html')  
        return HttpResponseRedirect(template.render({}, request))  
    else:  
        voting = Voting.objects.get(pk=voting_id)  
        return render(request, 'votings_detail.html', {'voting': [voting], 'title': 'Votacion'})
```

3. Creación de la vista que permite al administrador crear una nueva votación.

```
def new_voting(request):  
    form = NewVotingForm()  
    if not request.user.is_staff:  
        template = loader.get_template('403.html')  
        return HttpResponseRedirect(template.render({}, request))  
  
    if request.method == 'POST':  
        form = NewVotingForm(request.POST, request.FILES)  
  
        if form.is_valid():  
            item=form.save(commit=False)  
            item.save()  
  
            return redirect('/voting/allVotings')  
        else:  
            form = NewVotingForm()  
  
    return render(request, 'form.html', {  
        'form': form,  
        'title': 'Nueva Votación',  
    })
```

En el caso de que el usuario que intente crear una votación y no tenga permisos de administrador este será redirigido a una vista de prohibido.

4. Para poder realizar la nueva votación ha sido necesario crear un formulario para esta.

```
class NewVotingForm(forms.ModelForm):
    class Meta:
        model = Voting
        fields = ('name', 'desc', 'question', 'model', 'seats',)

    widgets = {
        'name': forms.TextInput(attrs={
            'class': INPUT_CLASSES
        }),
        'description': forms.Textarea(attrs={
            'class': INPUT_CLASSES
        }),
        'question': forms.Select(attrs={
            'class': INPUT_CLASSES
        }),
        'model': forms.Select(attrs={
            'class': INPUT_CLASSES
        }),
        'seats': forms.TextInput(attrs={
            'class': INPUT_CLASSES
        }),
    }
}
```

5. Creación de la vista que permite editar una votación. Se accederá a ella mediante la Primary Key y se reutilizará el formulario de creación de la votación.

```
def edit_voting(request, voting_id):
    voting=get_object_or_404(Voting, pk=voting_id)

    if not request.user.is_staff:
        template = loader.get_template('403.html')
        return HttpResponseRedirect(template.render({}, request))

    if request.method == 'POST':
        form = EditVotingForm(request.POST, request.FILES, instance=voting)

        if form.is_valid():

            voting_instance = form.save(commit=False)
            voting_instance.save()

            form.save_m2m()

            return redirect('/voting/allVotings')
    else:
        form = EditVotingForm(instance=voting)

    return render(request, 'form.html', {
        'form': form,
        'title': 'Modificar Votación',
    })
```

6. Como uno de los aspectos necesarios para crear una votación es el Auth, ha sido necesario implementar una vista que permita la creación de un Auth.

```
def new_auth(request):
    if not request.user.is_staff:
        template = loader.get_template('403.html')
        return HttpResponseRedirect(template.render({}, request))

    if request.method == 'POST':
        form = NewAuthForm(request.POST, request.FILES)
        if form.is_valid():
            item = form.save()
            return redirect('/base')

    else:
        form = NewAuthForm()

    return render(request, 'authForm.html', {
        'form': form,
        'title': 'Nuevo Auth',
    })
```

7. Para la creación de esta vista ha sido necesario realizar un formulario que permita crear un nuevo Auth.

```
class NewAuthForm(forms.ModelForm):
    class Meta:
        model = Auth
        fields = ('name', 'url', 'me')

    widgets = {
        'name': forms.TextInput(attrs={
            'class': INPUT_CLASSES
        }),
        'url': forms.TextInput(attrs={
            'class': INPUT_CLASSES
        }),
        'me': forms.CheckboxInput(attrs={
            'class': INPUT_CLASSES
        }),
    }
```

8. Por último ha sido necesario crear las vistas que permitan las operaciones start, stop y tally voting.

Para la creación de estas funciones el proceso a seguir ha sido trasladar a la vista las operaciones ya desarrolladas en la aplicación base.

```
def start_voting(request):
    if not request.user.is_staff:
        template = loader.get_template('403.html')
        return HttpResponseRedirect(template.render({}, request))
    else:
        if request.method == 'POST':
            voting_id = request.POST.get('voting_id')
            voting = get_object_or_404(Voting, pk=voting_id)
            voting.create_pubkey()
            voting.start_date = timezone.now()
            voting.save()

    return redirect('/voting/allVotings')

def stop_voting(request):
    if not request.user.is_staff:
        template = loader.get_template('403.html')
        return HttpResponseRedirect(template.render({}, request))
    else:
        if request.method == 'POST':
            voting_id = request.POST.get('voting_id')
            voting = get_object_or_404(Voting, pk=voting_id)
            voting.end_date = timezone.now()
            voting.save()

    return redirect('/voting/allVotings')

def tally_voting(request):
    if not request.user.is_staff:
        template = loader.get_template('403.html')
        return HttpResponseRedirect(template.render({}, request))
    else:
        if request.method == 'POST':
            voting_id = request.POST.get('voting_id')
            voting = get_object_or_404(Voting, pk=voting_id)
            token = request.session.get('auth-token', '')
            voting.tally_votes(token)

    return redirect('/voting/allVotings')
```

9. A continuación se mostrarán capturas del resultado final de nuestra página.

a. Formulario para la creación de una nueva votación

The screenshot shows a web application interface for creating a new voting session. At the top, there is a header bar with the URL '127.0.0.1:8000/voting/newVoting/'. Below the header, the page title is 'Decide' and the main section title is 'Nueva Votación'. The form fields include:

- Name:** A text input field.
- Desc:** A large text area for description.
- Question:** A text input field containing '-----'.
- Auths:** A text input field containing 'http://localhost:8000'.

At the bottom right of the form, there is a blue button labeled 'Crear' (Create).

b. Vista que muestra todas las votaciones y las operaciones start, stop y tally.

The screenshot shows a list of voting sessions. At the top, there is a header bar with the URL '127.0.0.1:8000/voting/allVotings/'. Below the header, the page title is 'Decide' and the main section title is 'Votaciones'. The table displays the following data:

Name	Start Date	End Date	Model	Actions	Datos
isisi	Dec. 18, 2023, 5:34 p.m.	None	IDENTITY	<button>Stop</button>	<button>Ver datos</button>
humano	Dec. 18, 2023, 5:25 p.m.	None	IDENTITY	<button>Stop</button>	<button>Ver datos</button>
Votación creada	None	None	IDENTITY	<button>Start</button>	
Votación abierta	Dec. 5, 2022, 5:38 p.m.	None	IDENTITY	<button>Stop</button>	<button>Ver datos</button>
Votación cerrada	Dec. 5, 2022, 4:50 p.m.	Dec. 5, 2022, 4:54 p.m.	IDENTITY	No more actions	<button>Ver datos</button>

At the bottom left, there is a message 'Showing 1 to 5 of 5 entries' and navigation links 'Previous', '1', and 'Next'.

c. Formulario de creación de Auth.

The screenshot shows a web application interface. At the top, there is a header bar with a search field containing '127.0.0.1:8000/voting/newAuth/'. Below the header, there is a navigation bar with tabs: 'Decide' (selected), 'Votación', 'Preguntas', 'Censo', 'Log out• admin'. The main content area has a title 'Nuevo Auth'. It contains three input fields: 'Name:' with an empty text input, 'Url:' with an empty text input, and 'Me: '. At the bottom right of the form is a button labeled 'Nuevo Auth'.

10. Vista utilizada para la creación de las preguntas utilizadas en las votaciones.

```
✓ def QuestionCreateView(request):
    if not request.user.is_staff:
        return render(request, '403.html')

    if request.method == 'POST':
        form = QuestionForm(request.POST)
        if form.is_valid():
            question = form.save()
            options_formset = QuestionOptionFormSet(request.POST, instance=question)
            if options_formset.is_valid():
                options_formset.save()
            return redirect('/base/')
    else:
        form = QuestionForm()

    return render(request, 'question_create.html', {'form': form})
```

11. Formulario necesario para crear las votaciones.

```
class QuestionOptionForm(forms.ModelForm):
    class Meta:
        model = QuestionOption
        fields = ['number', 'option']

QuestionOptionFormSet = forms.inlineformset_factory(Question, QuestionOption, form=QuestionOptionForm, extra=1, can_delete=True)

class QuestionForm(forms.ModelForm):
    class Meta:
        model = Question
        fields = ['desc', 'cattegory']

options = QuestionOptionFormSet()
```

12. Vista para mostrar todas las preguntas realizadas hasta ahora.

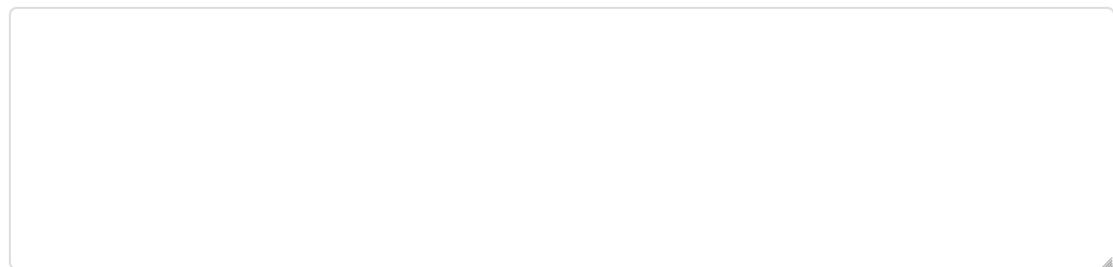
```
def all_question(request):
    if not request.user.is_staff:
        return render(request, '403.html')

    else:
        questions = Question.objects.all()
        return render(request, 'all_question.html', {'questions': questions, 'title': 'Preguntas'})
```

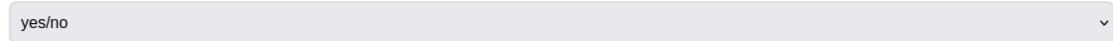
13. A continuación se detalla como es el resultado final en nuestro FrontEnd

En primer lugar este es el formulario utilizado para crear una nueva pregunta

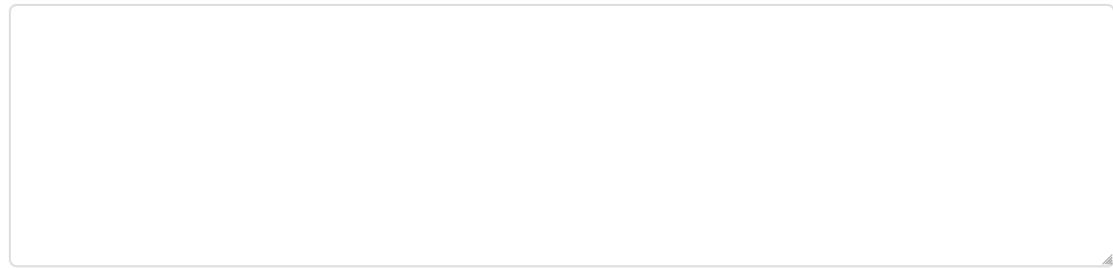
Desc:

A large, empty text input field with a thin black border, intended for the user to type in the description of the new question.

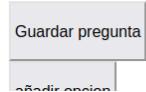
Cattegory:

A horizontal select dropdown menu with a light gray background and a thin black border. It contains the options "yes/no".

Number: Option:

A large, empty text input field with a thin black border, intended for the user to type in the number of the option.

Delete:



A set of two rectangular buttons. The top button is light gray with the text "Guardar pregunta" in black. The bottom button is also light gray but has very small, illegible text.

Resultado de mostrar todas las preguntas.

Decide

Votación Preguntas Censo

Preguntas

Show 10 entries

Search:

Name	Type
Eres un humano?	YES/NO
¿Es esto una votación de Si/No?	YES/NO
¿Es esto una votación correcta?	OPTIONS

Showing 1 to 3 of 3 entries

[Previous](#) [Next](#)

FrontEnd - Census

Con el objetivo de implementar los cambios en el FrontEnd a continuación se detallarán los pasos seguidos para integrar el módulo Census en nuestro sistema final.

1. Creación de la vista para mostrar todos los censos que existan hasta ese momento.

```
30  def all_census(request):  
31      census_list = Census.objects.all()  
32  
33      context = {  
34          'census_list': census_list,  
35          'messages': request.session.pop('all_census_messages', None),  
36      }  
37  
38      return render(request, 'all_census.html', context)
```

En esta vista, además de la funcionalidad de importar y exportar censos descritas anteriormente, se añade la funcionalidad de eliminar censos, pudiendo eliminar todos los censos o únicamente los seleccionados.

```

40  def delete_census(request):
41      if request.method == 'POST':
42          census_ids = request.POST.get('selected_censuses', '').split(',')
43          if not census_ids or not census_ids[0]:
44              # No se seleccionó ningún censo, eliminar todos
45              Census.objects.all().delete()
46              messages.success(request, 'Todos los censos han sido eliminados.')
47              return redirect('all_census')
48
49          censuses = Census.objects.filter(id__in=census_ids)
50          if not censuses.exists():
51              messages.error(request, 'No existen censos con los IDs proporcionados.')
52          else:
53              censuses.delete()
54              messages.success(request, 'Censos eliminados correctamente.')
55
56      return redirect('all_census')

```

A continuación se muestran capturas de la vista.

Voting ID	Voter ID	
1	1	<input type="checkbox"/>
1	2	<input type="checkbox"/>
2	2	<input type="checkbox"/>
2	3	<input type="checkbox"/>
2	4	<input type="checkbox"/>

Show 10 entries
Search:

Showing 1 to 5 of 5 entries
[Previous](#)[1](#)[Next](#)

Seleccionar idioma ▾

La interfaz presenta mensajes indicando errores o éxito en respuesta a las acciones realizadas. A continuación se muestran ejemplos.

Todos los censos han sido eliminados. ×

[Examinar...](#) No s...ivo. [Importar Censo](#)

[Exportar Censo](#)

[Eliminar censos](#)

Show 10 entries

Search:

Voting ID

Voter ID

No data available in table

Showing 0 to 0 of 0 entries

[Previous](#)[Next](#)

Seleccionar idioma ▾

Censo importado correctamente. ×

[Examinar...](#) No s...ivo. [Importar Censo](#)

[Exportar Censo](#)

[Eliminar censos](#)

Show 10 entries

Search:

Voting ID

Voter ID

1	1	<input type="checkbox"/>
1	2	<input type="checkbox"/>
2	2	<input type="checkbox"/>
2	3	<input type="checkbox"/>
2	4	<input type="checkbox"/>

Showing 1 to 5 of 5 entries

[Previous](#)[1](#)[Next](#)

Decide

Votación ▾ Preguntas ▾ Censo ▾ Log out• admin

El archivo contiene datos no existentes en la base de datos.



[Examinar...](#) No s...ivo. [Importar Censo](#)

[Exportar Censo](#)

[Eliminar censos](#)

Show 10 ▾ entries

Search:

Voting ID	Voter ID	
1	1	<input type="checkbox"/>
1	2	<input type="checkbox"/>
2	2	<input type="checkbox"/>
2	3	<input type="checkbox"/>
2	4	<input type="checkbox"/>

Showing 1 to 5 of 5 entries

[Previous](#)[1](#)[Next](#)

Decide

Votación ▾ Preguntas ▾ Censo ▾ Log out• admin

El archivo debe ser un archivo CSV.



[Examinar...](#) No s...ivo. [Importar Censo](#)

[Exportar Censo](#)

[Eliminar censos](#)

Show 10 ▾ entries

Search:

Voting ID	Voter ID	
1	1	<input type="checkbox"/>
1	2	<input type="checkbox"/>
2	2	<input type="checkbox"/>
2	3	<input type="checkbox"/>
2	4	<input type="checkbox"/>

Showing 1 to 5 of 5 entries

[Previous](#)[1](#)[Next](#)

2. Creación de la vista que permite al administrador crear un nuevo censo.

```

194     def new_census_form(request):
195         form = NewCensusForm()
196
197         if not request.user.is_staff:
198             template = loader.get_template('403.html')
199             return HttpResponseRedirect(template.render({}, request))
200
201         if request.method == 'POST':
202             form = NewCensusForm(request.POST, request.FILES)
203
204             if form.is_valid():
205                 voting_id = form.cleaned_data.get('voting_id')
206                 voter_id = form.cleaned_data.get('voter_id')
207
208                 if not validate_ids(voting_id, voter_id):
209                     request.session['form_messages'] = [{ 'message': 'Voting ID o Voter ID inválidos.', 'tag': 'error' }]
210                 else:
211                     try:
212                         item = form.save(commit=False)
213                         item.save()
214                         request.session['form_messages'] = [{ 'message': 'Censo creado exitosamente.', 'tag': 'success' }]
215                         return redirect('new_census')
216                     except IntegrityError:
217                         print("IntegrityError caught")
218                         request.session['form_messages'] = [{ 'message': 'Censos con este Voting ID o Voter Id ya existen', 'tag': 'error' }]
219                 else:
220                     request.session['form_messages'] = [{ 'message': f'{field.capitalize()}: {error}', 'tag': 'error' }
221                                         for field, errors in form.errors.items() for error in errors]
222                     if '__all__' in form.errors:
223                         for error in form.errors['__all__']:
224                             if 'already exists' in error:
225                                 request.session['form_messages'] = [{ 'message': 'Censos con este Voting ID o Voter ID ya existen', 'tag': 'error' }]
226
227                     messages = request.session.get('form_messages', None)
228                     response = render(request, 'new_census_form.html', {
229                         'form': form,
230                         'title': 'Nuevo Censo',
231                         'messages': messages,
232                     })
233
234                     if 'form_messages' in request.session:
235                         del request.session['form_messages']
236
237             return response

```

Para la creación de esta vista ha sido necesario realizar un formulario que permita crear un nuevo censo.

```

7      class NewCensusForm(forms.ModelForm):
8          class Meta:
9              model = Census
10             fields = ('voting_id', 'voter_id')
11

```

En esta vista, la validación de datos cobra gran importancia para evitar la creación de censos con información no existente en la base de datos.

A continuación se muestran capturas de la vista.

La interfaz presenta mensajes indicando errores o éxito en respuesta a las acciones realizadas. A continuación se muestran ejemplos.

Nuevo Censo

Voting ID o Voter ID inválidos.

Voting id:

Voter id:

[Crear Censo](#)

Nuevo Censo

Censo creado exitosamente.

Voting id:

Voter id:

[Crear Censo](#)

Visión global del proceso de desarrollo.

El proceso de desarrollo del proyecto se caracteriza por seguir un enfoque estructurado y colaborativo. En detalle, se describen las distintas etapas del ciclo de vida del desarrollo, desde la detección inicial del cambio a implementar hasta que dicho cambio se encuentra plenamente operativo en producción. Se tomará como ejemplo el proceso de implementación de Google OAuth.

1. Detección del cambio:

La etapa inicial del proceso de desarrollo de software implica la detección del cambio, la cual puede surgir por la identificación de una nueva funcionalidad que debe ser implementada o la necesidad de realizar una corrección en el sistema existente. En este ejemplo específico, nos encontramos ante la necesidad de incorporar una nueva funcionalidad para mejorar la experiencia del usuario y aumentar la eficiencia del sistema.

Sea cual sea la naturaleza del cambio, ya sea funcional, estructural o de corrección, es responsabilidad del desarrollador que lo detecta registrar esta modificación en una incidencia. Este registro se convierte en un paso crucial para documentar y realizar un seguimiento efectivo de las modificaciones a lo largo del desarrollo del proyecto, asegurando la trazabilidad y la coherencia en el proceso de desarrollo de software.

El registro detallado de las incidencias facilita la comunicación entre los miembros del equipo de desarrollo, proporciona transparencia en el proceso de desarrollo y sirve como referencia para futuras actualizaciones y mantenimiento del software.

En resumen, la detección y documentación adecuada de los cambios en el software en la etapa inicial del proceso de desarrollo son pasos cruciales para garantizar un desarrollo efectivo y una implementación exitosa de nuevas funcionalidades o correcciones en el sistema.

2. Creación de la issue:

En lo que respecta a la gestión de incidencias en el proyecto, nuestro equipo ha optado por utilizar la herramienta "Issues" proporcionada por GitHub. A través de esta plataforma, los desarrolladores tienen acceso a todas las incidencias que surgen durante el proceso de desarrollo y pueden registrar cambios utilizando diversas plantillas previamente creadas por el equipo.

Siguiendo con el ejemplo mencionado, una vez que se ha identificado la necesidad de la nueva funcionalidad, el desarrollador procederá al registro de dicha incidencia. En este caso particular, elegirá la plantilla designada con el nombre "Feature request" para documentar de manera específica los detalles relacionados con la nueva funcionalidad a implementar. Este enfoque estructurado facilita una comunicación clara y eficiente sobre los cambios planificados en el proyecto. En esta plantilla deberá llenar los siguientes campos:

- Título.
- Descripción.
 - ¿Su solicitud está relacionada con un problema? Descríbalo.
Una descripción clara y concisa del problema.
 - Descripción
Una descripción clara y concisa de lo que quiere desarrollar.
- Asignación.

- Etiqueta: deberá especificar el tipo y prioridad de la incidencia.
- Rama de desarrollo (si procede).

Según el ejemplo, la incidencia debería resultar en algo parecido a:

Issue: Feature request

Suggest an idea for this project. If this doesn't look right, [choose a different type](#).

Add a title

Implementación de la autenticación vía Google OAuth

Add a description

Write Preview

¿Su solicitud está relacionada con un problema? Describalo.
No aplica

Descripción
La funcionalidad a realizar consiste en la configuración de la API de Google para permitir la autenticación vía Google OAuth en la aplicación, gracias a ello, los usuarios podrán iniciar sesión con una cuenta de Google.

Markdown is supported Paste, drop, or click to add files

Assignees javierrc8 Labels Authentication enhancement Low priority Projects None yet Milestone No milestone Development Shows branches and pull requests linked to this issue Helpful resources GitHub Community Guidelines

Submit new issue

3. Asignación y creación de ramas:

Tras la creación de la incidencia, se debe asignar a un desarrollador para que realice el cambio, que puede ser el mismo que ha detectado la incidencia o cualquier otro miembro del equipo. El desarrollador designado para abordar la tarea procederá a la asignación y creación de una rama, siguiendo rigurosamente el flujo de trabajo establecido por GitFlow. La creación de esta rama tiene como propósito proporcionar un entorno aislado que permita el desarrollo de la funcionalidad sin incidir en otras partes del proyecto. Tomando el ejemplo mencionado, la rama se generaría con origen en "develop" y llevaría un nombre que refleje la función específica, como por ejemplo "feature/google-auth". Este enfoque de ramificación estructurado facilita un control preciso sobre los cambios, garantizando que cada funcionalidad se desarrolle de manera independiente antes de ser integrada al proyecto principal.

4. Desarrollo y commits:

Tras la creación de la rama, el desarrollador deberá realizar el siguiente paso según haya sido creada. En caso de haber sido creada a través de la interfaz de GitHub, deberá realizar un pull para tener esa rama en su repositorio local. Si en cambio, la rama ha sido creada a través de la línea de comandos del sistema, simplemente deberá acceder a esta. En cualquier caso, será siempre dentro de esta rama donde realizará la funcionalidad completa junto con las pruebas que sean necesarias para realizar la cobertura del código implementado.

Durante este proceso, el desarrollador podrá realizar varios commits, según crea necesario, que mantengan un changelog representativo y que permita realizar una lectura de los distintos incrementos en la funcionalidad. Para ello, el desarrollador deberá seguir las prácticas establecidas por "Conventional Commits", que en caso de cualquier duda podrá consultar en la wiki oficial <https://www.conventionalcommits.org/en/v1.0.0/>. Con esto se pretende mantener un historial claro y estructurado de los cambios realizados.

En el ejemplo, imagínese que el desarrollador tras haber realizado parte de la funcionalidad, desarrolla las pruebas unitarias pertinentes. El commit será del tipo "test: implement unit

test cases for google oauth”.

5. Creación de pull request:

Una vez la funcionalidad ha sido desarrollada y probada al completo, se procede a la creación de un pull request que solicite el volcado (en el ejemplo a la rama “develop”) del código realizado. El desarrollador deberá asociar la issue que resuelve y solicitar la revisión de al menos un miembro del equipo, que no haya participado en la realización de la funcionalidad. Hasta que este no apruebe los cambios, la funcionalidad no podrá ser volcada. Si surgen conflictos, deberán ser resueltos por el desarrollador propietario de la funcionalidad.

Al tratarse en el ejemplo de un volcado a “develop”, una rama principal que se encuentra configurada para lanzar los workflows diseñados en el actions, cuando se cree el pull request se lanzarán los distintos workflows configurados. Estos workflows son:

- Actions: realizará el build de la aplicación, lanzando todas las pruebas del sistema, asegurando que no se realice un volcado con errores en el código, he aquí una demostración de este workflow.
- Codacy: realizará un análisis del código desarrollado, midiendo la cobertura y detectando posibles bad smells.

The screenshot shows a GitHub pull request interface. At the top, it says "codacy-production bot commented 18 hours ago · edited". Below that is a section titled "Coverage summary from Codacy" with a link "See diff coverage on Codacy". It displays two boxes: "Coverage variation" (+1.78% target: -1.00%) and "Diff coverage" (92.54%). There are also links for "Coverage variation details" and "Diff coverage details". At the bottom, there are links for "See your quality gate settings" and "Change summary preferences". A note at the very bottom states: "You may notice some variations in coverage metrics with the latest Coverage engine update. For more details, visit the [documentation](#)".

build (3.10.12)	
succeeded 2 days ago in 5m 41s	
>	✓ Set up job
>	✓ Initialize containers
>	✓ Run actions/checkout@v3
>	✓ Set up Python 3.10.12
>	✓ psycopg2 prerequisites
>	✓ Install dependencies and config
>	✓ Install dependencies and config
>	✓ Run migrations (unnecessary)
>	✓ Run tests
>	✓ Codacy Coverage Reporter
>	✓ Post Set up Python 3.10.12
>	✓ Post Run actions/checkout@v3
>	✓ Stop containers
>	✓ Complete job

6. Revisión y aprobación:

La fase de revisión y aprobación desempeña un papel crucial en el proceso. El miembro designado para llevar a cabo la revisión deberá realizar una inspección exhaustiva de los cambios implementados por el desarrollador. Durante este proceso, se fomenta la retroalimentación constructiva, permitiendo la identificación y discusión de posibles mejoras o problemas potenciales. La pull request estará lista para ser fusionada una vez que haya obtenido la aprobación de todos los miembros que han sido solicitados para revisarla, asegurando así un consenso y una validación colectiva antes de la integración de los cambios en el proyecto principal. Este enfoque colaborativo garantiza la calidad y la coherencia en cada fase del desarrollo.

7. Volcado y cierre de la issue:

Después de que se haya completado el proceso de fusión, dado que la issue está vinculada a la pull request, se cerrará automáticamente. El volcado de la pull request marca el paso final para la incorporación exitosa de los cambios propuestos al proyecto. Este enfoque de cierre

automático garantiza una gestión eficiente y organizada de las incidencias, proporcionando una clara señal de que la funcionalidad o corrección asociada ha sido implementada con éxito en el proyecto principal. Este paso contribuye a mantener la transparencia y la trazabilidad en el desarrollo del proyecto, asegurando que cada cambio se documente y cierre adecuadamente en el contexto de la issue correspondiente.

Es importante resaltar que en cualquier fase del proceso, un desarrollador tiene la facultad, y en algunos casos la obligación, de dejar comentarios en la issue para informar sobre cualquier problema que surja durante la resolución del cambio. Este enfoque de comunicación abierta y colaborativa promueve la identificación temprana de posibles obstáculos, facilitando así la resolución eficiente de problemas y permitiendo una adaptación fluida en respuesta a los desafíos que puedan surgir durante el desarrollo del proyecto. La capacidad de comentar en la issue proporciona un canal directo para la comunicación entre los miembros del equipo, contribuyendo a la mejora continua y al éxito general del proyecto.

Entorno de desarrollo.

A continuación, se presenta una descripción detallada de los aspectos clave del entorno de desarrollo empleado en este proyecto, ofreciendo una visión integral de las herramientas y procesos que han contribuido a la creación y evolución del software.

1. Lenguaje de Programación.

El lenguaje de programación utilizado para el desarrollo de nuestro proyecto ha sido Python en la versión Python 3.

2. Entorno de desarrollo integrado (IDE).

Para el desarrollo del proyecto el grupo de desarrollo decidió utilizar como IDE Visual Studio code.

3. Sistema de control de versiones.

El sistema de control de versiones utilizado ha sido Git. A su vez se ha utilizado como plataforma de almacenamiento GitHub.

4. Sistema operativo

El sistema operativo recomendado por la universidad para la realización del proyecto ha sido Ubuntu 22.04.

Para lanzar el proyecto es necesario realizar los siguientes pasos.

1. Creación de un entorno virtual:

- a. Para la creación del entorno virtual es necesario la creación de un entorno virtual.

Para ello es necesaria la dependencia de python 3 “python3.10-venv”

- b. Tras realizar la instalación de la dependencia, creamos en entorno virtual con el nombre “python3.10 -m venv <nombre_del_entorno>”.

- c. Una vez creado el entorno lo activamos mediante el comando source <nombre_del_entorno>/bin/activate.

2. Es necesario instalar las versiones utilizadas por el equipo de trabajo para la realización del proyecto. Estas se encuentran en el fichero requirements.txt

A continuación se adjunta una captura de pantalla del requirements.txt con el objetivo de mostrar las versiones utilizadas.

```
Django==4.1
pycryptodome==3.15.0
django-restframework==3.14.0
django-cors-headers==3.13.0
requests==2.28.1
django-filter==22.1
psycopg2==2.9.4
coverage==6.5.0
jsonnet==0.18.0
django-nose==1.4.6
pynose==1.4.8
django-rest-swagger==2.2.0
selenium==4.7.2
django-allauth
django-import-export
```

- a. La instalación de las dependencias se realizará con el comando
pip install -r requirements.txt
- 3. Como paso previo a realizar las migraciones será crear una base de datos.
 - a. La base de datos se creará mediante el comando:
sudo su - postgres
psql -c "CREATE database decidedb OWNER decideuser"
- 4. Es necesario realizar las migraciones para la construcción correcta de la base de datos.
 - a. Esto se realizará mediante el comando: ./manage.py migrate y ./manage.py makemigrations
- 5. Para poder lanzar el proyecto es necesario realizar la configuración del fichero local_settings.py.
 - a. A continuación se detalla la configuración utilizada por los miembros del equipo de desarrollo.

```

ALLOWED_HOSTS = ["*"]

# Modules in use, commented modules that you won't use
MODULES = [
    'authentication',
    'base',
    'booth',
    'census',
    'mixnet',
    'postproc',
    'store',
    'visualizer',
    'voting',
]

BASEURL = 'http://localhost:8000'

APIS = {
    'authentication': BASEURL,
    'base': BASEURL,
    'booth': BASEURL,
    'census': BASEURL,
    'mixnet': BASEURL,
    'postproc': BASEURL,
    'store': BASEURL,
    'visualizer': BASEURL,
    'voting': BASEURL,
}

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'decidedb',
        'USER': 'decideuser',
        'PASSWORD': 'decide',
        'HOST': 'localhost',
        'PORT': '5432',
    }
}

# number of bits for the key, all auths should use the same number of bits
KEYBITS = 256

```

6. Para el correcto funcionamiento es necesario realizar la población inicial ya que algunas de las funcionalidades desarrolladas dependen de esto.
 - a. Esto se realizará mediante el comando `./manage.py loaddata populate.json`
7. Por último, lanzamos el servidor mediante el comando `./manage.py runserver` y accedemos a la url `http://localhost:8000`.

Ejercicio de propuesta de cambio.

A continuación se presenta el ciclo completo a realizar cuando se implementa un nuevo cambio/funcionalidad.

- En primer lugar empezamos con la creación de la issue.
 - En el caso de que sea una issue relacionada con un incremento se utilizará la plantilla proporcionada para ello.
 - En el caso de que sea una issue relacionada con un cambio en la configuración se utilizará la plantilla proporcionada para ello.
 - En el caso de que sea una issue que no se corresponda con los casos anteriores se procederá a crear una issue en blanco con la mayor cantidad de información posible.

The screenshot shows a user interface for creating a new issue. There are two main sections: 'Configuration request' and 'Feature request'. Each section has a text input field and a 'Get started' button. The 'Configuration request' section has placeholder text 'Describe the changes that need to be done.' and the 'Feature request' section has placeholder text 'Suggest an idea for this project'.

- En el proceso de la creación de la issue será necesario asignar las etiquetas correspondientes.
 - En función de la naturaleza de la issue.
 - En función de la prioridad que requiera la issue.

Issue: Feature request

Suggest an idea for this project. If this doesn't look right, choose a different type.

The screenshot shows a 'New Issue' form for a 'Feature request'. The form includes fields for 'Add a title' (with placeholder 'Issue relacionada con un incremento.'), 'Add a description' (with placeholder 'Una descripción clara y concisa del problema.'), 'Assignees' (set to 'edubusluc'), 'Labels' (set to 'Booth' and 'High priority'), 'Projects' (set to 'None yet'), 'Milestone' (set to 'No milestone'), and 'Development' (with a note about showing branches and pull requests). The form also includes a rich text editor toolbar and a file upload area.

Submit new issue

Podemos ver cómo para este caso hemos agregado las etiquetas de high priority para indicar que es un cambio de alta prioridad y la etiqueta Booth para indicar que el cambio va relacionado a este módulo.

A su vez ha sido necesario asignar la issue a un miembro del equipo de desarrollo.

También será necesario asignar la rama correspondiente a esta issue. Esto lo haremos más adelante puesto que la rama no ha sido creada.

- El siguiente como ya hemos adelantado será la creación de una rama para realizar los cambios relacionados con la issue.
 - En nuestro caso la rama tendrá el nombre feature/.... puesto que en nuestro proyecto se ha seguido GitFlow.
 - Esto se podrá realizar mediante el comando git branch feature/....
 - Para cambiar a esta rama lo haremos mediante el comando de git checkout <nombre_de_la_rama>
 - Esta operación también puede realizarse a través de la interfaz gráfica de github.
 - Una vez creada la rama esta se añadirá a la issue que hemos creado anteriormente.
- Ahora pasamos al entorno de producción para implementar el cambio que necesitamos realizar.
 - A continuación explicaremos la metodología que hemos seguido para realizar los commits.
 - Hemos seguido conventional commit por lo que los commits tendrán esta estructura.
 - feat: <mensaje_commit>. En el caso de que el commit a realizar implique el incremento en una funcionalidad a desarrollar.
 - test: <mensaje_commit>. En el caso de que el commit implique la creación de un test.
 - fix: <mensaje_commit>. En el caso de que el commit a realizar implique la corrección de un error existente.
 - chore: <mensaje_commit>. Para representar cambios que no afecten al código fuente del proyecto.
 - El commit se realizará mediante el comando git commit -m "Etiquetas explicadas previamente".
 - Seguidamente se realizará un git push para subir el cambio al repositorio remoto.
- Si la funcionalidad se ha completado de forma satisfactoria es hora de realizar la pull request para integrarlo con el resto del código del proyecto.
 - Una vez creada la pull request será necesario asignar un revisor que no haya tenido implicación en esta funcionalidad, consiguiendo así que la revisión no venga condicionada por el desarrollador. Este aprobará la pull request, dejando siempre un mensaje de confirmación y realizará el merge a la rama destino.

The screenshot shows a GitHub pull request review interface. At the top, there is a circular profile picture of a person and a green checkmark icon. Next to it, the text "elguille2 approved these changes yesterday" is displayed, along with a "View reviewed changes" link. Below this, a horizontal line separates the header from the main comment area. In the comment area, there is another circular profile picture followed by the text "elguille2 left a comment". Underneath the comment, the text "All tests working correctly" is shown. At the bottom of the comment area, there is a small smiley face emoji icon.

- En la creación de la pull request será necesario añadir el comentario closes #de la issue a cerrar, además de añadirla al apartado development.

chore: issue template added #63

The screenshot shows a GitHub pull request page for issue #63. The pull request has been merged and is now closed. A comment from javierrc8 contains the text "closes #61", which is highlighted with a red box. The right sidebar shows the "Development" section, which includes a note about merging closing issues and a link to the creation of the issue template. This section is also highlighted with a red box.

- Además en el caso de que la pull request se haya realizado sobre las ramas develop, frontend-integration o master, lanzará un ciclo de integración continua.

The screenshot shows a GitHub Actions build log for workflow "build (3.10.12)". The build succeeded yesterday in 6m 46s. The log details the steps taken: Set up job, Initialize containers, Run actions/checkout@v3, Set up Python 3.10.12, psycopg2 prerequisites, Install dependencies and config, Install dependencies and config, Run migrations (unnecessary), Run tests, Codacy Coverage Reporter, Post Set up Python 3.10.12, Post Run actions/checkout@v3, Stop containers, and Complete job. The entire log area is displayed.

- El revisor será el encargado de comprobar si el ciclo de integración continua se ha realizado de manera correcta y no se ha detectado ningún tipo de error.

Conclusiones y trabajo futuro.

El equipo de trabajo tras la realización completa del proyecto ha obtenido las siguientes conclusiones.

- Como equipo de desarrollo, creemos que hemos hecho un buen trabajo a la hora de la implementación de las nuevas funcionalidades, especialmente en la creación de un frontend que integre gran parte de las características de Decide, dejando un sistema más intuitivo para el usuario.
- Hemos aprendido cómo gestionar de forma correcta, las incidencias que han surgido en nuestro proyecto en especial, después de la migración de las issues a github, que ha hecho que este proceso de gestión sea más eficiente, transparente y colaborativo.
- El equipo de desarrollo ha obtenido valores importantísimos en cuanto a la gestión de equipos se refiere ya que hemos tenido que desarrollar una serie de características que nos han permitido resolver los conflictos de manera pacífica.
- En cuanto a la gestión de código se refiere hemos aprendido aspectos acerca de cómo mantener un repositorio organizado y ordenado, y realizar de manera correcta la resolución de conflictos tras integrar las diferentes funcionalidades desarrolladas.

Las mejoras a realizar en un futuro son las siguientes.

- Implementar el módulo booth dentro del FrontEnd para que a cada usuario le salga que votaciones puede realizar.
- Realizar los test de selenium. Aunque se haya conseguido realizar de manera correcta los test de selenium para algunos de los módulos estos no se han podido integrar en la aplicación completa y en el flujo de integración continua.
- Realizar test de carga con Locust para medir el rendimiento de nuestra aplicación con una alta carga de trabajo.
- Desplegar nuestra aplicación en Render para cumplir con el ciclo completo de integración continua.