MODELO DE DATOS DEL SISTEMA

Plataforma Digital para la Recepción y Gestión de Viviendas - TECHO Chile

**Desarrollador:** Rodrigo Faúndez  
**Proyecto:** APT Fase 2 - Sistema TECHO Chile  
**Fecha:** Octubre de 2025

## 1. RESUMEN EJECUTIVO

Este documento detalla la estructura completa del modelo de datos implementado para la Plataforma Digital de TECHO Chile. El sistema está organizado en **6 aplicaciones Django** interconectadas que gestionan de manera integral el ciclo completo de viviendas sociales, desde la planificación hasta el seguimiento postventa. El modelo comprende **27 entidades principales** con más de **150 campos** y **35 relaciones** entre tablas.

## 2. ARQUITECTURA DEL MODELO DE DATOS

### 2.1 Organización por Aplicaciones Django

El modelo de datos sigue el patrón de **aplicaciones especializadas** de Django, permitiendo modularidad y mantenibilidad:

* **core** - Sistema base de usuarios, roles y configuración
* **proyectos** - Gestión de proyectos habitacionales y beneficiarios
* **reportes** - Documentación oficial y actas de entrega
* **incidencias** - Sistema de seguimiento de problemas y observaciones
* **ficha\_postventa** - Evaluaciones y seguimiento post-entrega
* **django.contrib.auth** - Sistema de autenticación y permisos

### 2.2 Principios de Diseño Aplicados

**Normalización de Base de Datos:** - **Primera Forma Normal (1NF):** Todos los campos contienen valores atómicos - **Segunda Forma Normal (2NF):** Eliminación de dependencias parciales - **Tercera Forma Normal (3NF):** Eliminación de dependencias transitivas

**Integridad Referencial:** - Foreign Keys con on\_delete strategies apropiadas - Unique constraints en campos de identificación únicos - Composite unique constraints para códigos por contexto

**Escalabilidad y Performance:** - Índices en campos de búsqueda frecuente - JSONField para datos flexibles (elementos\_disponibles) - Campos de auditoría (fecha\_creacion, fecha\_actualizacion)

## 3. APLICACIÓN CORE - SISTEMA BASE

### 3.1 Modelo CoreUsuario (Usuario Principal)

**class** CoreUsuario(AbstractUser):  
 email = models.CharField(unique=True, max\_length=254) *# Login principal*  
 nombre = models.CharField(max\_length=100)  
 apellido\_paterno = models.CharField(max\_length=100)  
 apellido\_materno = models.CharField(max\_length=100, blank=True, null=True)  
 rut = models.CharField(unique=True, max\_length=12, blank=True, null=True)  
 telefono = models.CharField(max\_length=15)  
 empresa = models.CharField(max\_length=200)

*# Relaciones geográficas*  
 region = models.ForeignKey(CoreRegion, on\_delete=models.DO\_NOTHING)  
 comuna = models.ForeignKey(CoreComuna, on\_delete=models.DO\_NOTHING)  
   
 *# Relaciones organizacionales*  
 rol = models.ForeignKey(CoreRol, on\_delete=models.DO\_NOTHING)  
 constructora = models.ForeignKey(CoreConstructora, on\_delete=models.DO\_NOTHING)

**Características técnicas:** - Extiende AbstractUser de Django para autenticación completa - RUT único como identificador secundario para usuarios chilenos - Relaciones con ubicación geográfica para filtros territoriales - Vínculo con constructoras para usuarios externos

### 3.2 Sistema de Roles y Permisos

**CoreRol - Definición de Roles del Sistema:** - **Superuser:** Administrador completo del sistema - **Familia:** Beneficiarios que pueden crear observaciones y ver su vivienda - **Constructor:** Acceso a viviendas asignadas y carga de evidencias - **SERVIU:** Solo lectura y descarga de reportes estadísticos - **Coordinador TECHO:** Gestión completa de proyectos y seguimiento

### 3.3 Modelo Geográfico (Region-Comuna)

**class** CoreRegion(models.Model):  
 nombre = models.CharField(unique=True, max\_length=100) *# "Región Metropolitana"*  
 codigo = models.CharField(unique=True, max\_length=10) *# "13"*  
 activo = models.BooleanField(default=True)  
  
**class** CoreComuna(models.Model):  
 nombre = models.CharField(max\_length=100) *# "Santiago"*  
 codigo = models.CharField(max\_length=10) *# "13101"*  
 region = models.ForeignKey(CoreRegion, on\_delete=models.CASCADE)  
   
 **class** Meta:  
 unique\_together = (('nombre', 'region'),) *# Comuna única por región*

**Funcionalidades implementadas:** - Filtros geográficos para reportes y dashboards - Asignación territorial de usuarios y proyectos - Estadísticas por división administrativa

### 3.4 Gestión de Constructoras

**class** CoreConstructora(models.Model):  
 nombre = models.CharField(unique=True, max\_length=150)  
 rut = models.CharField(max\_length=15, blank=True, null=True)  
 direccion = models.CharField(max\_length=255)  
 contacto = models.CharField(max\_length=100) *# Persona responsable*  
 telefono = models.CharField(max\_length=30)  
 email = models.CharField(max\_length=254)  
   
 *# Ubicación de la constructora*  
 region = models.ForeignKey(CoreRegion, on\_delete=models.DO\_NOTHING)  
 comuna = models.ForeignKey(CoreComuna, on\_delete=models.DO\_NOTHING)

## 4. APLICACIÓN PROYECTOS - GESTIÓN HABITACIONAL

### 4.1 Modelo ProyectosProyecto (Proyectos Habitacionales)

**class** ProyectosProyecto(models.Model):  
 codigo = models.CharField(unique=True, max\_length=20)*# "TECHO-2025-001"*  
 siglas = models.CharField(max\_length=10) *# "T25-001"*  
 nombre = models.CharField(max\_length=200) *# "Villa Esperanza"*  
 fecha\_entrega = models.DateField() *# Entrega planificada*  
 fecha\_termino\_postventa = models.DateField(blank=True, null=True)

*# Fin seguimiento*  
   
*# Coordenadas GPS del proyecto*  
 coordenadas\_s = models.DecimalField(max\_digits=10, decimal\_places=5) *# Latitud Sur*  
 coordenadas\_w = models.DecimalField(max\_digits=10, decimal\_places=5) *# Longitud Oeste*  
   
*# Relaciones*  
 comuna = models.ForeignKey(CoreComuna, on\_delete=models.DO\_NOTHING)  
 region = models.ForeignKey(CoreRegion, on\_delete=models.DO\_NOTHING)  
 constructora = models.ForeignKey(CoreConstructora, on\_delete=models.DO\_NOTHING)  
 creado\_por = models.ForeignKey(CoreUsuario, on\_delete=models.DO\_NOTHING)

**Funcionalidades clave:** - Seguimiento temporal completo del proyecto - Ubicación GPS para mapping y logística - Trazabilidad de creación y modificaciones - Relación con constructora responsable

### 4.2 Modelo ProyectosVivienda (Viviendas Individuales)

**class** ProyectosVivienda(models.Model):  
 codigo = models.CharField(max\_length=10) *# "V-001", único por proyecto*  
 familia\_beneficiaria = models.CharField(max\_length=200) *# Nombre representativo*  
 estado = models.CharField(max\_length=20) *# "entregada", "construccion", etc.*  
   
 *# Fechas del ciclo de vida*  
 fecha\_entrega = models.DateField(blank=True, null=True)  
 fecha\_inicio\_postventa = models.DateField(blank=True, null=True)  
 fecha\_termino\_postventa = models.DateField(blank=True, null=True)  
   
 *# Relaciones principales*  
 proyecto = models.ForeignKey(ProyectosProyecto, on\_delete=models.CASCADE)  
 tipologia = models.ForeignKey(ProyectosTipologiaVivienda, on\_delete=models.DO\_NOTHING)  
 beneficiario = models.OneToOneField(ProyectosBeneficiario, on\_delete=models.SET\_NULL)  
   
 **class** Meta:  
 unique\_together = (('proyecto', 'codigo'),) *# Código único por proyecto*

### 4.3 Sistema de Tipologías de Vivienda

**class** ProyectosTipologiaVivienda(models.Model):  
 codigo = models.IntegerField(unique=True) *# 1, 2, 3...*  
 nombre = models.CharField(max\_length=100) *# "Casa Tipo A"*  
 descripcion = models.TextField() *# Especificaciones técnicas*  
 metros\_cuadrados = models.DecimalField(max\_digits=10, decimal\_places=5)  
 numero\_ambientes = models.PositiveIntegerField() *# Habitaciones + living + cocina*  
 tipo\_estructura = models.CharField(max\_length=100) *# "Madera", "Metalcom", etc.*  
  
**class** ProyectosRecinto(models.Model):  
 codigo = models.CharField(max\_length=10) *# "HAB1", "COCINA"*  
 nombre = models.CharField(max\_length=100) *# "Habitación Principal"*  
 descripcion = models.TextField() *# Especificaciones del recinto*  
 elementos\_disponibles = models.JSONField() *# Lista flexible de elementos*  
 tipologia = models.ForeignKey(ProyectosTipologiaVivienda, on\_delete=models.CASCADE)  
   
 **class** Meta:  
 unique\_together = (('tipologia', 'codigo'),) *# Código único por tipología*

**Ejemplo de elementos\_disponibles:**

{  
 "elementos\_fijos": ["ventana", "puerta", "interruptor\_luz"],  
 "instalaciones": ["punto\_electricidad", "toma\_agua"],  
 "acabados": ["piso\_ceramica", "muro\_pintado"]  
}

### 4.4 Gestión de Beneficiarios

**class** ProyectosBeneficiario(models.Model):  
 nombre = models.CharField(max\_length=100)  
 apellido\_paterno = models.CharField(max\_length=100)  
 apellido\_materno = models.CharField(max\_length=100, blank=True, null=True)  
 rut = models.CharField(unique=True, max\_length=12, blank=True, null=True)  
 email = models.CharField(max\_length=254, blank=True, null=True)  
  
**class** ProyectosTelefono(models.Model):  
 numero = models.CharField(max\_length=20) *# +56912345678*  
 beneficiario = models.ForeignKey(ProyectosBeneficiario, on\_delete=models.CASCADE)  
 activo = models.BooleanField(default=True) *# Para números obsoletos*

## 5. APLICACIÓN REPORTES - DOCUMENTACIÓN OFICIAL

### 5.1 Modelo ReportesActaRecepcion (Actas de Entrega)

**class** ReportesActaRecepcion(models.Model):  
 numero\_acta = models.CharField(unique=True, max\_length=50) *# "ACTA-2025-001"*  
 fecha\_entrega = models.DateTimeField() *# Momento exacto de entrega*  
 lugar\_entrega = models.CharField(max\_length=200) *# Dirección específica*  
   
 *# Responsables TECHO*  
 representante\_techo = models.CharField(max\_length=100) *# Nombre del coordinador*  
 cargo\_representante = models.CharField(max\_length=100) *# "Coordinador Regional"*  
 rut\_representante = models.CharField(max\_length=12)  
 telefono\_representante = models.CharField(max\_length=20)  
   
 *# Información constructora*  
 jefe\_construccion = models.CharField(max\_length=100, blank=True, null=True)  
 numero\_voluntarios = models.PositiveIntegerField() *# Voluntarios participantes*  
   
 *# Especificaciones técnicas de la vivienda entregada*  
 superficie\_construida = models.DecimalField(max\_digits=10, decimal\_places=5) *# m²*  
 numero\_ambientes = models.PositiveIntegerField()  
 tipo\_estructura = models.CharField(max\_length=20) *# "Madera", "Metalcom"*  
   
 *# Servicios básicos al momento de entrega*  
 tiene\_electricidad = models.BooleanField()  
 tiene\_agua\_potable = models.BooleanField()  
 tiene\_alcantarillado = models.BooleanField()  
   
 *# Estado de la entrega*  
 estado\_estructura = models.BooleanField() *# Estructura completa*  
 estado\_instalaciones = models.BooleanField() *# Instalaciones funcionando*  
 entregado\_beneficiario = models.BooleanField() *# Entregado efectivamente*  
 plazo\_correcciones = models.PositiveIntegerField() *# Días para correcciones*  
   
 *# Relaciones*  
 vivienda = models.OneToOneField(ProyectosVivienda, on\_delete=models.CASCADE)  
 beneficiario = models.ForeignKey(ProyectosBeneficiario, on\_delete=models.DO\_NOTHING)  
 proyecto = models.ForeignKey(ProyectosProyecto, on\_delete=models.DO\_NOTHING)

### 5.2 Composición Familiar en Actas

**class** ReportesFamiliarBeneficiario(models.Model):  
 nombre\_completo = models.CharField(max\_length=200)  
 rut = models.CharField(max\_length=12)  
 parentesco = models.CharField(max\_length=20) *# "Jefe Hogar", "Hijo/a", "Cónyuge"*  
 edad = models.PositiveIntegerField()  
 telefono = models.CharField(max\_length=20, blank=True, null=True)  
 acta\_recepcion = models.ForeignKey(ReportesActaRecepcion, on\_delete=models.CASCADE)

## 6. APLICACIÓN INCIDENCIAS - SISTEMA DE SEGUIMIENTO

### 6.1 Modelo IncidenciasObservacion (Observaciones/Problemas)

**class** IncidenciasObservacion(models.Model):  
 *# Identificación del problema*  
 elemento = models.CharField(max\_length=200) *# "Puerta principal", "Instalación eléctrica"*  
 detalle = models.TextField() *# Descripción detallada del problema*  
 prioridad = models.CharField(max\_length=10) *# "ALTA", "MEDIA", "BAJA"*  
 es\_urgente = models.BooleanField(default=False) *# Requiere atención inmediata*  
   
 *# Gestión temporal*  
 fecha\_creacion = models.DateTimeField(auto\_now\_add=True)  
 fecha\_vencimiento = models.DateField(blank=True, null=True) *# Calculada según configuración*  
 fecha\_cierre = models.DateTimeField(blank=True, null=True) *# Cuándo se resolvió*  
 fecha\_ultima\_actualizacion = models.DateTimeField(auto\_now=True)  
   
 *# Seguimiento y responsabilidades*  
 observaciones\_seguimiento = models.TextField(blank=True) *# Notas del proceso*  
 creado\_por = models.ForeignKey(CoreUsuario, on\_delete=models.SET\_NULL, related\_name='observaciones\_creadas')  
 asignado\_a = models.ForeignKey(CoreUsuario, on\_delete=models.SET\_NULL, related\_name='observaciones\_asignadas')  
   
 *# Clasificación*  
 tipo = models.ForeignKey(IncidenciasTipoObservacion, on\_delete=models.DO\_NOTHING)  
 estado = models.ForeignKey(IncidenciasEstadoObservacion, on\_delete=models.DO\_NOTHING)  
   
 *# Ubicación del problema*  
 proyecto = models.ForeignKey(ProyectosProyecto, on\_delete=models.CASCADE)  
 vivienda = models.ForeignKey(ProyectosVivienda, on\_delete=models.CASCADE)  
 recinto = models.ForeignKey(ProyectosRecinto, on\_delete=models.SET\_NULL, blank=True, null=True)

*# Archivos adjuntos*  
 archivo\_adjunto = models.FileField(upload\_to='observaciones/', blank=True, null=True)  
 id\_externo = models.CharField(max\_length=50, blank=True, null=True) *# Para integraciones*

### 6.2 Sistema de Estados y Tipos

**class** IncidenciasEstadoObservacion(models.Model):  
 codigo = models.IntegerField(unique=True) *# 1=Abierta, 2=En proceso, 3=Resuelta*  
 nombre = models.CharField(unique=True, max\_length=50) *# "ABIERTA", "EN\_PROCESO", "RESUELTA"*  
 descripcion = models.TextField() *# Explicación del estado*  
  
**class** IncidenciasTipoObservacion(models.Model):  
 nombre = models.CharField(max\_length=100) *# "Estructural", "Instalaciones", "Acabados"*  
 descripcion = models.TextField() *# Criterios de clasificación*

### 6.3 Trazabilidad y Archivos

**class** IncidenciasSeguimientoObservacion(models.Model):  
 fecha = models.DateTimeField(auto\_now\_add=True)  
 accion = models.CharField(max\_length=100) *# "Creada", "Asignada", "Resuelta"*  
 comentario = models.TextField() *# Detalles del cambio*  
 observacion = models.ForeignKey(IncidenciasObservacion, on\_delete=models.CASCADE)  
 usuario = models.ForeignKey(CoreUsuario, on\_delete=models.SET\_NULL)  
 estado\_anterior = models.ForeignKey(IncidenciasEstadoObservacion, on\_delete=models.SET\_NULL, related\_name='seguimientos\_desde')  
 estado\_nuevo = models.ForeignKey(IncidenciasEstadoObservacion, on\_delete=models.SET\_NULL, related\_name='seguimientos\_hacia')  
  
**class** IncidenciasArchivoAdjuntoObservacion(models.Model):  
 archivo = models.FileField(upload\_to='observaciones/adjuntos/')  
 nombre\_original = models.CharField(max\_length=255) *# Nombre del archivo subido*  
 descripcion = models.CharField(max\_length=255) *# Descripción del contenido*  
 fecha\_subida = models.DateTimeField(auto\_now\_add=True)  
 observacion = models.ForeignKey(IncidenciasObservacion, on\_delete=models.CASCADE)  
 subido\_por = models.ForeignKey(CoreUsuario, on\_delete=models.DO\_NOTHING)

## 7. APLICACIÓN FICHA\_POSTVENTA - EVALUACIONES POST-ENTREGA

### 7.1 Modelo FichaPostventaFichaPostventa (Evaluación Integral)

**class** FichaPostventaFichaPostventa(models.Model):  
 *# Fechas y contexto de la evaluación*  
 fecha\_creacion = models.DateField(auto\_now\_add=True)  
 fecha\_evaluacion = models.DateField() *# Cuándo se hizo la visita*  
 familia\_presente = models.BooleanField() *# Familia estaba en casa*  
 jefe\_hogar\_presente = models.BooleanField() *# Jefe de hogar participó*  
   
 *# EVALUACIÓN TÉCNICA (Infraestructura y servicios)*  
 *# Servicios básicos funcionando*  
 agua\_potable\_funciona = models.BooleanField()  
 electricidad\_funciona = models.BooleanField()  
 alcantarillado\_funciona = models.BooleanField()  
 gas\_funciona = models.BooleanField()  
   
 *# Estados estructurales (escala 1-5: 1=Muy malo, 5=Excelente)*  
 estado\_general\_vivienda = models.IntegerField() *# Evaluación general*  
 estado\_techumbre = models.IntegerField() *# Techo y cubiertas*  
 estado\_muros = models.IntegerField() *# Paredes y estructura*  
 estado\_pisos = models.IntegerField() *# Pisos y pavimentos*  
 estado\_puertas\_ventanas = models.IntegerField() *# Aberturas*  
 estado\_instalacion\_electrica = models.IntegerField() *# Sistema eléctrico*  
 estado\_instalacion\_sanitaria = models.IntegerField() *# Sistema de agua/desagüe*  
   
 *# EVALUACIÓN SOCIAL (Satisfacción familiar)*  
 *# Satisfacción con la vivienda (escala 1-5)*  
 satisfaccion\_general = models.IntegerField() *# Satisfacción general*  
 satisfaccion\_tamano = models.IntegerField() *# Tamaño de la vivienda*  
 satisfaccion\_distribucion = models.IntegerField() *# Distribución de espacios*  
 satisfaccion\_ubicacion = models.IntegerField() *# Ubicación del proyecto*  
   
 *# Necesidades identificadas*  
 requiere\_reparaciones = models.BooleanField() *# Necesita reparaciones*  
 detalle\_reparaciones = models.TextField(blank=True) *# Descripción de reparaciones*  
 requiere\_mejoras = models.BooleanField() *# Necesita mejoras*  
 detalle\_mejoras = models.TextField(blank=True) *# Descripción de mejoras*  
   
 *# EVALUACIÓN COMUNITARIA (Integración social)*  
 adaptacion\_familiar = models.IntegerField() *# Adaptación al nuevo hogar*  
 integracion\_comunitaria = models.IntegerField() *# Integración con vecinos*  
 conoce\_vecinos = models.BooleanField() *# Conoce a los vecinos*  
 participa\_organizaciones = models.BooleanField() *# Participa en organizaciones*  
   
 *# Acceso a servicios (escala 1-5: 1=Muy difícil, 5=Muy fácil)*  
 acceso\_salud = models.IntegerField() *# Acceso a centros de salud*  
 acceso\_educacion = models.IntegerField() *# Acceso a escuelas/colegios*  
 acceso\_transporte = models.IntegerField() *# Acceso a transporte público*  
 acceso\_comercio = models.IntegerField() *# Acceso a comercio/servicios*  
   
 *# Observaciones y seguimiento*  
 observaciones\_familia = models.TextField(blank=True) *# Comentarios familia*  
 observaciones\_tecnicas = models.TextField(blank=True) *# Observaciones técnicas*  
 observaciones\_sociales = models.TextField(blank=True) *# Observaciones sociales*  
 recomendaciones = models.TextField(blank=True) *# Recomendaciones evaluador*  
   
 requiere\_seguimiento = models.BooleanField() *# Requiere nueva visita*  
 fecha\_proximo\_seguimiento = models.DateField(blank=True, null=True) *# Fecha próxima visita*  
   
 *# Relaciones y auditoría*  
 vivienda = models.OneToOneField(ProyectosVivienda, on\_delete=models.CASCADE)  
 evaluador = models.ForeignKey(CoreUsuario, on\_delete=models.DO\_NOTHING, related\_name='fichas\_evaluadas')  
 actualizada\_por = models.ForeignKey(CoreUsuario, on\_delete=models.SET\_NULL, related\_name='fichas\_actualizadas')  
 fecha\_actualizacion = models.DateTimeField(auto\_now=True)  
 activa = models.BooleanField(default=True) *# Para historiales*

### 7.2 Gestión de Archivos y Auditoría

**class** FichaPostventaArchivoFicha(models.Model):  
 tipo = models.CharField(max\_length=20) *# "foto", "documento", "video"*  
 archivo = models.FileField(upload\_to='fichas\_postventa/')  
 descripcion = models.CharField(max\_length=200) *# Descripción del archivo*  
 fecha\_subida = models.DateTimeField(auto\_now\_add=True)  
 ficha = models.ForeignKey(FichaPostventaFichaPostventa, on\_delete=models.CASCADE)  
 subido\_por = models.ForeignKey(CoreUsuario, on\_delete=models.DO\_NOTHING)  
  
**class** FichaPostventaHistorialFicha(models.Model):  
 fecha\_cambio = models.DateTimeField(auto\_now\_add=True)  
 campo\_modificado = models.CharField(max\_length=100) *# Nombre del campo modificado*  
 valor\_anterior = models.TextField() *# Valor antes del cambio*  
 valor\_nuevo = models.TextField() *# Valor después del cambio*  
 observaciones = models.TextField(blank=True) *# Razón del cambio*  
 ficha = models.ForeignKey(FichaPostventaFichaPostventa, on\_delete=models.CASCADE)  
 usuario = models.ForeignKey(CoreUsuario, on\_delete=models.DO\_NOTHING)

## 8. CONFIGURACIÓN Y ADMINISTRACIÓN DEL SISTEMA

### 8.1 Configuraciones Operativas

**class** CoreConfiguracionObservacion(models.Model):  
 dias\_vencimiento\_normal = models.IntegerField(default=30) *# Días para observaciones normales*  
 horas\_vencimiento\_urgente = models.IntegerField(default=24) *# Horas para observaciones urgentes*  
 fecha\_modificacion = models.DateTimeField(auto\_now=True)  
 modificado\_por = models.ForeignKey(CoreUsuario, on\_delete=models.SET\_NULL)

### 8.2 Sistema de Autenticación Django

El sistema utiliza el framework de autenticación estándar de Django con las siguientes tablas:

* **auth\_group** - Grupos de usuarios
* **auth\_permission** - Permisos específicos del sistema
* **auth\_group\_permissions** - Relación grupos-permisos
* **core\_usuario\_groups** - Asignación de usuarios a grupos
* **core\_usuario\_user\_permissions** - Permisos directos a usuarios
* **django\_content\_type** - Tipos de contenido para permisos
* **django\_admin\_log** - Log de acciones en el admin
* **django\_session** - Sesiones de usuarios

## 9. RELACIONES Y CONSTRAINTS PRINCIPALES

### 9.1 Relaciones One-to-One

* **ProyectosVivienda ↔ ProyectosBeneficiario** - Una vivienda por beneficiario
* **ProyectosVivienda ↔ ReportesActaRecepcion** - Un acta por vivienda
* **ProyectosVivienda ↔ FichaPostventaFichaPostventa** - Una ficha por vivienda

### 9.2 Relaciones One-to-Many Críticas

* **ProyectosProyecto → ProyectosVivienda** - Un proyecto tiene múltiples viviendas
* **CoreRegion → CoreComuna** - Una región tiene múltiples comunas
* **ProyectosBeneficiario → ProyectosTelefono** - Un beneficiario puede tener múltiples teléfonos
* **IncidenciasObservacion → IncidenciasSeguimientoObservacion** - Una observación tiene múltiples seguimientos

### 9.3 Unique Constraints Importantes

* **core\_usuario.email** - Email único para login
* **core\_usuario.rut** - RUT único para identificación
* **proyectos\_proyecto.codigo** - Código único de proyecto
* **(proyectos\_vivienda.proyecto\_id, proyectos\_vivienda.codigo)** - Código único por proyecto
* **(core\_comuna.nombre, core\_comuna.region\_id)** - Comuna única por región
* **reportes\_actarecepcion.numero\_acta** - Número de acta único

## 10. ÍNDICES Y OPTIMIZACIONES

### 10.1 Índices Implementados

* **Campos de búsqueda frecuente:** email, rut, codigo, fecha\_creacion
* **Foreign Keys:** Automáticamente indexados por Django
* **Campos de filtro:** region\_id, comuna\_id, estado, activo

### 10.2 Optimizaciones de Consultas

* **select\_related()** para reducir queries en relaciones FK
* **prefetch\_related()** para relaciones Many-to-Many
* **Lazy loading** para campos de texto largo no siempre necesarios

## 11. INTEGRIDAD Y VALIDACIONES

### 11.1 Validaciones a Nivel de Modelo

*# Ejemplo de validaciones personalizadas*  
**def** clean(self):  
 **if** self.fecha\_entrega **and** self.fecha\_termino\_postventa:  
 **if** self.fecha\_termino\_postventa <= self.fecha\_entrega:  
 **raise** ValidationError('Fecha término postventa debe ser posterior a entrega')

### 11.2 Triggers y Señales Django

@receiver(post\_save, sender=IncidenciasObservacion)  
**def** calcular\_fecha\_vencimiento(sender, instance, created, \*\*kwargs):  
 **if** created:  
 config = CoreConfiguracionObservacion.objects.first()  
 **if** instance.es\_urgente:  
 instance.fecha\_vencimiento = instance.fecha\_creacion + timedelta(hours=config.horas\_vencimiento\_urgente)  
 **else**:  
 instance.fecha\_vencimiento = instance.fecha\_creacion + timedelta(days=config.dias\_vencimiento\_normal)  
 instance.save()

## 12. MÉTRICAS DEL MODELO DE DATOS

### 12.1 Estadísticas de Estructura

| Métrica | Valor |
| --- | --- |
| **Total de tablas** | 27 |
| **Aplicaciones Django** | 6 |
| **Modelos principales** | 21 |
| **Tablas de sistema Django** | 6 |
| **Foreign Keys** | 35+ |
| **Unique constraints** | 12 |
| **Campos total estimado** | 150+ |
| **Relaciones One-to-One** | 3 |
| **Relaciones Many-to-Many** | 2 (vía auth system) |

### 12.2 Capacidad y Escalabilidad

**Estimación de registros por tabla (producción):** - **core\_usuario** - 1,000 usuarios (coordinadores, beneficiarios, constructores) - **proyectos\_proyecto** - 100 proyectos anuales - **proyectos\_vivienda** - 5,000 viviendas por año - **incidencias\_observacion** - 10,000 observaciones anuales - **ficha\_postventa\_fichapostventa** - 5,000 fichas anuales

## 13. BACKUP Y RECUPERACIÓN

### 13.1 Estrategia de Backup

* **Backup completo diario** de toda la base de datos
* **Backup incremental cada 6 horas** para tablas críticas
* **Retención de 30 días** para backups diarios
* **Retención de 1 año** para backups mensuales

### 13.2 Tablas Críticas Prioritarias

1. **proyectos\_vivienda** - Información central del sistema
2. **reportes\_actarecepcion** - Documentación legal
3. **ficha\_postventa\_fichapostventa** - Evaluaciones oficiales
4. **core\_usuario** - Usuarios del sistema
5. **incidencias\_observacion** - Seguimiento de problemas

## 14. CONCLUSIONES TÉCNICAS

### 14.1 Fortalezas del Modelo

**Normalización Apropiada:** - Eliminación de redundancia de datos - Integridad referencial garantizada - Estructura modular y mantenible

**Trazabilidad Completa:** - Auditoría en tablas críticas (created\_by, updated\_by, timestamps) - Historial de cambios en fichas postventa - Seguimiento detallado de observaciones

**Flexibilidad y Escalabilidad:** - JSONField para datos variables (elementos\_disponibles) - Soft deletes mediante campos ‘activo’ - Estructura preparada para múltiples regiones

### 14.2 Consideraciones de Rendimiento

**Optimizaciones Implementadas:** - Índices en campos de búsqueda frecuente - Foreign Keys con estrategias on\_delete apropiadas - Campos de auditoría para debugging y análisis

**Áreas de Mejora Futura:** - Implementar particionado por región/año para tablas grandes - Considerar denormalización controlada en reportes frecuentes - Implementar cache de consultas complejas

### 14.3 Impacto Organizacional

Este modelo de datos robusto permite a TECHO Chile:

* **Seguimiento integral** desde planificación hasta postventa
* **Trazabilidad completa** de procesos y decisiones
* **Reportería avanzada** para medición de impacto social
* **Gestión eficiente** de recursos y coordinación
* **Transparencia** en la gestión de proyectos habitacionales

La arquitectura implementada constituye una base sólida para el crecimiento y evolución futura del sistema, permitiendo incorporar nuevas funcionalidades sin requerir cambios estructurales mayores.

**Versión del modelo:** 2.0  
**Última actualización:** 15 de octubre de 2025  
**Total de migraciones aplicadas:** 15