

# Modelos de Diseño

## Planificación y Diseño de Sistemas Computacionales

Escuela Ingeniería Informática de Valladolid  
Universidad de Valladolid

14 de diciembre de 2019



Universidad de Valladolid

Arroyo Calle, Adrián  
González Caminero, Juan  
Valdunciel Sánchez, Pablo

## Índice

1. Modelo lógico de la base de datos	4
2. Patrón <i>Front Controller</i> de Django [1] [2]	5
3. Diagrama de clases	5
4. Module Style	6
4.1. Descomposition style . . . . .	6
4.2. Uses style . . . . .	6
4.3. Generalization style . . . . .	6
5. Diagramas de casos de uso	6

## Índice de figuras

1.	Modelo lógico de la base de datos . . . . .	4
2.	Patrón <i>Front Controller</i> de Django . . . . .	5
3.	Diagrama de clases . . . . .	7
4.	Descomposition style . . . . .	8
5.	Uses style . . . . .	9
6.	Generalization style . . . . .	10
7.	Llamada <code>as_view()</code> sobre una subclase de <i>TemplateView</i> - <i>Template rendering</i> de Django . . . . .	11
8.	<i>Add election</i> . . . . .	12
9.	CU1 - Obtener la distribución de escaños . . . . .	13
10.	CU2.A - Introducir los datos de una elección mediante GUI . . . . .	14
11.	CU2.B - Introducir los datos de una elección mediante la subida de un archivo . . . . .	15
12.	CU3 - Modificar parámetros de la configuración por defecto . . . . .	16

# 1. Modelo lógico de la base de datos

El SGBD utilizado ha sido *sqlite3*, un sistema de gestión de bases de datos relacionales que viene incorporado de forma nativa en el framework *Django* utilizado. Los tipos básicos de *sqlite3* son:

1. NULL
2. BLOB
3. INTEGER
4. REAL
5. TEXT

Las fechas, DATE, pueden almacenarse como TEXT, REAL o INTEGER. El tipo VARCHAR permite limitar la longitud de un campo de tipo TEXT.

En la *Figura 1* se muestra el modelo lógico de la base de datos.

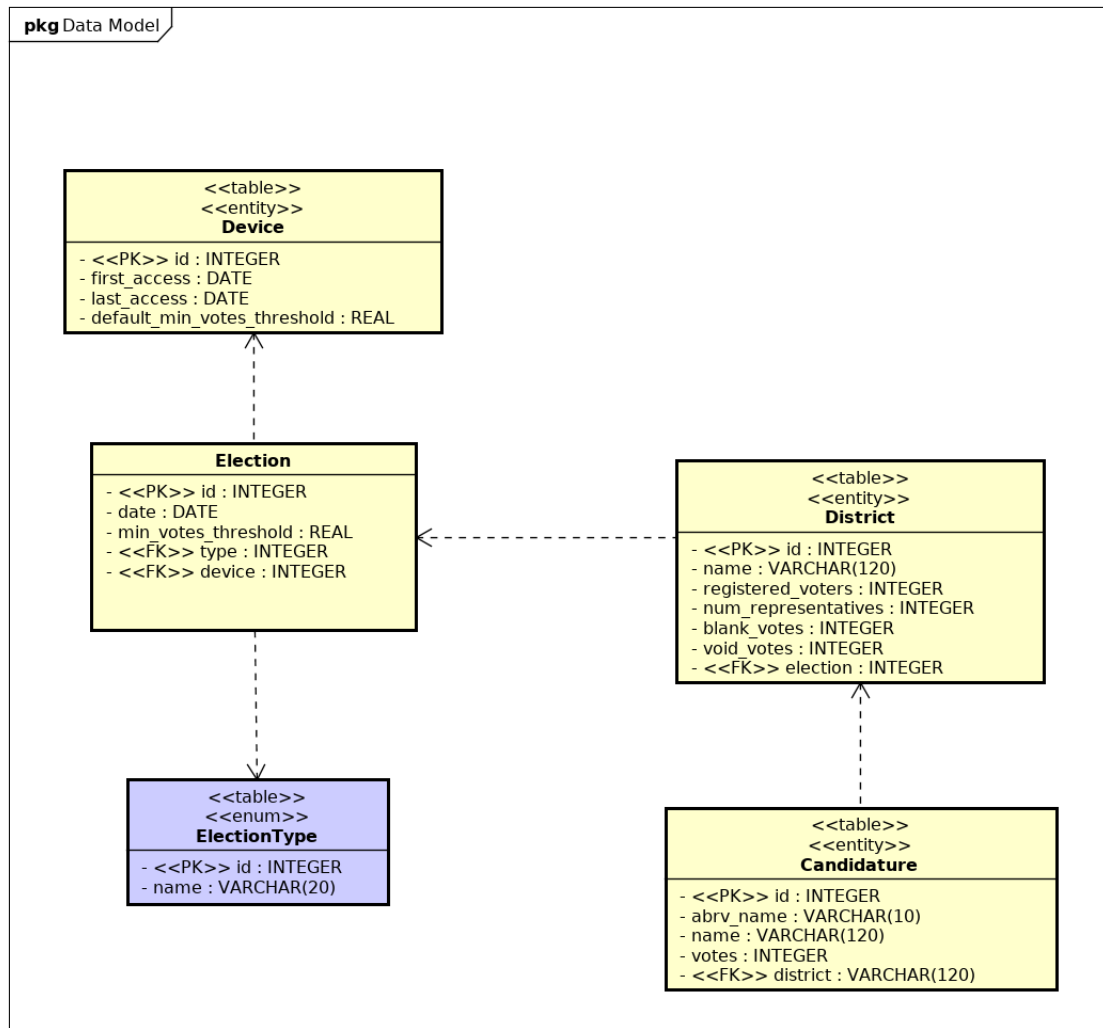


Figura 1: Modelo lógico de la base de datos

## 2. Patrón *Front Controller* de Django [1] [2]

Django utiliza el patrón *Front Controller*, en el que un controlador maneja todas las peticiones web. En Django el *Front Controller* está basado en los ficheros *wsgi.py* y *urls.py* en el que se establece la correspondencia entre las URLs y las vistas. Cuando un usuario solicita una página, este es el algoritmo que Django sigue:

1. Django determina el módulo raíz URLconf que tiene que usar. Por defecto este módulo está en el fichero *urls.py*, aunque se puede cambiar.
2. Django carga el módulo Python y busca la variable *urlpatterns*. Esta variable contiene un conjunto de pares que establecen la correspondencia entre las URLs y las vistas.
3. Django busca una a una todas la URLs y obtiene la que coincide con la solicitada por el usuario.
4. Una vez que ha encontrado la URL correspondiente, Django importa y llama a la Vista correspondiente, que en nuestro caso consiste en una llamada al método *as\_view()* de la clase Vista correspondiente. El método *as\_view()* recibe como argumentos:
  - Una instancia de *HttpRequest* (la petición HTTP).
  - Los argumentos de la URL si esta tiene.
5. Si ninguna de las URLs registradas coincide con la solicitada o se produce una excepción durante el proceso, Django invoca la vista de tratamiento de errores adecuada.

En la *Figura 2* podemos ver el funcionamiento del patrón *Front Controller* en Django.

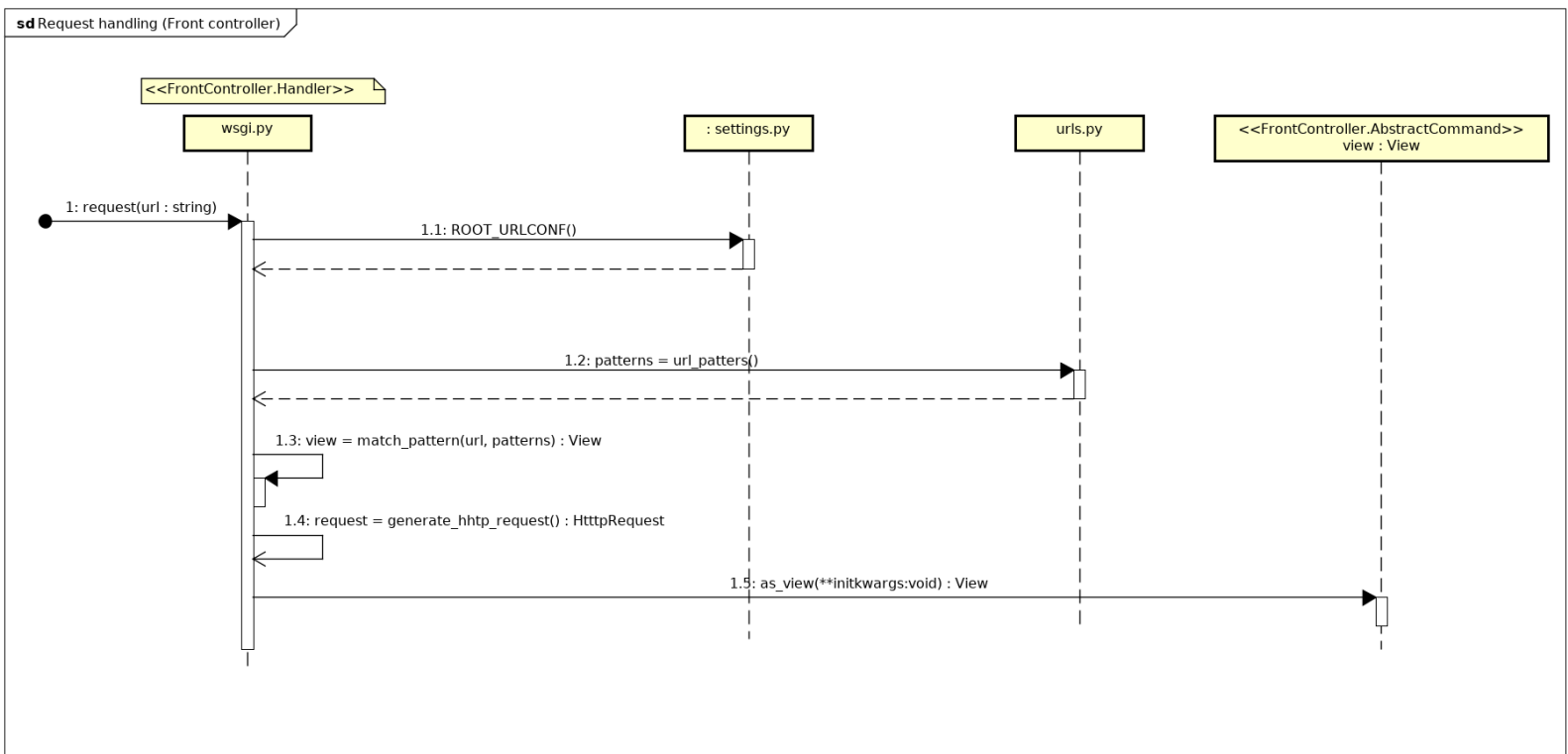


Figura 2: Patrón *Front Controller* de Django

## 3. Diagrama de clases

Ver *Figura 3*.

## 4. Module Style

### 4.1. Descomposition style

Ver *Figura 4*.

### 4.2. Uses style

Ver *Figura 5*.

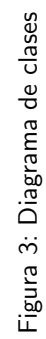
### 4.3. Generalization style

Ver *Figura 6*.

## 5. Diagramas de casos de uso

Los diagramas en los que se modelan los casos de uso son:

- *Template rendering* - Ver *Figura 7*
- *Add election* - Ver *Figura 8*
- CU1 - Obtener la distribución de escaños - Ver *Figura 9*
- CU2 - Introducir datos de unas elecciones
  - Mediante GUI - Ver *Figura 10*
  - Mediante la subida de un archivo - Ver *Figura 11*
- CU3 - Modificar parámetros de la configuración por defecto - Ver *Figura 12*



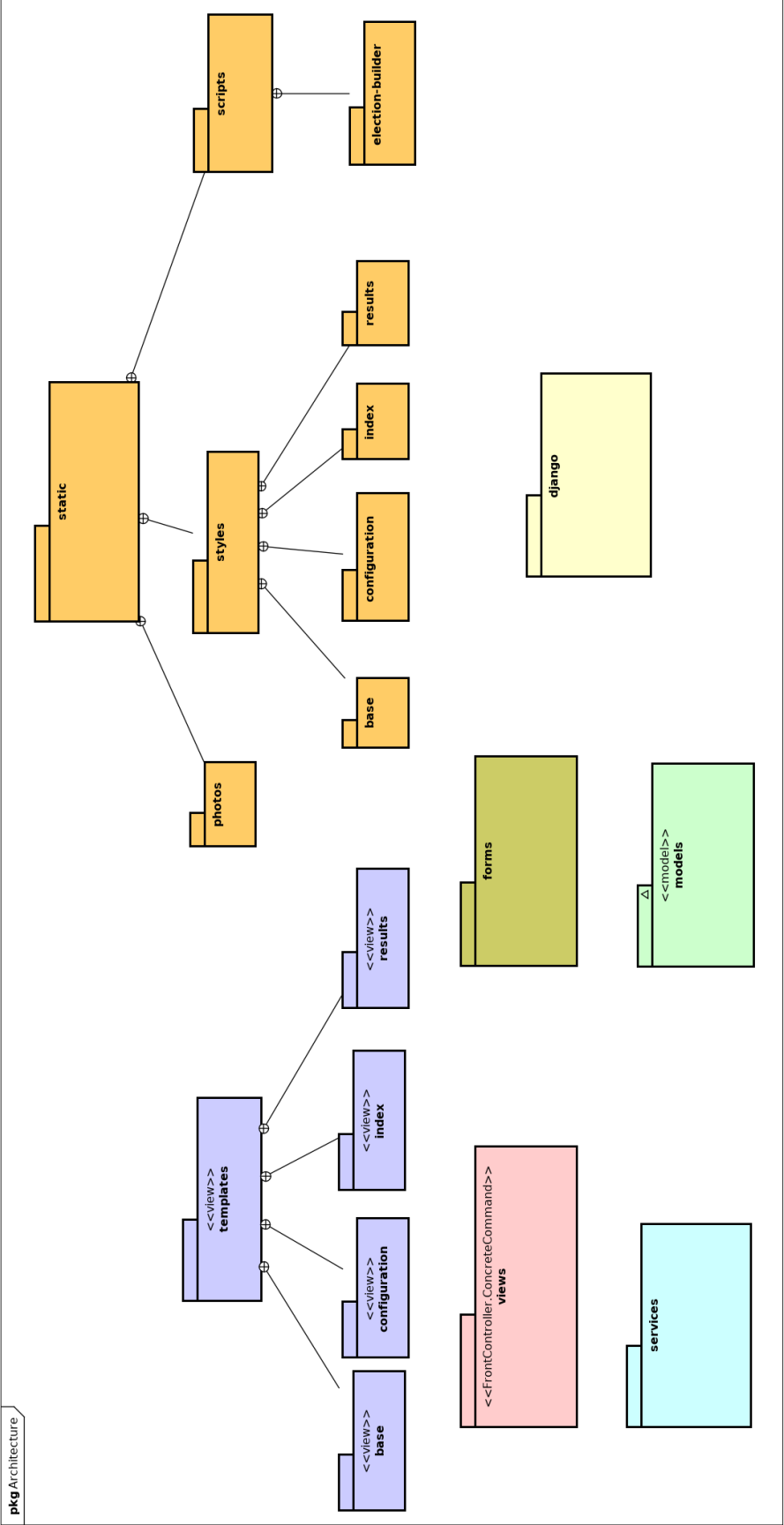


Figura 4: Decomposition style



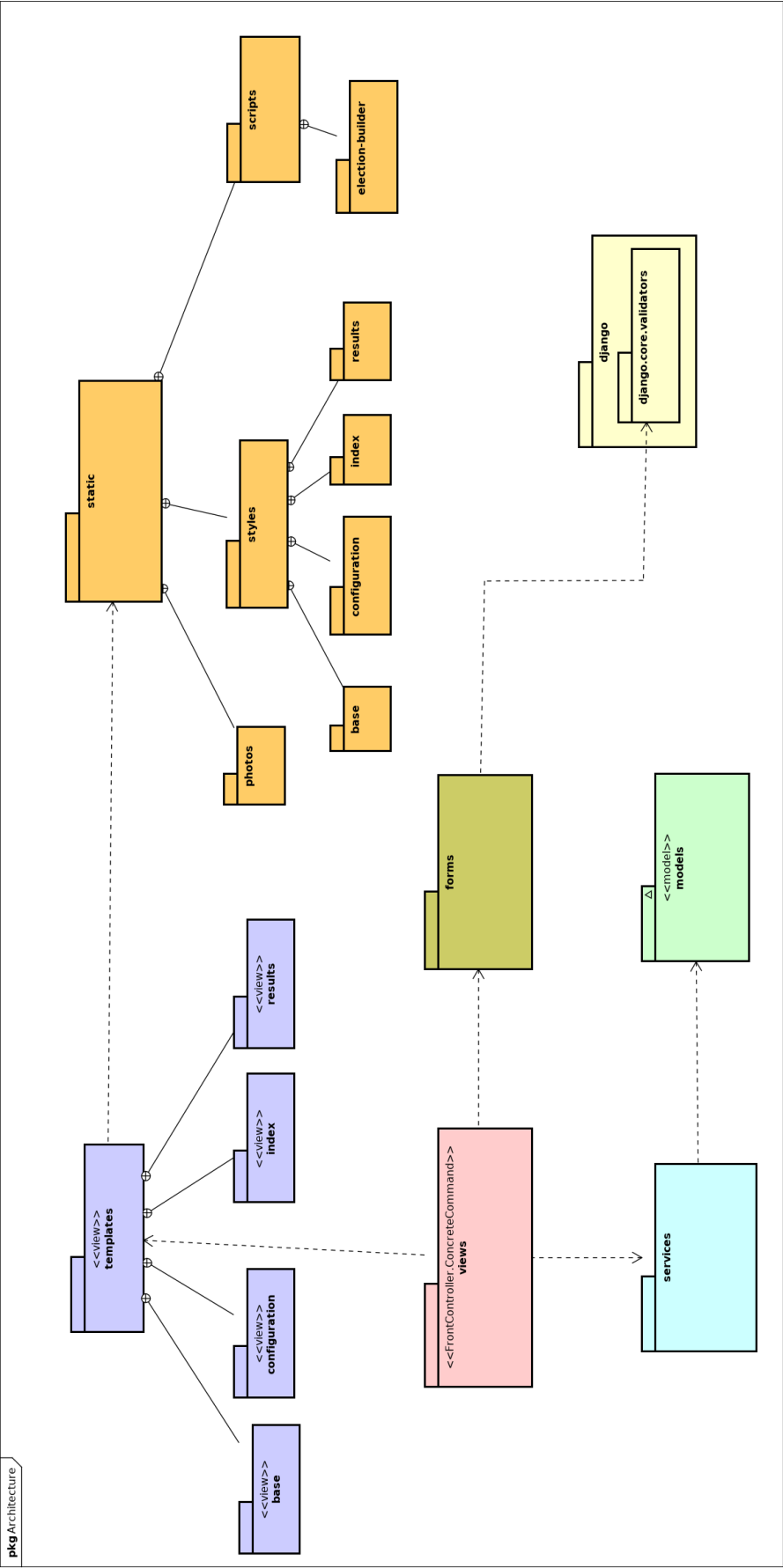


Figura 5: Uses style

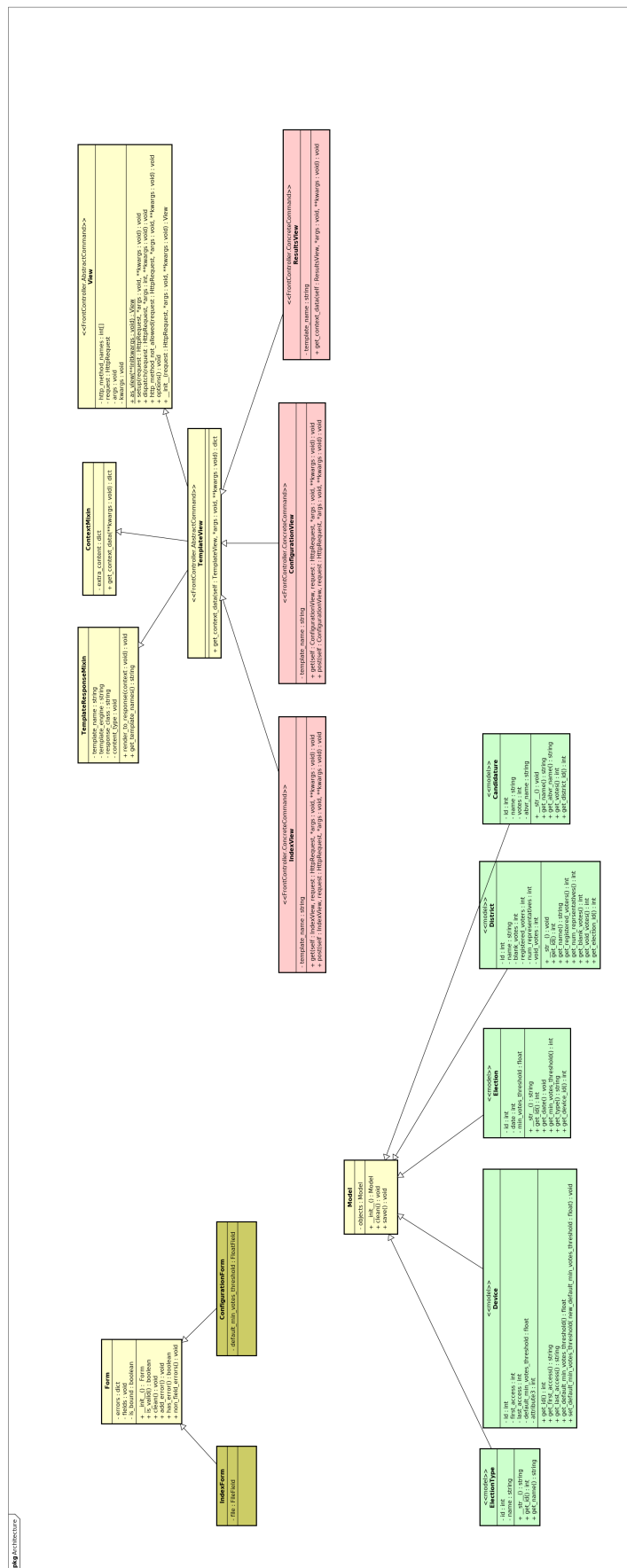


Figura 6: Generalization style

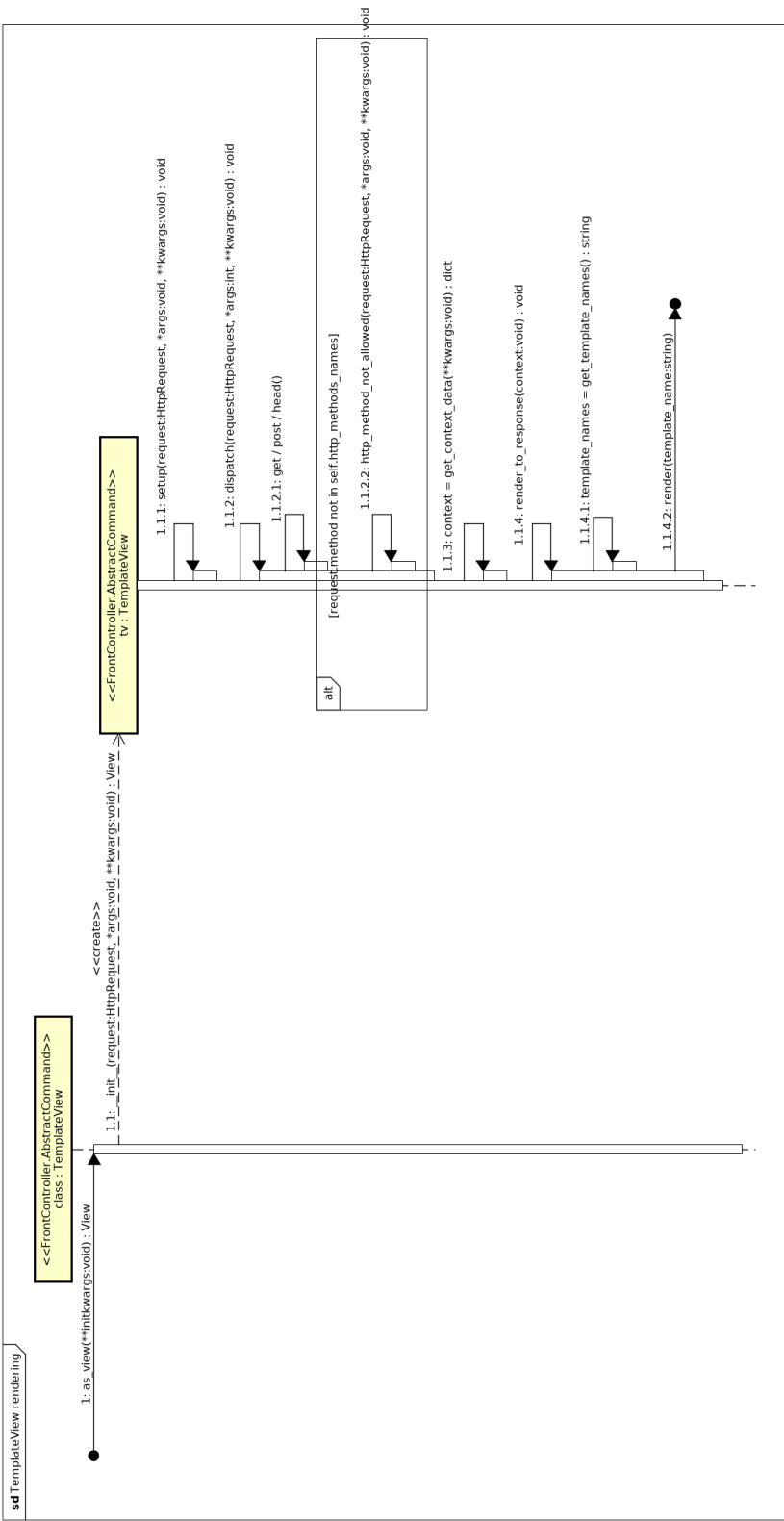


Figura 7: Llamada `as_view()` sobre una subclase de `TemplateView` - *Template rendering* de Django

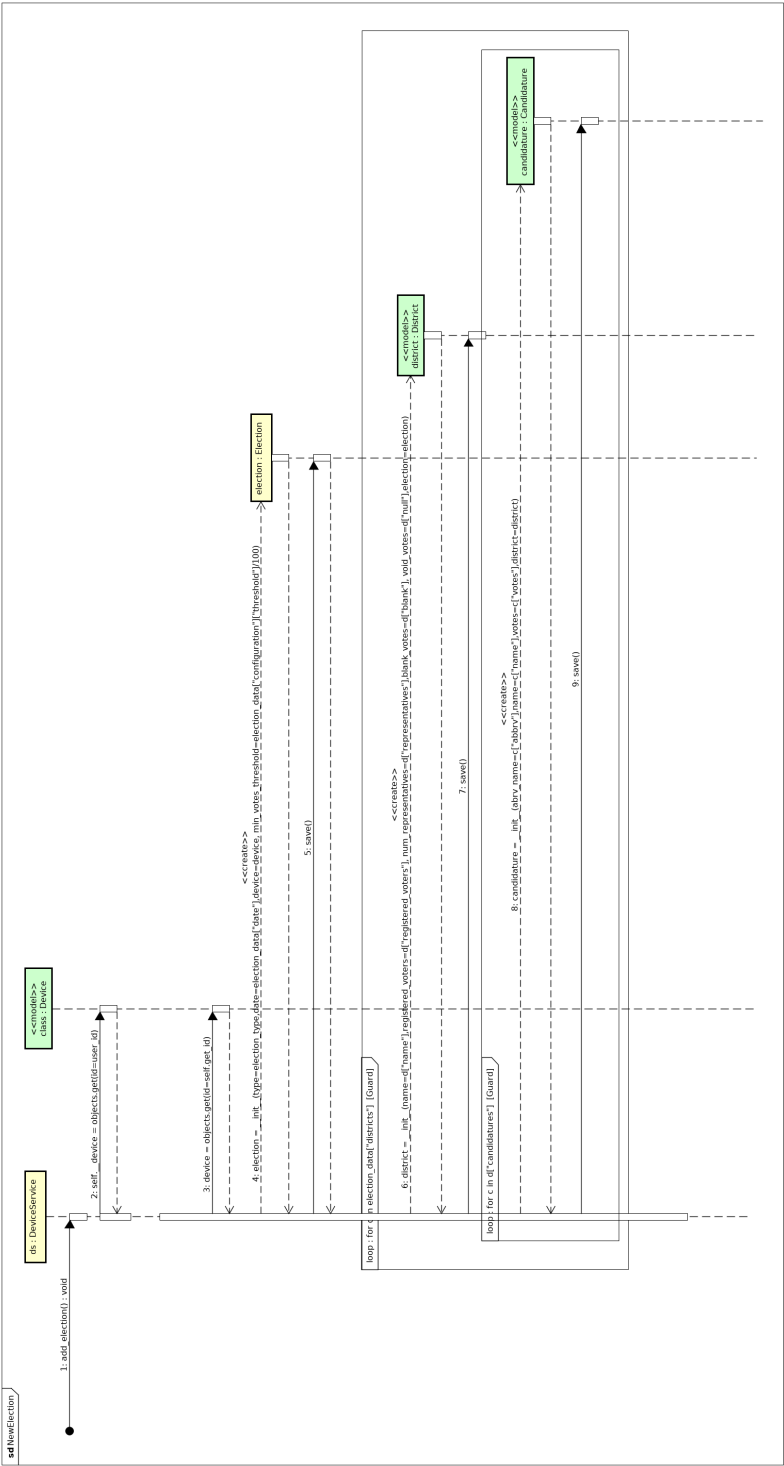


Figura 8: Add election





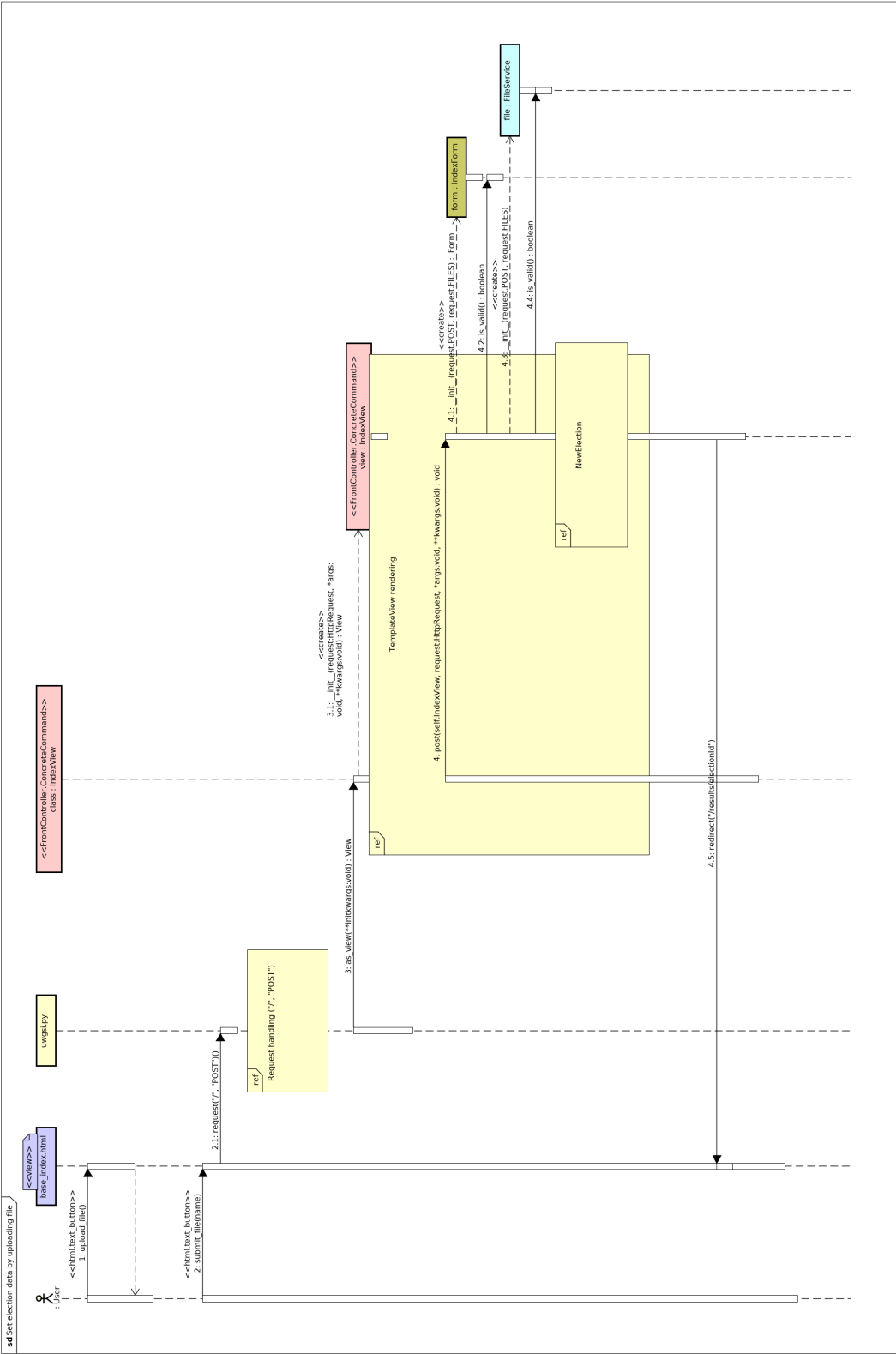


Figura 11: CU2.B - Introducir los datos de una elección mediante la subida de un archivo





## Referencias

- [1] D. documentation, “How django processes a request.” <https://docs.djangoproject.com/es/3.0/topics/http/urls/#how-django-processes-a-request>. [Online; accedido 10 Diciembre, 2019].
- [2] Y. Crespo González-Carvajal, “Planificación y diseño de sistemas computacionales - bloque arquitectura del software.” [https://aulas.inf.uva.es/pluginfile.php/44659/mod\\_resource/content/2/BloqueD1.2.pdf](https://aulas.inf.uva.es/pluginfile.php/44659/mod_resource/content/2/BloqueD1.2.pdf). [Online; accedido 13 Diciembre, 2019].