SERVICIO REST PARA LA ADMINISTRACIÓN PUBLICA

Administración electrónica

JOSEP FERRANDIS JORGE

Índice de contenidos

Índice de ilustraciones

Introducción

En esta memoria se pretende explicar el proceso de creación de un servicio REST el cual está diseñado para ser consumido por la administración pública.

Para ello el trabajo se va a dividir en dos grandes secciones que formarán parte del desarrollo de la memoria, estas dos partes son: planteamiento, donde se describirá la idea del producto que se ha llevado a cabo donde se aportará la información necesaria sobre su valor y cabida del producto dentro de la administración pública. El segundo punto del desarrollo será la implementación, donde se aportará información mucho más detallada del producto.

El objetivo principal de este trabajo ha sido realizar una aplicación web donde se realice de forma sencilla la gestión de empadronamientos y la gestión de pagos y consulta de impuestos. En todo momento se ha tenido en mente la sencillez a la hora de desarrollar tanto el modelo de datos como la explotación y generación de estos.

Con este objetivo se pretende construir un servicio que facilite a las administraciones públicas, más concretamente a ayuntamientos a realizar la gestión de estos dos apartados ya mencionados. Se busca optimizar los tiempos de atención al cliente, simplificar las tareas para los trabajadores de los ayuntamientos y satisfacer las necesidades del ciudadano.

Desarrollo

Como se ha explicado en la introducción este apartado de la memoria se va a dividir en dos partes, el planteamiento y la implementación.

Planteamiento

Con esta aplicación, lo que se pretende es acercar la administración electrónica al ciudadano. Es una aplicación compartida entre varias administraciones públicas, en este caso, Ayuntamientos, que van a tener la posibilidad de consultar datos comunes a las distintas administraciones.

Esta aplicación va a poder ser utilizada en diferentes contextos, ya que se va a tratar de un servicio dentro de la Administración Pública el cual va a poder ser consumido por diferentes tipos de usuarios:

- Servicios web existentes dentro de la administración publica que requieran información referente a empadronamientos o registro de pagos de tasas o impuestos.

- El ciudadano, el cual mediante el proceso de identificación pertinente va a poder consultar sus datos personales referentes a padrones tributarios y al padrón de habitantes.
- Los funcionarios que tienen acceso a esta plataforma con la identificación de empleado público.

Un ejemplo real podría ser la solicitud de alguna subvención, la cual requiere que el ciudadano este empadronado en el municipio y no tenga deudas pendientes.

Hay que destacar que mediante este servicio se mantendrán los datos siempre actualizados, ya la gestión de los datos se va a realizar de forma centralizada, la cual dispondrá de las réplicas y copias de seguridad necesarias.

Es necesario detallar que esta aplicación cumple con los principales objetivos de la interoperabilidad, ya que el objetivo común de todas las administraciones es actuar en beneficio del ciudadano. Además, cumple el requisito de que la información y organización ha sido consensuada y validada por cada una de ellas. La aplicación que se va a desarrollar satisface los tres tipos de interoperabilidad organizativa que existen, ya que: los sistemas tecnológicos interactúan entre sí, *Interoperabilidad Técnica*, la información intercambiada es interpretada por todas las administraciones y ayuntamientos*, Interoperabilidad Semántica*, y, por último, la información se archiva o se guarda en un soporte electrónico capaz de conservar la información con el paso del tiempo, *Interoperabilidad en el tiempo*.

La Administración electrónica, según la Ley 39/2015 de 1 de octubre, del Procedimiento Administrativo Común de las Administraciones Públicas, debe garantizar a los ciudadanos seguridad, y debe actuar en beneficio de éste, para ello debe mejorar la agilidad de los procedimientos, reduciendo los tiempos de tramitación, lo cual, se efectúa también con esta aplicación. Es una ventaja para el ciudadano, poder consultar información sobre sus datos, tantos referentes al padrón de habitantes, como datos relativos a información tributaria de forma rápida y efectiva.

La aplicación dispone de un sistema de seguridad por medio del cual solo puede acceder a su información el propio ciudadano, asegurando la confidencialidad de los datos. Cabe destacar que la aplicación solo podrá ser accesible solicitando un token el cual tan solo tendrá una validez de 15 minutos. La generación de dicho token queda fuera del alcance de este proyecto. Una solución muy válida podría ser integrar este módulo de autentificación con la Identidad Electrónica para las Administraciones (Cl@ve). Por motivos de demostración se ha habilitado una forma de simular la obtención de éste para poder efectuar todas las pruebas pertinentes.

Esta aplicación, también cumple el artículo 41 de la LAECSP de Interoperabilidad de los Sistemas de Información, ya que todas las Administraciones Públicas que utilizan esta plataforma, podrán acceder a la información solicitada por el interesado, puesto que todas disponen de la misma herramienta, aplicando medidas informáticas, tecnológicas, organizativas y de seguridad que garantizan al ciudadano su servicio y confidencialidad.

Implementación

En este punto de la memoria se va a explicar con más detalle el producto que se ha desarrollado. Como se ha comentado en los puntos anteriores se trata de una aplicación web, pero más concretamente se trata de una aplicación back-office la cual utiliza el estándar REST como método de comunicación. Este apartado se va a dividir en tres, en el cual en primer lugar se explicará el modelo de datos utilizado, a continuación, se mostrarán los detalles técnicos y por ultimo se presentarán todas las funcionalidades implementadas.

Modelo de datos

En cuanto al modelo de datos que se ha utilizado en el servicio REST desarrollado, pero antes comentar que se ha optado por un motor de base de datos relacional llamado PostgreSQL. Se ha elegido PostgreSQL ya que se trata de uno de los gestores de base de datos más utilizados en la actualidad y además por ser de código abierto. Estas dos características garantizan el gran soporte que tiene y tendrá esta tecnología.

Explicada la elección de la base de datos se va a explicar el modelo de datos, el cual se muestra en la siguiente imagen:

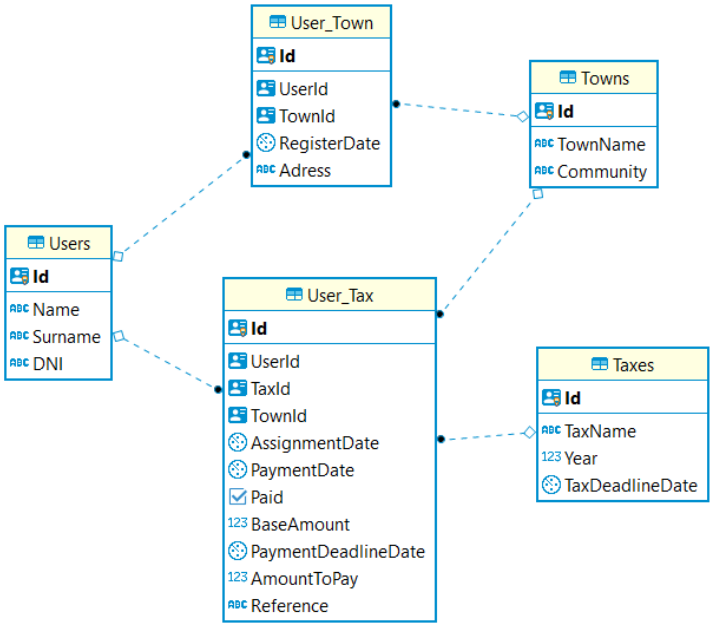

Ilustración 1 Modelo de datos

Como se observa en la imagen anterior, el modelo de datos se ha desarrollado buscando la máxima sencillez, por una parte, se puede observar la relación entre pueblos y usuarios la cual proporcionará la utilización de los datos necesaria para la gestión de empadronamientos. De la misma forma se observa la relación entre usuarios e impuestos que dará soporte a la gestión de los impuestos.

Hay que destacar que las tablas Taxes y Towns, disponen de datos pre-rellenados los cuales se pueden ver en el anexo [1].

Aspectos técnicos

En este punto se van a explicar distintos aspectos técnicos de la aplicación desarrollada:

En primer lugar, en cuanto la arquitectura utilizada, cabe destacar que se ha hecho uso de *Onion Design* el cual permite separar la aplicación en cuatro capas: capa de aplicación, capa de negocio, cada de dominio y capa de datos. La comunicación entre las distintas capas se ha realizado haciendo uso de interfaces. En las siguientes dos imágenes se muestran dos representaciones de dicha arquitectura, donde la primera muestra una visión más general y la segundo una visión más detallada.

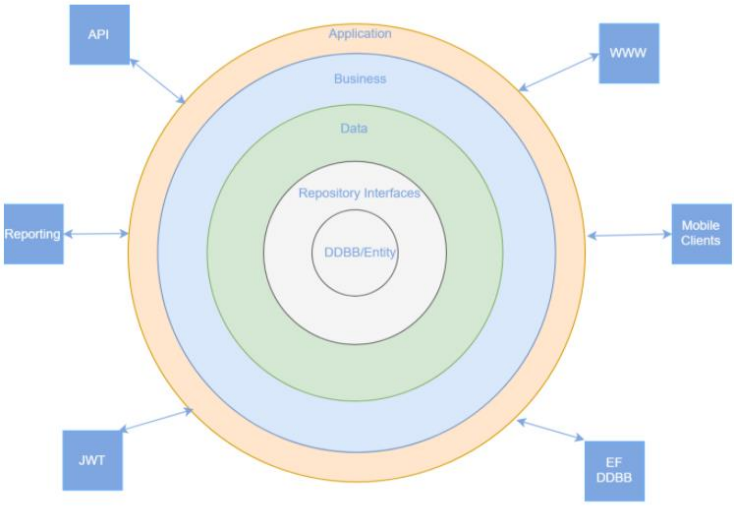

Ilustración 2 Arquitectura Onion Design general

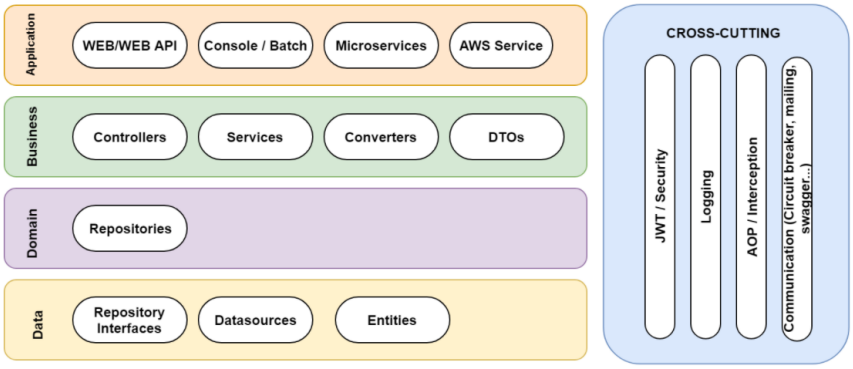

Ilustración 3 Arquitectura Onion Design detallada

La aplicación que se ha desarrollado cuenta con un sistema de control de excepciones customizadas donde resulta muy sencillo para el desarrollador propagar excepciones modificando el mensaje de la excepción y el código de retorno. Además, esta gestión de errores permite tener un código mucho mas limpio ya que no se necesitará ningún bloque de try catch a menos que sea estrictamente necesario.

Cabe destacar que se ha implementado un proceso de migración automática de base de datos, donde la propia aplicación será capaz de generar tanto el esquema como las entidades en la base de datos con la que esté conectada.

El servicio REST dispone de un modulo especifico de gestión de logs, el cual define distintos niveles*: Debug, Error, Fataly e Information;* y distintas formas de almacenar dichos logs: consola, sistema de archivos, base de datos o servidores externos de gestión de logs como ser GrayLog. En este caso se ha decidido mostrarlos simplemente por pantalla en un nivel mínimo de *debug* y almacenarlos en una base de datos SQLite.

Como se ha comentado en el apartado anterior, la aplicación garantiza la seguridad haciendo uso de autentificación mediante token JWT el cual es configurable desde la sección de configuración de la aplicación. Allí se va a poder modificar los parámetros de seguridad donde se podrá elegir distintas opciones, como por ejemplo la utilización de un secretkey o un certificado para la generación del token o el tiempo de validez del mismo.

En lo que al tratamiento de datos respecta, comentar que se ha realizado una normalización de los datos a la hora de ser introducidos desde el exterior, de esta forma se garantiza el correcto funcionamiento de la aplicación.

Se ha hecho uso de la herramienta Swagger basada en el estándar OpenAPI, la cual permite generar un cliente web en tiempo de ejecución. Esta herramienta es muy útil para realizar pruebas rápidas, pero a su vez para generar documentación sobre la API desarrollada, ya que dispone de toda la información referente a los endpoints, objetos y parámetros de entrada.

Funcionalidad

En este punto de la implantación se va a explicar todas las funcionalidades que se han desarrollado. Cada funcionalidad está recogida en un endpoint o punto de acceso a la API REST desarrollada:

Dar de alta a un ciudadano en el sistema

Esta funcionalidad permite crear ciudadanos nuevos en el sistema, el endpoint que da soporte a esta función es: “/Inhabitants/createuser”, se trata de un método POST y los parámetros de entrada en formato JSON son:

```
{
   "name":"string",
   "surname":"string",
   "dni":"string"
}
```

Tabla de códigos de retorno:

CÓDIGO	DESCRIPCIÓN
201	El ciudadano se ha creado correctamente
400	Existe un error en el formato de alguno de los campos que se han enviado
401	Acceso no autorizado
409	El ciudadano ya ha sido registrado con anterioridad
500	Error en el servidor

Dar de baja a un ciudadano en el sistema

Esta funcionalidad permite borrar ciudadanos en el sistema, esto borrara la relación que existe entre el usuario y el municipio y se borraran los registros de impuestos que pertenezcan a este ciudadano.
El endpoint que da soporte a esta función es: "/Inhabitants/deleteuser", se trata de un método DELETE y los parámetros de entrada son "name" y "surname".

Tabla de códigos de retorno:

CÓDIGO	DESCRIPCIÓN
200	El ciudadano se ha borrado correctamente
400	Existe un error en el formato de alguno de los campos que se han enviado
401	Acceso no autorizado
404	El ciudadano no existe en el sistema
500	Error en el servidor

Empadronar a un ciudadano

Esta funcionalidad permite empadronar a un ciudadano en un municipio determinado. El ciudadano tan solo podrá estar empadronado en un municipio así que en el momento que se empadrona en uno nuevo el registro anterior se eliminará.
El endpoint que da soporte a esta función es: "/Inhabitants/registerusertown", se trata de un método POST y los parámetros de entrada en formato JSON son:

```
{
   "name":"string",
   "surname":"string",
   "townName":"string",
   "adress":"string"
}
```

Tabla de códigos de retorno:

CÓDIGO	DESCRIPCIÓN
201	El ciudadano se ha empadronado correctamente
400	Existe un error en el formato de alguno de los campos que se han enviado
401	Acceso no autorizado
404	El ciudadano o el pueblo no existe en el sistema
500	Error en el servidor

Comprobar que un ciudadano pertenece a un pueblo

Esta funcionalidad va a permitir saber si un usuario está empadronado en un municipio
El endpoint que da soporte a esta función es: "/Inhabitants/userbelongstown", se trata de un método POST y los parámetros de entrada en formato JSON son:

```
{
   "name":"string",
   "surname":"string",
   "townName":"string"
}
```

Tabla de códigos de retorno:

CÓDIGO	DESCRIPCIÓN
200	El ciudadano pertenece al municipio
400	Existe un error en el formato de alguno de los campos que se han enviado
401	Acceso no autorizado
404	El ciudadano o el pueblo no existe en el sistema
409	El ciudadano no está empadronado en dicho municipio
500	Error en el servidor

Asignar un impuesto a un ciudadano

Mediante esta funcionalidad se va a permitir asignar un impuesto a un ciudadano. En esta asignación se establecerá en qué municipio debe realizar el pago, el nombre del impuesto, la referencia de éste y la cantidad a abonar.
El endpoint que da soporte a esta función es: "/Inhabitants/asigntaxuser", se trata de un método POST y los parámetros de entrada en formato JSON son:

```
{
   "name":"string",
   "surname":"string",
   "townName":"string",
   "taxName":"string",
   "taxYear":0,
   "baseAmount":0,
   "reference":"string"
}
```

Tabla de códigos de retorno:

CÓDIGO	DESCRIPCIÓN
201	Se le ha asignado correctamente el impuesto al ciudadano
400	Existe un error en el formato de alguno de los campos que se han enviado
401	Acceso no autorizado
404	El ciudadano o el impuesto no existe en el sistema
409	Impuesto previamente asignado al ciudadano
500	Error en el servidor

Marcar como pagado un impuesto asignado a un ciudadano

Mediante esta funcionalidad se va a permitir marcar como pagado un impuesto que previamente había sido asignado a un ciudadano. Esta funcionalidad va a comprobar si la fecha en la que se realiza el pago se encuentra dentro del periodo estipulado. En caso de que dicha fecha haya expirado deberá de obtener la información actualizada en la cual se recalculará el impuesto a pagar y la fecha límite de pago (dicha funcionalidad se describe en el siguiente punto).
El endpoint que da soporte a esta función es: "/Inhabitants/paytaxuser", se trata de un método POST y los parámetros de entrada en formato JSON son:


```
{
   "name":"string",
   "surname":"string",
   "taxName":"string",
   "taxYear":0,
   "reference":"string"
}
```


Tabla de códigos de retorno:

CÓDIGO	DESCRIPCIÓN
200	Se ha marcado impuesto como pagado
400	• Existe un error en el formato de alguno de los campos que se han enviado • El usuario ya ha realizado el pago anteriormente
401	Acceso no autorizado
404	El ciudadano o el impuesto no existe en el sistema
409	Fecha de pago expirada
500	Error en el servidor

Obtener la lista de impuestos actualizada de un ciudadano

Mediante esta funcionalidad, se va a poder obtener toda la información sobre los impuestos de un ciudadano, se comprobará que todos los pagos no realizados estén expirados o fuera de plazo. En caso de estar fuera de plazo se comprobará la cantidad de días que han pasado aplicándose un recargo de 5% por cada 10 días transcurridos, llegando a un máximo de 20% donde se asignará un año para pagar.

El endpoint que da soporte a esta función es: "/Inhabitants/getupdatedtaxuserinfo", se trata de un método GET y los parámetros de entrada son "name" y "surname".

Tabla de códigos de retorno:

CÓDIGO	DESCRIPCIÓN
200	Se ha obtenido la información de forma satisfactoria
400	Existe un error en el formato de alguno de los campos que se han enviado
401	Acceso no autorizado
404	El ciudadano no existe en el sistema
500	Error en el servidor

El objeto de retorno que contiene la información correspondiente a los impuestos de un ciudadano es el siguiente (Caso de código 200):


```
{
   "name":"string",
   "surname":"string",
   "taxes":[
      {
         "townName":"string",
         "taxName":"string",
         "taxYear":0,
         "amountToPay":0,
         "reference":"string",
         "paymentDeadLine":"2021-01-17T19:48:11.293Z",
         "paid":true,
         "surcharge":true
      }
   ]
}
```

Conclusiones

Es necesario actuar para transformar la administración y avanzar hacia un modelo cada vez mas interoperable, en beneficio del ciudadano. Las Administraciones publicas y otras entidades, deben tener la capacidad y la voluntad de actuar de forma conjunta. La cooperación entre ellas es esencial para proporcionar los servicios a los ciudadanos y garantizarles sus derechos. Esta cooperación requiere que haya una fluidez, lo cual se consigue con la interoperabilidad siempre con la máxima seguridad y eficiencia.

Este es el objetivo de la aplicación que se ha desarrollado, aportar al ciudadano la máxima confidencialidad y protección de datos y sobre todo comodidad y simplicidad para la realización de gestiones de forma rápida, efectiva y segura.

Como se ha visto a largo de esta memoria, el servicio desarrollado cumple con todos estos principios, ya que va a ofrecer la posibilidad de realizar las gestiones referentes al padrón de habitantes y a los padrones tributarios de una forma mucho más rápida y efectiva que de la forma tradicional donde se debería acceder presencialmente a las oficinas de atención al ciudadano. Además, también cumple con un mecanismo de seguridad que garantiza las identidades de los usuarios que acceden al servicio, como se ha comentado mediante el uso de un token con una validez limitada.

Mediante este tipo de aplicaciones se ofrece la posibilidad al ciudadano de realizar trámites de forma electrónica, lo cual implica una reducción considerable en los tiempos de espera y simplificación de los procesos administrativos. Por otro la administración podrá atender las solicitudes del ciudadano de una forma más rápida y segura.

Anexos:

[1] Datos de impuestos y pueblos en la base de datos:

Como se ha comentado las tablas de pueblos e impuestos están pre-rellenadas. Esto se realiza durante el proceso de la creación de la base de datos.

Los impuestos que están en el sistema son:

- ibi
- vehículos
- basura

En los cuales se ha añadido entrada para el año 2021, 2022 y 2023.
Hay que destacar que el impuesto de basura del año 2021 se ha preparado para que la fecha de pago sea anterior a la fecha actual y se pueda observar cómo funcionan los recargos.

Pueblos que están en el sistema son:

- ademuz
- ador
- adzaneta de albaida
- agullent
- alacuás
- albaida
- albal
- albalat de la ribera
- albalat dels sorells
- albalat dels tarongers
- alberique
- alborache
- alboraya
- albuixech
- alcácer
- alcántara de júcar
- alcira
- alcublas
- alcudia de crespins
- aldaya
- alfafar
- alfahuir
- alfara de la baronía
- alfara del patriarca
- alfarp
- alfarrasí
- algar de palancia
- algemesí
- algimia de alfara
- alginet
- almácera
- almiserat
- almoines
- almusafes
- alpuente
- alquería de la condesa
- andilla
- anna
- antella
- aras de los olmos
- ayelo de malferit
- ayelo de rugat
- ayora
- barcheta
- bárig
- bélgida
- bellreguart
- bellús
- benagéber
- benaguacil
- benavites
- benegida
- benetúser
- beniarjó
- beniatjar
- benicolet
- benicull
- benifairó de la valldigna
- benifairó de los valles
- benifayó
- beniflá
- benigánim
- benimodo
- benimuslem
- beniparrell
- benirredrá
- benisanó
- benisoda
- benisuera
- bétera
- bicorp
- bocairente
- bolbaite
- bonrepós y mirambell
- bufali
- bugarra
- buñol
- burjasot
- calles
- camporrobles
- canals

- canet de berenguer
- carcagente
- cárcer
- carlet
- carrícola
- casas altas
- casas bajas
- casinos
- castellón de rugat
- castellonet
- castielfabib
- catadau
- catarroja
- caudete de las fuentes
- cerdá
- chella
- chelva
- chera
- cheste
- chirivella
- chiva
- chulilla
- cofrentes
- corbera
- cortes de pallás
- cotes
- cuart de les valls
- cuart de poblet
- cuartell
- cuatretonda
- cullera
- daimuz
- domeño
- dos aguas
- el puig
- emperador
- enguera
- énova
- estivella
- estubeny
- faura
- favareta
- fontanares
- fortaleny
- foyos
- fuente encarroz
- fuente la higuera
- fuenterrobles
- gabarda
- gandía
- gátova
- genovés
- gestalgar
- gilet
- godella
- godelleta
- guadasequies
- guadasuar
- guardamar de la safor
- higueruelas
- jalance
- jaraco
- jarafuel
- játiva
- jeresa
- la alcudia
- la eliana
- la granja de la costera
- la yesa
- líria
- llanera de ranes
- llaurí
- llombay
- llosa de ranes
- loriguilla
- losa del obispo
- luchente
- lugar nuevo de fenollet
- lugar nuevo de la corona
- lugar nuevo de san jerónimo
- macastre
- manises
- manuel
- marines
- masalavés
- masalfasar
- masamagrell
- masanasa
- meliana
- millares
- miramar
- mislata
- mogente
- moncada
- montaberner
- montesa
- montichelvo
- montroy
- montserrat
- museros
- náquera
- navarrés
- novelé
- oliva
- ollería
- olocau
- onteniente
- otos
- paiporta
- palma de gandía
- palmera
- palomar
- paterna
- pedralba
- petrés
- picaña
- picasent
- piles
- pinet
- poliñá de júcar
- potríes
- puebla de farnals
- puebla del duc
- puebla de san miguel
- puebla de vallbona
- puebla larga
- puzol
- quesa
- rafelbuñol
- rafelcofer
- rafelguaraf
- ráfol de salem
- real
- real de gandía
- requena
- ribarroja del turia
- riola
- rocafort
- rotglá y corberà
- rótova

- rugat
- sagunto
- salem
- san antonio de benagéber
- san juan de énova
- sedaví
- segart
- sellent
- sempere
- señera
- serra
- siete aguas
- silla
- simat de valldigna
- sinarcas
- sollana
- sot de chera
- sueca
- sumacárcel
- tabernes blanques
- tabernes de valldigna
- teresa de cofrentes
- terrateig
- titaguas
- torrebaja
- torrella
- torrente
- torres torres
- tous
- tuéjar
- turís
- utiel
- valencia
- vallada
- vallanca
- vallés
- venta del moro
- villalonga
- villamarchante
- villanueva de castellón
- villar del arzobispo
- villargordo del cabriel
- vinalesa
- yátova
- zarra