

Universidad Tecnológica Nacional - Facultad Regional Santa Fe

Ingeniería en Sistemas de Información

Proyecto Final de Carrera

***“Sistema de Autoliquidación WEB de Tasas,
Derechos y Servicios para Municipios”***

Alumnos:

CHAPINO, Matías Jesús (LU:14660 – chapino.matias@gmail.com)

JACQUET, Juan Manuel (LU:14714 – jmjacquet@gmail.com)

Director:

Ing.QUAGLIOTTI, Leandro Hernán

AÑO 2020

Índice General

Introducción	3
Problemática	4
Objetivos y Alcances	4
Organización del Informe	6
Capítulo 1	7
1.1. La Organización	7
1.2. Conceptos Teóricos	8
1.2.1. Derecho de Registro e Inspección	8
1.2.2. Interés Punitorio	9
Capítulo 2	11
2.1. Equipo de Trabajo	11
2.2. Metodología de Desarrollo de Software	11
2.3. Análisis de Esfuerzos y Plazos estimados	13
Capítulo 3	15
3.1. Arquitectura del Sistema	15
3.1.1. Arquitectura Cliente/Servidor	15
3.2. Arquitectura y Diseño	17
3.2.1. Estructura de la Aplicación y consideraciones generales:	19
3.3. Modelo de Datos	21
Capítulo 4	24
4.1. Esquema del Sitio	24
4.2. Funcionalidades del Sistema	25
Capítulo 5	40
5.1. Herramientas de Gestión del Proyecto	40
5.2. Herramientas y Tecnologías	40
Capítulo 6	43
6.1. Control de Accesos	43

6.2. Estructuración del Proyecto	45
6.3. Interacción de Componentes y Tecnologías	46
6.3.1. Proceso de Sincronización	46
6.3.2. Proceso de Interacción con el Sistema	46
6.3.3. Proceso de Interacción con la API	47
Capítulo 7	48
7.1. Testing	48
7.2. Metodología Utilizada	48
7.3. Integración Continua (CI)	49
7.4. Políticas de Publicación	51
7.5. Control de Versiones (GIT)	51
Capítulo 8	53
8.1 Seguridad	53
8.1.1. Seguridad en el Protocolo	55
Capítulo 9	57
9.1 Extensibilidad del Sistema	57
Conclusión	60
Referencias Bibliográficas	62
Anexos	64
A. Modelo MVT de Django	64
B. Estructura de una Aplicación DJANGO	65
C. Tipos de Testing	67
C.1. Testing Estructural o de Caja Blanca	67
C.2. Testing Basado en Modelos o de Caja Negra	67
C.3. Pruebas Funcionales y No Funcionales	68
D. Refactoring de código fuente	68
Glosario de Definiciones	69

Introducción

En los últimos años, ha aumentado la necesidad de implementar proyectos que acerquen al ciudadano con el Estado, de modo que este último provea servicios de mayor calidad basados en alta tecnología. Se revela en todos los sectores de la sociedad (público, privado, civil, académica) la necesidad de usar y optimizar las oportunidades que presentan las Tecnologías de la Información y Comunicación (TIC) para mejorar la gobernabilidad, para implementar nuevos canales de comunicación entre gobiernos y ciudadanos, para tejer y reforzar redes comunitarias, para asegurar la transparencia gubernamental, para ingresar en forma proactiva a la Sociedad de la Información y para no quedar al margen de la nueva economía. Se necesitan formas innovadoras de interacción entre los ciudadanos y los gobernantes locales, nuevas concepciones de la política local, utilizando medios electrónicos.

El proyecto consistió en desarrollar un sistema que permitiera a los contribuyentes auto gestionar sus cuentas corrientes en relación al Municipio, apuntando a aprovechar las TIC's para brindar una mejor y mayor eficiencia en la atención al público, mejorar el funcionamiento del proceso de trabajo, obtener una mayor transparencia de la gestión pública, obtener un ahorro importante desde el punto de vista económico y de tiempo. Una de las premisas del proyecto ha sido ensayar tecnologías inherentes al desarrollo de software de calidad haciendo más eficiente y racionalizando el modo de trabajo, reemplazando el trabajo manual. Así, ha sido posible evaluar fortalezas y debilidades en distintos aspectos del desarrollo y comportamiento de este tipo de software en distintas áreas.

El presente informe tiene como propósito describir el proceso de desarrollo de un sistema web de mediano porte para Municipios y Comunas, clientes de Grupo Guadalupe SRL.

Grupo Guadalupe SRL es una empresa de software con sede en la ciudad de Santa Fe que ofrece sistemas para Municipios/Comunas y cuenta entre su cartera de clientes con más de 140 entidades municipales.

Problemática

Anteriormente a la implementación del sistema de Autogestión Web, cada contribuyente para acceder a su cuenta corriente y poder realizar consultas y/o gestiones sobre la misma debía concurrir al Municipio y realizar el trámite de forma presencial. Entre los trámites más comunes se encuentran:

- Consulta de Cuenta Corriente.
- Reimpresión de Boletas.
- Pago de periodos adeudados.
- Liquidación de DDJJ del Derecho Registro e Inspección (DReI).

Esto ocasionaba no sólo una pérdida de tiempo al contribuyente sino también un desperdicio de los recursos humanos del municipio (destinando gran parte de su tiempo a la atención al público).

Otro de los inconvenientes provocados al Municipio eran los costos de impresión de las boletas y de logística para el reparto de las mismas.

Objetivos y Alcances

El objetivo entonces, es lograr que el contribuyente pueda efectuar el pago de impuestos y multas, simplificando y automatizando los procesos de cobros rutinarios con el objeto de reducir en forma considerable la necesidad de operaciones manuales, tiempo y uso de insumos del Municipio.

El objetivo general del proyecto fue diseñar y desarrollar un sistema web de autogestión de Tasas, Derechos y Servicios Municipales orientados al ciudadano. A continuación se enumeran los objetivos específicos y los beneficios asociados a la implementación del sistema:

- Reducir la cantidad de contribuyentes en el sector de atención al público del municipio.

- Reducir costos de impresión a través de la boleta digital como así también los costos asociados al reparto de boletas.
- Brindar al ciudadano el acceso y gestión de su cuenta corriente de forma online.
- Brinda a los Estudios Contables el acceso a la cuenta corriente de sus clientes para la liquidación del DRel.
- Brindar estadísticas al Municipio sobre DDJJ de DRel.
- Brindar estadísticas al Municipio de Pagos en línea.
- Diseñar mecanismos de visualización de deuda y descarga de boletas.
- Diseñar mecanismos de pago de deuda en línea.
- Desarrollar algoritmos para el cálculo de intereses (según tributo).
- Interpretar la carga manual del formulario de DRel e identificar los datos necesarios para la autoliquidación.
- Desarrollar una arquitectura flexible que permita incorporar nuevos tributos.
- Establecer normas de sincronización de datos con el Sistema de Gestión Local del Municipio.
- Diseñar e implementar una interfaz de usuario sencilla y adaptable a múltiples dispositivos.

Organización del Informe

El presente documento se organizó de manera que los conceptos se vayan introduciendo gradualmente y conteniendo tanto fundamentos teóricos como detalles de implementación.

El Capítulo 1, se centra en describir la organización sobre la cual fue desarrollado el sistema e introduce conceptos teóricos sobre los tributos utilizados.

El Capítulo 2, realiza una descripción del proyecto, describiendo la metodología usada y los principales requerimientos.

El Capítulo 3, se centra en aspectos de diseño del sistema, describiendo las arquitecturas y servicios utilizados.

El Capítulo 4, describe el producto realizado, detallando funcionalidades y módulos desarrollados.

El Capítulo 5, describe de forma resumida las tecnologías y herramientas involucradas en el desarrollo del proyecto.

El Capítulo 6, se basa en la implementación del sistema, mostrando funcionalidades, estructura del código.

El Capítulo 7, describe prácticas realizadas para el control de calidad y despliegue del software en cuestión.

El Capítulo 8, describe los mecanismos de seguridad propios del Framework y los desarrollados e implementados en el sistema.

El Capítulo 9, describe los módulos desarrollados previendo la extensibilidad del sistema, posibles módulos a incorporar en futuras versiones.

Por último, se presentan las conclusiones obtenidas del desarrollo del presente trabajo final, teniendo en cuenta el problema y los objetivos planteados.

Capítulo 1 - Contexto

Como etapa inicial de análisis de sistemas en todo proceso de desarrollo, es fundamental comprender el área en el cual se va a trabajar, el funcionamiento y estructura de la organización comprometida, así como sus procesos. El presente capítulo intenta describir brevemente la organización sobre la cual fue desarrollado el sistema e introduce conceptos teóricos sobre los tributos a ser impresos/liquidados en el sistema de autogestión, para situarnos en contexto.

1.1. La Organización

El sistema no fue desarrollado para una organización en particular, sino para satisfacer las necesidades de un conjunto de Municipios los cuales son clientes de Grupo Guadalupe SRL. El proyecto se centró en el Área de Tributos de estos Municipios, específicamente en la emisión y recaudación de impuestos.

El tributo es un ingreso público que consiste en prestaciones pecuniarias exigidas por una administración pública con una fecha de cumplimiento. Es un pago que se usa para satisfacer determinadas necesidades de la administración y que impactará en los ciudadanos.

Cada Municipio dispone por Ordenanza todos los Tributos que administra como así también el detalle de cómo se gestiona cada uno de ellos. Dicha Ordenanza establece a grandes rasgos la periodicidad en la emisión para cada Tributo, la forma de generación de la deuda con sus respectivos valores y las fechas de vencimientos de cada período.

La emisión de un periodo consta básicamente de dos pasos. En primer lugar se realiza la generación de la deuda para un periodo determinado. Este proceso consiste en recorrer el padrón de los contribuyentes pertenecientes al Tributo y por cada uno de estos, realizar el cálculo correspondiente según se indique en la Ordenanza. El resultado del proceso es una cuota/boleta asociada a cada uno de los ciudadanos del padrón. Una vez realizada la generación del Tributo, el segundo paso consiste en la impresión en papel de las boletas y posterior reparto de estas.

Del lado del Contribuyente, cuando este recibe la boleta puede pagarla tanto en el propio Municipio como en otros lugares de pago, según los convenios formalizados entre el Organismo y los entes recaudadores (Bancos, Mutuales, Link, Banelco, etc.). Esta opción de pago sólo está disponible siempre y cuando no se supere la fecha del segundo vencimiento.

Una vez vencida la cuota, la cancelación de la deuda se debe realizar en el Municipio dado que se aplica una fórmula específica a cada Tributo para realizar el cálculo de los intereses punitivos derivados del incumplimiento del pago a término.

Entre los Tributos más comunes se encuentran:

- Tasa General Inmueble Urbana y Rural (TGIU/TGIR).
- Contribuciones por Mejoras (CMEJ).
- Servicios de Agua Potable y Cloaca (SERV).
- Tasas Administrativas.
- Moratorias/Convenios.
- Derecho de Registro e Inspección (DRel).

1.2. Conceptos Teóricos

1.2.1. Derecho de Registro e Inspección

El Derecho de Registro e Inspección (DRel) es un tributo de abono mensual aplicado sobre locales comerciales, industriales y de servicios. Lo cobra el gobierno municipal por los servicios que presta:

- Registro, habilitación y control de actividades comerciales, industriales, científicas, de investigación y lucrativas en general.
- Preservación de la salubridad, seguridad e higiene.
- Fiscalización de la fidelidad de pesas y medidas.
- Inspección y control de instalaciones eléctricas, motores, máquinas en general y generadores.

- Supervisión de vidrieras y publicidad en las mismas o en el local habilitado; inspección y habilitación de elementos publicitarios fuera del local inscripto instalados en o hacia la vía pública, en vehículos en general o en locales e instalaciones de terceros, previa autorización especial reglamentaria.
- Habilitación de mesas, sillas y similares con fines comerciales, en la vía pública o espacios públicos.

El DRel es un impuesto autoliquidado y tiene un trato especial con respecto a los demás tributos. Cada contribuyente posee una serie de actividades asociadas a su padrón. Por ordenanza se establece un monto mínimo y una alícuota por cada actividad. Los pasos para la liquidación mensual del impuesto son los siguientes:

- Por cada una de las actividades se declara la base imponible del período y se calcula el monto de la actividad según la alícuota asociada a esta. Si el valor obtenido no supera el monto mínimo de la actividad se toma este último como valor declarado.
- Se calcula un subtotal derivado de la suma de los valores declarados en cada actividad.
- Pueden hacerse diferentes cálculos adicionales, los cuales pueden ser montos fijos o porcentajes sobre el subtotal de las actividades.
- El monto total de la liquidación queda definido como la suma del subtotal de las actividades más los adicionales calculados.

En caso de una presentación mal liquidada para un periodo, se puede realizar una Rectificativa para corregir el inconveniente. La Rectificativa se realiza de la misma forma que la liquidación normal, solo se corrigen las bases imponibles de las actividades erróneas. Si el periodo presenta pagos registrados se declara el monto de los mismos.

1.2.2. Interés Punitorio

El interés punitorio se produce cuando el pago de la cuota correspondiente no se realiza en la fecha pactada. Se trata del incumplimiento de la obligación. El cálculo

se realiza al momento de querer efectuar el pago de la cuota y se suma al valor nominal de esta.

Cada Municipio tiene múltiples formas de realizar el cálculo de interés y a su vez puede diferir para cada tributo. Las formas más utilizadas son las siguientes:

- *Coeficiente Diario*: Se establece un coeficiente de interés diario y se calculan los días transcurridos entre el primer vencimiento y la fecha del cálculo del interés. El Interés a Pagar resulta de la multiplicación del Importe Nominal por el Coeficiente de Interés por la Cantidad de Días Transcurridos.
- *Coeficiente Periódico Diario*: el cálculo es similar al del Coef. Gral. de Interés Diario con la diferencia que el coeficiente varía según una tabla de periodos con fecha de inicio y fin. Esto permite configurar el interés histórico del tributo en los diferentes momentos según la Ordenanza correspondiente y aplicar el coeficiente adecuado en cada periodo.
- *Coeficiente Mensual*: Se establece un coeficiente de interés mensual y se calculan la cantidad de meses transcurridos entre el primer vencimiento y la fecha del cálculo del interés. El Interés a Pagar resulta de la multiplicación del Importe Nominal por el Coeficiente de Interés por la Cantidad de Meses Transcurridos.

Capítulo 2 - Equipo y Metodología de Desarrollo

El presente Capítulo detalla aspectos relacionados con el proyecto de desarrollo, describiendo el equipo de trabajo y la metodología de desarrollo aplicada.

2.1. Equipo de Trabajo

Con el fin de concretar el proyecto, se creó un equipo de trabajo interdisciplinario, conformado por:

- Asesores/Jefes de Áreas de diferentes Municipios, quienes definieron los requerimientos del sistema, cambios y mejoras y evacuaron las dudas surgidas a lo largo del proceso y coordinaron los esfuerzos de trabajo;
- Desarrolladores y autores del proyecto, estudiantes avanzados de la carrera de Ingeniería en Sistemas de la UTN FRSF: Matías Jesús Chapino y Juan Manuel Jacquet (ambos empleados de Grupo Guadalupe SRL) quienes llevamos a cabo el desarrollo y conducción del proyecto.

2.2. Metodología de Desarrollo de Software

Como metodología de desarrollo de software, se optó por utilizar una combinación de Metodologías Ágiles [3], las cuales dan mayor valor al individuo, a la colaboración con el cliente y al desarrollo incremental del software con iteraciones muy cortas. Este enfoque tiene una gran efectividad en proyectos con requisitos cambiantes y cuando se exige reducir drásticamente los tiempos de desarrollo pero manteniendo una alta calidad.

Se utilizó Scrum [7] en virtud de ofrecer un conjunto de buenas prácticas para trabajar colaborativamente en equipo y una forma exitosa de realizar incrementos funcionales de negocio (Sprint). Sin embargo, esta metodología carece de una definición de etapas para conseguir el producto de software por eso se complementó con ciertas herramientas de la Programación Extrema (XP). La combinación de

Scrum y XP [2] supuso una gran ayuda en el proceso de desarrollo evitando la documentación excesiva y haciendo de los clientes parte del equipo.

Para el desarrollo del proyecto, se dividió la metodología de trabajo en dos grandes fases:

- *Investigación*: Compuesta por actividades de investigación y documentación de los métodos utilizados hasta el momento. Las mismas tienen fecha de inicio y fin, procedencia y grado de relevancia.
- *Desarrollo*: Nos basamos en Scrum, tomando algunas de las principales ventajas de dicha metodología, tales como Product Backlog, Sprint Backlog, Weekly StandUp Meetings y Sprint Review [2]. Al ser solo dos programadores no se llegó a conformar el “equipo” liderado por un Scrum Master. Se tomaron turnos según el Sprint a desarrollar (en base a los conocimientos de cada uno). La revisión de las nuevas funcionalidades quedó cubierta por la persona que no “escribió” el código (con el objetivo de evaluar los datos de entrada correctamente).

Para dar por finalizado cada Sprint en primer lugar se realizaron pruebas de Control de Calidad, con el fin de identificar y resolver errores surgidos en el testeo de la aplicación, y se corrieron los Test Automáticos pertinentes (Pruebas Funcionales y Unitarias). Luego de testear la aplicación y verificar el cumplimiento de todos los puntos detallados en el Sprint Backlog, se dio paso al Despliegue del mismo. Se integró la rama de prueba a la versión de producción y se actualizó la documentación correspondiente.

Finalmente se comunicó a los Clientes/Usuarios del sistema sobre la nueva funcionalidad (según fuese el requerimiento).

Estas etapas se realizaron de forma iterativa e incremental. Cada iteración del ciclo de vida incluyó estas etapas y el objetivo de cada iteración fue ir incorporando nuevas funcionalidades al sistema mientras iba avanzando el proyecto, hasta obtener el producto completo terminado.

El proyecto se dividió en Iteraciones (las cuales estaban compuestas por un conjunto de actividades bien definidas):

- Análisis, Planificación e Investigación de métodos existentes.
- Determinación de Actividades a desarrollar, separación en Sprints.
- “n” Sprints (por cada uno: desarrollo, corrección de errores y entrega de versión).
- Documentación y preparación de la defensa del proyecto.

Se eligió esta forma de trabajar dado que permitió tener un mayor control en la evolución del Proyecto [5] realizando iteraciones cortas y fijas que aportaron valor a cada versión terminada; se obtuvo un seguimiento más minucioso ante posibles cambios, mejor predicción de tiempos y control de riesgos.

2.3. Análisis de Esfuerzos y Plazos estimados

La estimación de esfuerzos planteada inicialmente (etapa de Análisis y Planificación) [6] pudo ser respetada a la perfección y se obtuvieron en tiempo y forma los resultados esperados. Todo esto pudo ser logrado gracias al apoyo y buena predisposición del equipo de relevamiento (asesores y jefes de área), lo cual creemos fue decisivo para haber concretado el total de las funcionalidades requeridas en el proyecto.

Con respecto a la etapa de desarrollo iterativo, las estimaciones de los primeros Sprints fueron revisadas y corregidas, principalmente por falta de conocimiento en las diferentes herramientas utilizadas, ya sea propias del Framework Django (plugins y librerías externas) como de puesta en marcha para el ambiente de desarrollo (uso de Docker containers, versiones del lenguaje, software de gestión del proyecto, etc).

También se tuvieron en cuenta ciertos requerimientos no incluidos en el planteo inicial, como por ejemplo la utilización de una API (application programming interface) Rest para trabajar sobre un entorno seguro, por lo tanto se añadió el requerimiento de utilizar HTTPS y la creación de un mecanismo de autenticación segura para dicha API.

Con las Iteraciones posteriores se pudo respetar lo planificado casi a rajatablas ajustando a los tiempos estimados inicialmente, esto es en gran parte a la utilización de herramientas de monitoreo continuo del proyecto. Para ello se utilizó un gráfico de trabajo pendiente a lo largo del tiempo para visualizar el progreso en el cumplimiento de los objetivos/requisitos. Esto permitió la comparación del avance real contra el planificado y así se detectó casi al instante cualquier desviación con respecto a las fechas pactadas inicialmente.

Capítulo 3 - Arquitectura y Modelo de Datos

En el presente capítulo se dará una breve descripción de conceptos de arquitectura de software, estilos y patrones arquitectónicos utilizados.

3.1. Arquitectura del Sistema

La arquitectura de software [7] es el diseño de más alto nivel de la estructura de un sistema. Una arquitectura de software consiste en un conjunto de patrones y abstracciones coherentes que proporcionan un marco definido para interactuar con el código fuente. La arquitectura se selecciona y diseña en base a los requerimientos y restricciones del sistema a desarrollar, es decir, en base a requerimientos funcionales y no funcionales y del tipo de sistema a desarrollar.

Bass, Clement y Kazman [6] definen la arquitectura de software como la estructura del sistema, que incluyen los componentes de software, las propiedades visibles externamente de esos componentes y las relaciones entre ellos. Lo más común a la hora de definir el tipo de arquitectura a utilizar en un proyecto de desarrollo, es adoptar una arquitectura conocida en función de sus ventajas e inconvenientes para el caso concreto.

3.1.1. Arquitectura Cliente/Servidor

La arquitectura Cliente/Servidor (C/S) [8], es un modelo de diseño de software en donde las tareas se reparten entre los proveedores de recursos o servicios, llamados “servidores”, y quienes demandan estos servicios, llamados “clientes”.

El cliente realiza peticiones al servidor y recibe la respuesta a dicha petición. El servidor recibe y procesa la petición, y devuelve la respuesta solicitada al cliente.

A continuación se presenta un esquema básico de una arquitectura C/S, donde el medio de comunicación es Internet, podría ser una Intranet también, o una red local.

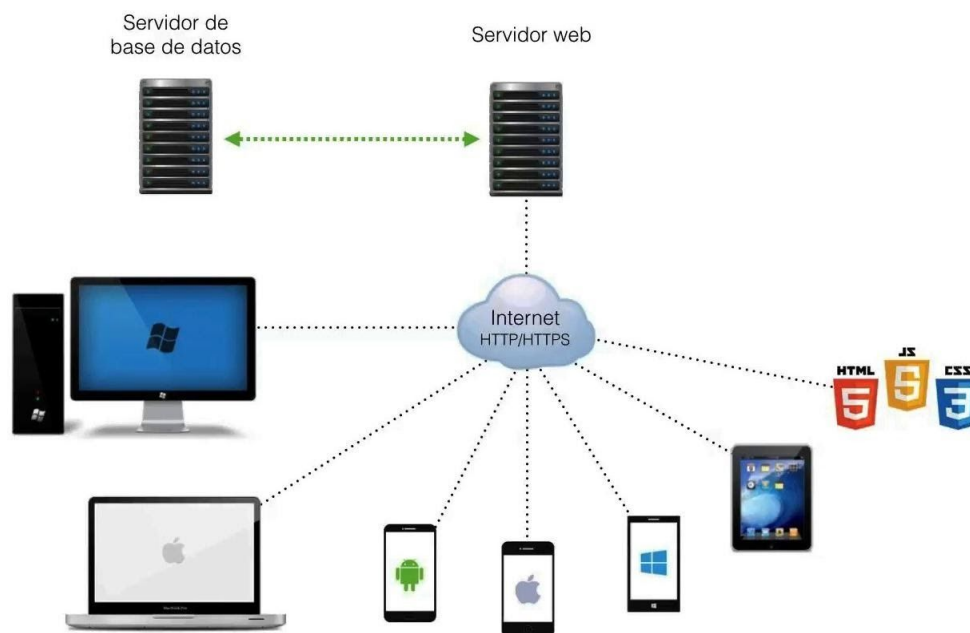


Fig.1 - Arquitectura Cliente-Servidor.

La arquitectura C/S, ha evolucionado a lo largo del tiempo, pasando de arquitecturas monolíticas (de una sola capa), hasta arquitecturas de dos y tres capas.

En una arquitectura C/S de tres capas, el cliente implementa la capa de presentación, y tenemos un servidor para implementar la lógica de negocio (capa de negocio) y otro para el acceso a datos (capa de datos). Ésta última es la que más ventajas proporciona, destacándose que es un sistema débilmente acoplado, escalable y con mayor facilidad de mantener.

REST (Representational State Transfer) [9] es una arquitectura de software para sistemas hipermedias distribuidos tales como la Web, es una forma de proporcionar interoperabilidad entre los sistemas informáticos en Internet. El término se originó en el año 2000 por Roy Fielding, el padre de la especificación HTTP, y ha pasado a ser ampliamente utilizado por la comunidad de desarrollo.

Si bien el término REST se refería originalmente a un conjunto de principios de arquitectura, en la actualidad se usa en un sentido más amplio para describir

cualquier interfaz entre sistemas que utilice directamente HTTPS para obtener datos o indicar la ejecución de operaciones sobre los datos, en cualquier formato (XML, JSON, etc.) sin las abstracciones adicionales de los protocolos de intercambio de mensajes, como SOAP (Simple Object Access Protocol).

3.2. Arquitectura y Diseño

El sistema se basa en el mencionado modelo Cliente/Servidor de 3 capas (ver anexo “*Modelo MVT de Django*”). Las peticiones pueden ser tanto de un cliente que esté interactuando con la interfaz web como de un servicio que solicite datos a la API.

Para el Servidor Remoto, se contrató un servicio de hosting VPS en Webfaction (empresa que brinda servicios de hosting/vps) el cual provee una Infraestructura de servicios basados en la Nube, conocido como IaaS (Infrastructure as a Service), los cuales están formados por una serie de servidores cuya escalabilidad es fácil de modificar e implementar. Dicho servicio es customizable y sencillo de administrar, provee herramientas para monitoreo del tráfico, almacenamiento y otro tipo de servicios extra (por ejemplo estadísticas sobre el uso e ingresos al sistema). IaaS permite escalabilidad a medida que se vayan aumentando los requerimientos en el Servidor. El Servidor de Bases de Datos Remoto es MySQL en su versión 5.7 y es provisto y administrado por Webfaction.

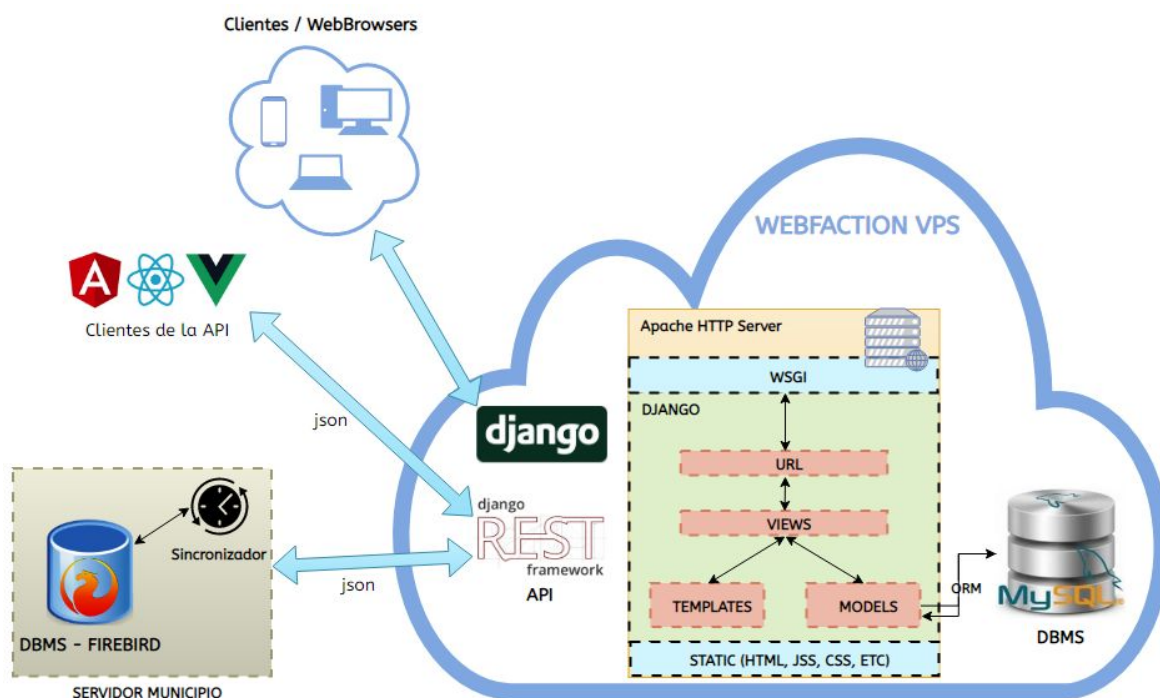


Fig.2 - Esquema del sistema IaaS brindado por Webfaction.

El Motor de Bases de Datos del servidor local es Firebird en su versión 2.5 administrado localmente como parte del servicio de sistemas locales TaDeSe por parte de Grupo Guadalupe SRL.

La comunicación entre Sincronizador/Servidor Remoto se realiza mediante el envío/recepción de paquetes JSON (actualización de datos) a través de la API Rest.

El Framework Django [10] recibe solicitudes al servidor de aplicaciones *Apache* en un puerto específico definido según sea el módulo. Procesa la petición e interactúa con el servidor de Base de Datos (DBMS) si dicha solicitud lo requiere y luego genera la respuesta con el resultado obtenido.

3.2.1. Estructura de la Aplicación y consideraciones generales:

- Se tiene una base MySQL (servidas por WebFaction) por cada municipio, lo cual permite mantener un total aislamiento de datos.

- Cada base posee un código y nombre únicos definidos según lo asigne la empresa Grupo Guadalupe SRL.
- El código fuente (aplicación *TaDeSe WEB*) es único para todos los municipios, lo cual brinda la ventaja de tener un solo punto de actualización.
- El módulo *mod_wsgi* encargado de interactuar entre Django y Apache, nos permite definir múltiples instancias de una misma aplicación llamados “*workers*” los cuales trabajan de manera aislada en diferentes hilos de ejecución del servidor.
- Se creó un dominio “*boletaweb.com.ar*” bajo el cual cada uno de los municipios acceden a la aplicación mediante la asignación de un subdominio propio. Por ej. “*muniNN.boletaweb.com.ar*” en donde *muniNN* es el nombre asignado al municipio.

Bajo estas premisas se procedió a desarrollar una solución específica para poder manejar los requerimientos de cada uno de los subdominios y direccionarlos dentro de la aplicación a cada base de datos correspondiente. En los archivos de configuración del servidor Apache definimos una serie de variables de entorno por cada subdominio existente.

El ciclo de vida del requerimiento queda definido de la siguiente manera:

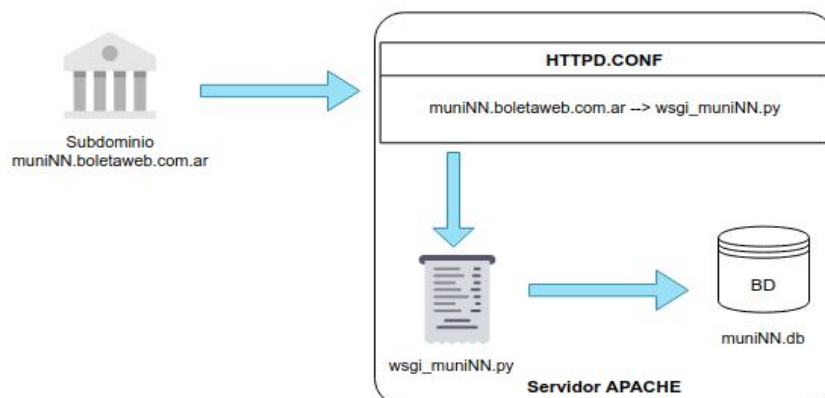


Fig.3 - Ciclo de Vida del Requerimiento.

El requerimiento es atendido y direccionado según lo define cada archivo de instancias WSGI a los subdominios que lo requieran.

Los archivos de tipo “*wsgi_muniNNN.py*” poseen la configuración inicial de cada municipio, definiendo la base de datos a ser accedida, código de municipio y directorio de archivos estáticos utilizados en la aplicación (se puede configurar su logo, archivos de estilo CSS, etc).

Una vez iniciado el requerimiento y seteadas las variables, el servidor crea una instancia de Django ya configurada, apuntando a la base de datos del subdominio que emitió la solicitud. Dicha instancia permanece a la espera un tiempo determinado (definido en el archivo de configuración del servidor) y, cuando no posee más requerimientos, es terminada.

De esta manera podemos atender requerimientos de diferentes subdominios, cada uno direccionado a la instancia que corresponde con su espacio de memoria e hilo de ejecución propio. El nivel de seguridad y los permisos fueron establecidos según el origen y la solicitud del requerimiento.

3.3. Modelo de Datos

Se procede a describir el modelo de tablas del sistema, dejando afuera las tablas propias del framework Django (autenticación, usuarios, etc).

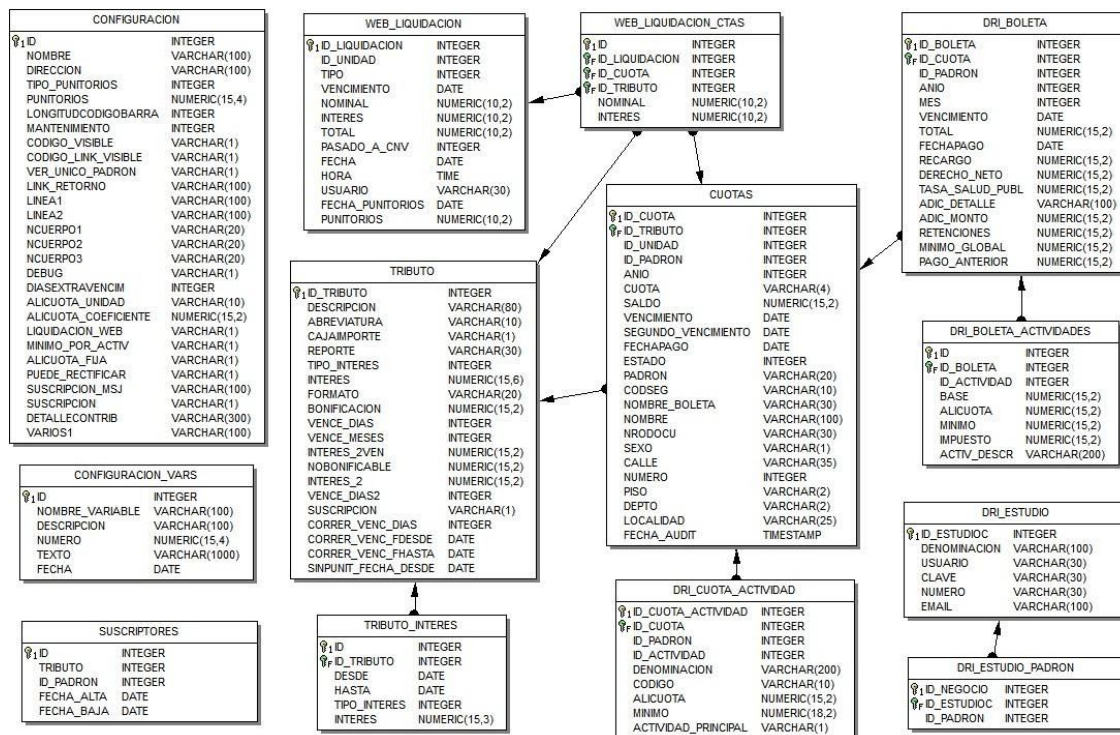


Fig.4 - Modelo de Tablas del Sistema.

Tabla	Descripción
TRIBUTOS	Se definen los Tributos que intervienen, define los cálculos punitorios, bonificaciones y vencimientos.
TRIBUTOS_INTERES	Rango de intereses según el tipo de Tributo y cálculos punitorios.
CUOTAS	Tabla principal del sistema, contiene las cuotas que componen la cta.cte de cada contribuyente. En la misma se definen datos como el tributo, vencimientos, saldo, datos del padrón y titular.
DRI_BOLETA	Almacena los datos de liquidación de cada cuota de DRel, define su composición y adicionales.

DRI_BOLETA_ACTIVIDADES	Contiene el detalle de la actividad, declaración de Base Imponible, alícuotas y mínimo para cada detalle de la Boleta de DRel.
DRI_CUOTA_ACTIVIDAD	Definición de Actividades y detalle de alícuotas y montos mínimos de cada actividad para cada padrón/período emitidos por el Municipio.
DRI_ESTUDIO	Contiene los datos de cada estudio contable definido en el sistema. Datos de acceso y referencias.
DRI_ESTUDIO_PADRON	Detalla los Padrones a cargo de cada Estudio Contable.
WEB_LIQUIDACION	Contiene la liquidación de cuotas (conjunto) generada por los contribuyentes con el saldo actualizado al día de la fecha. Utilizada para el Pago ONLine de un conjunto de cuotas.
WEB_LIQUIDACION_CUOTAS	Detalle de las cuotas contenidas en cada liquidación WEB, define importe original e intereses calculados.
SUSCRIPTORES	Listado de Padrones que están suscritos a la boleta electrónica (evita el envío de la boleta física).
CONFIGURACION	Definición de parámetros de configuración del sitio del Municipio. Define la información que muestra tanto en el sitio como en las boletas. Utilizado también para el cálculo de las boletas de DRel y definición de variables globales de cálculo.
CONFIGURACION_VARS	Variables de configuración dinámicas, sin necesidad de modificar los campos de la Base de Datos (por ejemplo cuando se hace un convenio con una nueva entidad de Pago).

Para la interacción con dichas tablas, no fue necesario utilizar SQL directo (con la excepción de alguna consulta específica) sino mediante el módulo ORM (Mapeo Objeto Relacional) de Django. El mismo brinda una capa de abstracción que permite definir y consultar a través de código Python las tablas, relaciones e índices, los cuales son interpretados y transformados en código SQL.

El ORM [10] permite el cambio de modelo, pudiendo añadir, modificar y eliminar campos con sólo reescribir el código python, donde luego será interpretado como sentencias de tipo DDL en el motor de base de datos (CREATE/UPDATE/DELETE).

Así, por ejemplo la tabla **DRI_ESTUDIO** queda definida en el archivo *models.py* de la siguiente manera:

```
class DriEstudio(models.Model):
    id_estudioc = models.IntegerField(primary_key=True)
    denominacion = models.CharField(db_column='Denominacion', max_length=100, null=True)
    usuario = models.CharField(max_length=30, blank=True, null=True)
    clave = models.CharField(max_length=30, blank=True, null=True)
    numero = models.CharField(max_length=30, null=True)
    email = models.CharField(max_length=100, blank=True, null=True)
    class Meta:
        db_table = 'dri_estudio'
    def __str__(self):
        return u'%s - %s' % (self.numero, self.denominacion)
```

y es instanciada como un objeto de clase, lo cual permite acceder a campos y métodos definidos en el mismo. Por ejemplo la sentencia “*DriEstudio.objects.get(id_estudioc=9)*” es interpretada como una consulta SQL de tipo:

*“Select * from dri_estudio where id_estudioc = 9”.*

Esta función brindada por el ORM no sólo agiliza el manejo de objetos para el desarrollo, además ofrece una capa de abstracción con respecto al motor de base de datos utilizado, lo cual hace el código SQL resultante apto para diferentes Gestores de Bases de Datos, como por ejemplo PostgreSQL, Oracle, SQLite, SQL Server, etc.

Capítulo 4 - Funcionalidades

El Capítulo describe las funcionalidades del sistema. Se plantea un esquema general del sitio y se describen los módulos que lo componen, definiendo en cada caso cómo están definidas las pantallas y las salidas del mismo.

4.1. Esquema del Sitio

El siguiente esquema muestra cómo fue planteado el sistema [12] para cumplir con los requerimientos y refleja las relaciones entre las páginas descritas en la sección siguiente:

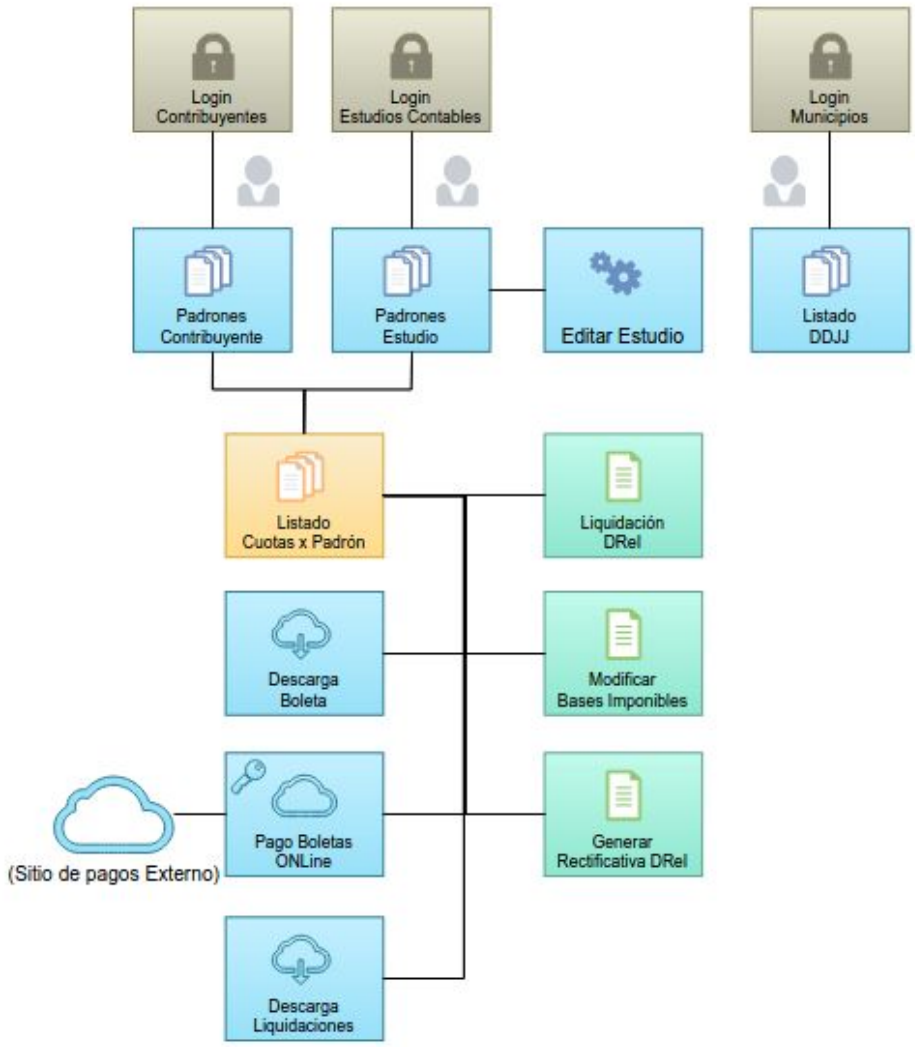


Fig.5 - Mapa del Sitio Web de Autogestión.

4.2. Funcionalidades del Sistema

A continuación se detallan las funcionalidades, ingresos y salidas de cada pantalla del sistema:

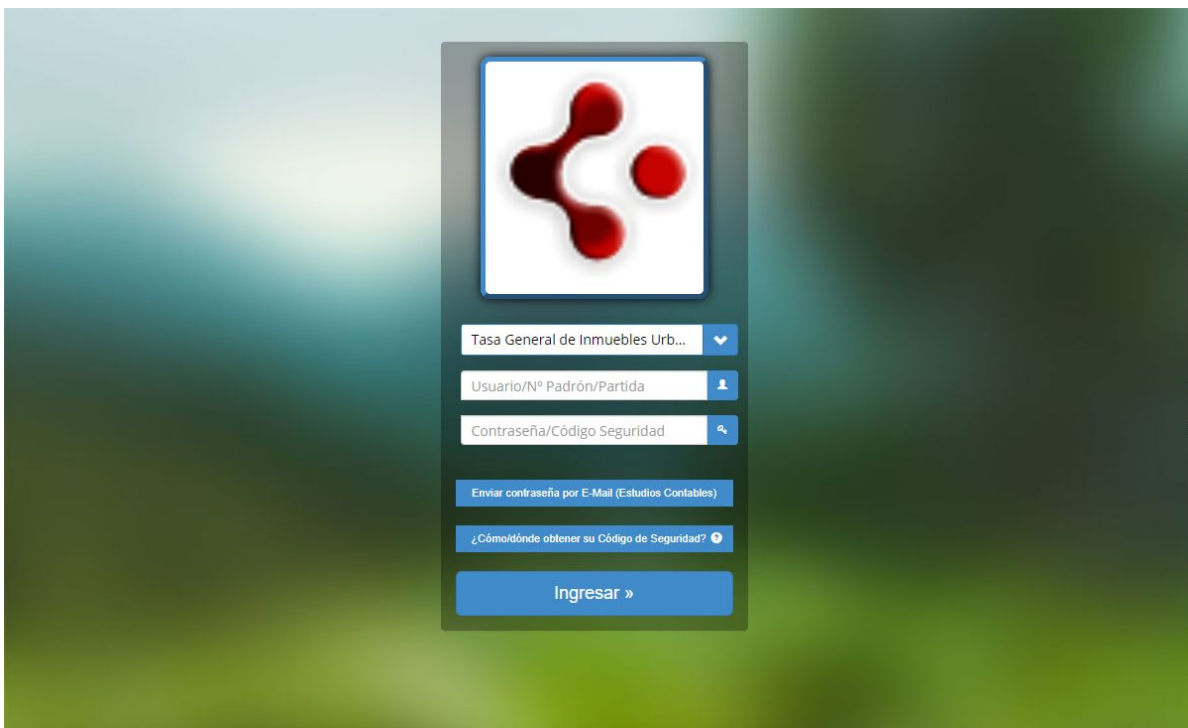
Logueo/Validación de ingreso según sea Estudio Contable, Contribuyente o Municipio	
Detalle	<p>Valores a ingresar para la validación del Logueo(puede configurarse la cantidad de datos requeridos). Según sea el tipo de usuario los datos requeridos son:</p> <ul style="list-style-type: none">● Contribuyente: Tributo/Nº Padrón/Cód.Seguridad● Estudio Contable: Usuario/Contraseña● Municipio: Usuario/Contraseña <p>El logo y el listado de tributos son configurables según el Municipio. Si el estudio no recuerda el código, el mismo puede ser enviado por Email.</p>
	

Fig.6 - Login Sistema.

Consulta de Padrones del Contribuyente

Detalle

Al ingresar, el contribuyente visualiza el listado de padrones a su nombre, pudiendo obtener el listado de cuotas del mismo (cuenta corriente). En la grilla existe una búsqueda rápida para poder filtrar por el padrón o tributo deseado.

N° Padrón/Partida	Tributo	Total Cuotas	Ver Cuotas
	Tasa General de Inmuebles Urbana (TGIU)	14 cuotas	
	ERI (IRE)	12 cuotas	

Fig.7 - Consulta Padrones del Contribuyente.

Consulta de Padrones de DRel a cargo de un Estudio Contable

Detalle

Al ingresar, el estudio contable visualiza el listado de padrones a su cargo (relación del Estudio con sus Clientes), lo que agiliza la búsqueda y liquidación de los períodos de DRel. Asimismo puede seleccionar alguno para ver el estado de su cuenta corriente. En la grilla existe una búsqueda rápida para poder filtrar por el padrón o responsable deseado.

The screenshot displays a web application interface for consulting DRel records. The main area is titled 'Listado Completo de Padrones DRel a cargo del Estudio ESTUDIO EJEMPLO'. It features a search bar labeled 'Buscar Padrón/Representante:'. Below the search bar is a table with the following data:

N° Padrón/Partida	Responsable	Total Cuotas	Ver Cuotas
033/122	AGUIRRE JORGE OMAR	11 cuotas.	Q
050/155	CHARANI JORGE	11 cuotas.	Q
043/140	FIGUEROA WALTER	11 cuotas.	Q
049/154	GIMONDI SOFIA	11 cuotas.	Q
041/138	MANCINELLI CLAUDIA A	11 cuotas.	Q
034/130	MARCOS MARIA	11 cuotas.	Q
044/146	SALUZZO MARIA ROSA	11 cuotas.	Q
039/136	SEGATORE GRACIELA	11 cuotas.	Q
042/139	SICILIANO LUCAS ARIEL	11 cuotas.	Q

The sidebar on the left contains the following elements:

- Logo: MUNICIPIO GRUPO GUADALUPE
- Navigation: Estudio Contable, 1 - ESTUDIO EJEMPLO, Padrones Encontrados 9
- Information: Información Adicional, Deuda visualizada desde el año 2010 en adelante, Para deudas anteriores al 2010 debe consultar en el municipio.
- Actions: Descargar Instructivo, ¿Cómo obtener el Código?

Fig.8 - Consulta Padrones de DRel del Estudio Contable.

Consulta de Deuda y Estado de cta. cte. de una cuenta/Padrón determinado

Detalle

Se consulta el estado de cuenta corriente (cuotas adeudadas/pagadas) según el Padrón/Año seleccionado. Dicha pantalla puede llevar a acciones tales como Impresión, Pago ONLine, Generar y Rectificar DRel, etc.
Pueden filtrarse las cuotas por Año y/o Padrón.
En todo momento pueden bajarse el manual de procedimientos (PDF).

GRUPO GUADALUPE Inicio Desconectarse / Salir

MUNICIPIO Consultas y autogestión Tasas municipales Trámites On-Line

Listado Completo de Cuotas del padrón 0410U (TGIU)

Buscar: 16 cuotas encontradas.

<input type="checkbox"/>	Año	Cuota	Estado	Vencim.	2ºVencim.	F.Pago	Saldo s/Interés
<input type="checkbox"/>	2020	5	VENCIDO	10/05/2020	10/06/2020		\$111,94
<input type="checkbox"/>	2020	4	VENCIDO	15/05/2020	15/06/2020		\$111,94
<input type="checkbox"/>	2020	3	VENCIDO	25/04/2020	25/05/2020		\$111,94
<input type="checkbox"/>	2020	2	VENCIDO	10/03/2020	10/04/2020		\$111,94
<input type="checkbox"/>	2020	1	VENCIDO	10/02/2020	10/03/2020		\$111,94
<input type="checkbox"/>	2019	11	VENCIDO	10/12/2019	10/01/2020		\$111,94
<input type="checkbox"/>	2019	9	VENCIDO	10/10/2019	10/11/2019		\$111,94
<input type="checkbox"/>	2019	7	VENCIDO	10/08/2019	10/09/2019		\$111,94
<input type="checkbox"/>	2019	5	VENCIDO	24/06/2019	24/07/2019		\$200,00
<input type="checkbox"/>	2019	4	VENCIDO	24/05/2019	24/06/2019		\$200,00
<input type="checkbox"/>	2019	3	VENCIDO	24/04/2019	24/05/2019		\$200,00
<input type="checkbox"/>	2019	2	VENCIDO	24/03/2019	24/04/2019		\$201,00
<input type="checkbox"/>	2019	1	VENCIDO	24/02/2019	24/03/2019		\$200,00
<input type="checkbox"/>	2019	1	VENCIDO	10/02/2019	10/03/2019		\$111,94
<input type="checkbox"/>	2018	12	VENCIDO	24/01/2019	24/02/2019		\$200,00
<input type="checkbox"/>	2018	11	VENCIDO	10/12/2018	10/01/2019		\$111,94

Todos Todos Todos \$2320.40 \$0.00

Liquidación - Pago ONLine Cajero 24
Cuotas Seleccionadas: 0
Total sin Punitivos: \$ 0.00
Generar Liquidación
Generar Pago


Información Adicional
Deuda visualizada desde el año 2010 en adelante
Para deudas anteriores al 2010 debe consultar en el municipio.
Descargar Instructivo
¿Cómo obtener el Código?

Fig.9 - Consulta de Cuotas del Padrón seleccionado.

Impresión actualizada según punitorios al día de la fecha

Detalle

Cada cuota puede ser descargada al instante en formato .PDF con el saldo actualizado al día de la fecha. La misma está homologada por entidades de cobro tales como bancos (BNA, Santa Fé, Macro, Credicoop), mutuales y de pagos ONLine (Plus Pagos, Rapipago, Cajero24, PagAR, Banelco).




Comuna de PRUEBA
 Dirección: xxxxxxx - Santa Fe
 Tel: 0999999


Contribuyente
Fecha Impresión
05/10/2020
Nro. Cuota
166010

Boleta Tasa General de Inmuebles Urbana

Padrón/Cuenta	Razón Social	Documento	Tributo
0410U	PAGLIARICI GERARDO L		TGIU
Año	Periodo	Vencimiento	Importe Orig.
2020	5	10/05/2020	\$ 111,94
1er. Vencimiento	Fecha	Punitorios	Importe
	10/10/2020	\$ 10.97	\$ 122.91
2do. Vencimiento	Fecha	Punitorios	Importe
	10/10/2020	\$ 10.97	\$ 122.91



99999 00000 11010 20000 00122 91101 02000 00012 29100 00166 01020 20052



Comuna de PRUEBA
 Dirección: xxxxxxx - Santa Fe
 Tel: 0999999

Municipio
Fecha Impresión
05/10/2020
Nro. Cuota
166010

Boleta Tasa General de Inmuebles Urbana

Padrón/Cuenta	Razón Social	Documento	Tributo
0410U	PAGLIARICI GERARDO L		TGIU
Año	Periodo	Vencimiento	Importe Orig.
2020	5	10/05/2020	\$ 111,94
1er. Vencimiento	Fecha	Punitorios	Importe
	10/10/2020	\$ 10.97	\$ 122.91
2do. Vencimiento	Fecha	Punitorios	Importe
	10/10/2020	\$ 10.97	\$ 122.91




Fig.10 - Boleta PDF Generada por el Sistema.

Generación de Liquidación ONLine de un conjunto de cuotas según padrón

Detalle

Pueden generarse Liquidaciones (conjunto de varias cuotas de períodos no necesariamente consecutivos) para ser pagadas ONLine o impresas dentro de una única boleta de pago (punitorios al día de la fecha).

GRUPO GUARDALUPE

MUNICIPIO
GRUPO GUARDALUPE
Consultas y autogestión | Trámites On-Line

Inicio | Desconectarse / Salir

Listado Completo de Cuotas del padrón 0410U (TGIU)

Buscar: 16 cuotas encontradas.

<input type="checkbox"/>	Año	Cuota	Estado	Vencim.	2ºVencim.	F.Pago	Saldo sin interés
<input checked="" type="checkbox"/>	2020	5	VENCIDO	10/05/2020	10/06/2020		\$111,94
<input checked="" type="checkbox"/>	2020	4	VENCIDO	15/05/2020	15/06/2020		\$111,94
<input checked="" type="checkbox"/>	2020	3	VENCIDO	25/04/2020	25/05/2020		\$111,94
<input checked="" type="checkbox"/>	2020	2	VENCIDO	10/03/2020	10/04/2020		\$111,94
<input checked="" type="checkbox"/>	2020	1	VENCIDO	10/02/2020	10/03/2020		\$111,94
<input type="checkbox"/>							\$111,94
<input type="checkbox"/>							\$111,94
<input type="checkbox"/>							\$111,94
<input type="checkbox"/>							\$200,00
<input type="checkbox"/>							\$200,00
<input type="checkbox"/>							\$200,00
<input type="checkbox"/>							\$201,00
<input type="checkbox"/>	2019	2	VENCIDO	24/03/2019	24/04/2019		\$201,00
<input type="checkbox"/>	2019	1	VENCIDO	24/02/2019	24/03/2019		\$200,00
<input type="checkbox"/>	2019	1	VENCIDO	10/02/2019	10/03/2019		\$111,94
<input type="checkbox"/>	2018	12	VENCIDO	24/01/2019	24/02/2019		\$200,00
<input type="checkbox"/>	2018	11	VENCIDO	10/12/2018	10/01/2019		\$111,94

Liquidación OnLine

El monto total actualizado de la Liquidación es de : \$ 626.75

Guardar e Imprimir | Cancelar

Información Adicional
Deuda visualizada desde el año 2010 en adelante.
Para deudas anteriores al 2010 debe consultar en el municipio.

Descargar Instructivo

¿Cómo obtener el Código?

Todos | Todos | Todos | \$2320.40 \$0.00

Fig.11 - Generación de una Liquidación ONLine (varias cuotas).



Comuna de PRUEBA
Dirección: xxxxxx - Santa Fe
Tel: 0999999

N° Liquidación
18
Fecha Impresión
05/10/2020

BOLETA LIQUIDACION

N° Unidad	N° Padrón	Razon Social
331	0410U	
Vencimiento	Concepto	Domicilio
10/10/2020	TGIU	
Nominal	Acc. x Mora	Total a Pagar
\$ 559,70	\$ 67,05	\$ 626,75
Cantidad Cuotas	Detalle Cuotas a Pagar	
5	1/2020 - 2/2020 - 3/2020 - 4/2020 - 5/2020	



99999 00099 81010 20000 00626 75101 02000 00062 67500 00000 01820 20102

N° Unidad	N° Padrón	Razon Social
331	0410U	
Vencimiento	Concepto	Domicilio
10/10/2020	TGIU	
Nominal	Acc. x Mora	Total a Pagar
\$ 559,70	\$ 67,05	\$ 626,75
Cantidad Cuotas	Detalle Cuotas a Pagar	
5	1/2020 - 2/2020 - 3/2020 - 4/2020 - 5/2020	



99999 00099 81010 20000 00626 75101 02000 00062 67500 00000 01820 20102

N° Unidad	N° Padrón	Razon Social
331	0410U	

Fig.12 - Boleta PDF de la Liquidación ONLine (varias cuotas).

Generación e Impresión de la Declaración Jurada Mensual de un Padrón de Derecho y Registro de Inspección (DRel)

Detalle

Pueden liquidarse los períodos de DRel de acuerdo a los requerimientos de cada Padrón (se declaran las bases imponibles correspondientes con cada actividad y, de acuerdo a su alícuota y adicionales, se generan los montos que conformarán el importe final de la boleta).

Una vez generada puede ser impresa, editada o seleccionada para su posterior pago ONLINE.

MUNICIPIO GRUPO GUADALUPE
Consultas y autogestión | Trámites On-Line

Inicio | Desconectarse / Salir

Padrón: 000/000 Autoliquidación de Boletas de DRel Período: 12 / 2020 Vencimiento: 15/01/2021

Detalle Actividad	Base	Alícuota 0/100	Mínimo Exigible	Total
(*) 171 - VENTA DE ARTICULOS PARA EL HOGAR	\$ 100000	0,01	\$ 100,00	\$1000,00
57 - FERRERIA	\$ 35000	0,01	\$ 0	\$350,00
105 - PINTURERIA	\$ 21500	0,01	\$ 0	\$215,00
177 - VENTA DE CARBON	\$ 0	0,01	\$ 0	\$0
178 - VENTA DE CIGARRILLOS	\$ 0	0,01	\$ 0	\$0
Mínimo Boleta: \$ 100,00				SubTotal: \$1565,00

Derecho Neto \$ 20

Tasa Salud Pública \$ 0

Sin Adicional \$ 0

Retenciones \$ 0

Recargo Punitivos \$ 0

Total a Pagar: \$ 1585,00

Cancelar y Volver Guardar

Información Adicional
Deuda visualizada desde el año 2010 en adelante
Para deudas anteriores al 2010 debe consultar en el municipio.

Descargar Instructivo

¿Cómo obtener el Código?

Fig.13 - Generación de la Cuota de DRel según declaración de Bases Imponibles.

Modificación de Bases imponibles para un período ya liquidado de DRel

Detalle

Se pueden editar las bases imponibles de los períodos de DRel ya liquidados estén o no pagados (útil para la DDJJ Anual). En todo momento el estudio/contribuyente puede modificar las bases imponibles ya declaradas en períodos anteriores.

The screenshot shows a web application interface for 'Reliquidar Boleta de DRel'. The header includes the logo for 'GRUPO GUADALUPE MUNICIPIO' and navigation links for 'Inicio' and 'Desconectarse / Salir'. The main content area displays a table with columns for 'Detalle Actividad', 'Base', 'Alícuota 000', 'Mínimo Exigible', and 'Total'. The table lists several activities with their respective base amounts and tax rates. Below the table, there are input fields for 'Derecho Neto', 'Tasa Salud Pública', 'Sin Adicional', 'Retenciones', and 'Recargo/Punitivos'. The total amount to be paid is shown as '\$ 1485.00'. A 'Cancelar y Volver' button is located at the bottom left, and a 'Guardar' button is at the bottom right.

Detalle Actividad	Base	Alícuota 000	Mínimo Exigible	Total
(*) 171 - VENTA DE ARTICULOS PARA EL HOGAR	\$ 90000	0,01	\$ 100,00	\$900,00
57 - FERRETERIA	\$ 35000,00	0,01	\$ 0,00	\$350,00
105 - PINTURERIA	\$ 21500,00	0,01	\$ 0,00	\$215,00
177 - VENTA DE CARBON	\$ 0,00	0,01	\$ 0,00	\$0
178 - VENTA DE CIGARRILLOS	\$ 0,00	0,01	\$ 0,00	\$0
Mínimo Boleta: \$ 100,00				SubTotal: \$1465,00
				Derecho Neto \$ 20
				Tasa Salud Pública \$ 0
				Sin Adicional \$ 0
				Retenciones \$ 0
				Recargo/Punitivos \$ 0
				Total a Pagar: \$ 1485,00

Fig.14 - Modificación de Bases Imponibles de una cuota de DRel existente.

Rectificativa de períodos liquidados y pagos de DRel

Detalle

Permite generar rectificativas de períodos de DRel ya generados y pagados erróneamente (valor inferior al que deberían haber declarado).
Generalmente esto ocurre cuando un contribuyente declaró y pagó un monto incorrecto, con lo cual se genera una boleta adicional de pago (ingresando como crédito a favor el importe ya abonado).

Detalle Actividad	Base	Alicuota 0/00	Mínimo Exigible	Total
(*) 171 - VENTA DE ARTICULOS PARA EL HOGAR	\$ 120000	0,01	\$ 100,00	\$1200,00
57 - FERRETERIA	\$ 45000	0,01	\$ 0	\$450,00
105 - PINTURERIA	\$ 18500	0,01	\$ 0	\$185,00
177 - VENTA DE CARBON	\$ 0	0,01	\$ 0	\$0
178 - VENTA DE CIGARRILLOS	\$ 0	0,01	\$ 0	\$0
Mínimo Boleta:	\$ 100,00			SubTotal: \$1835,00

Importe ya Abonado (boleta a Rectificar) \$ 0
 Derecho Neto \$ 0
 Tasa Salud Pública \$ 0
 Sin Adicional \$ 0
 Devoluo/Reteno \$ 0
 Recargo/Punitivos \$ 0
Total a Pagar: \$ 1835,00

Fig.15 - Rectificativa de boleta de DRel existente y saldada.

Generación de Ticket para Pago ONLINE de un conjunto de cuotas seleccionado

Detalle

Pago ONLINE de cuotas de acuerdo a convenios con diferentes entidades que brinden dicho servicio (por ejemplo Cajero24, PlusPagos, BicaÁgil, etc) con sus distintas formas de pago (Débito, Crédito en Cuotas, etc).
Al seleccionar las cuotas y clicar en "Pagar", el usuario será redireccionado al sitio de pago de la entidad.

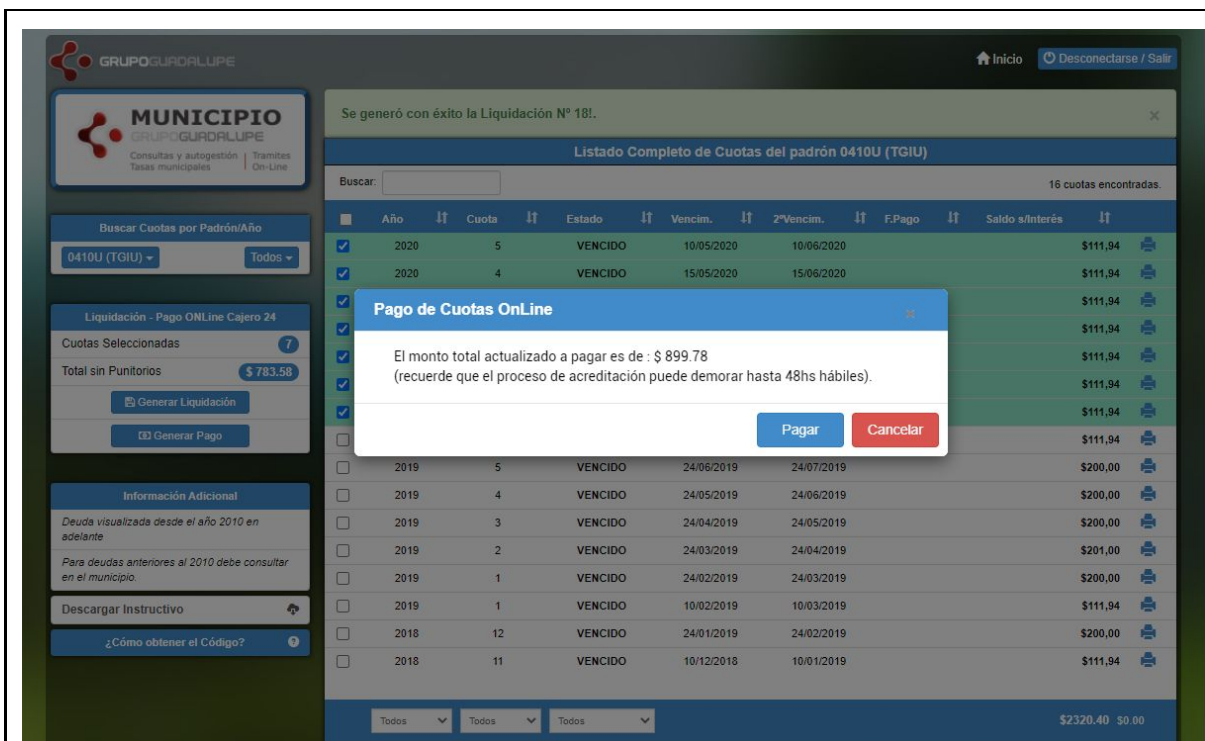


Fig.16 - Generación de Ticket de Pago ONLine.

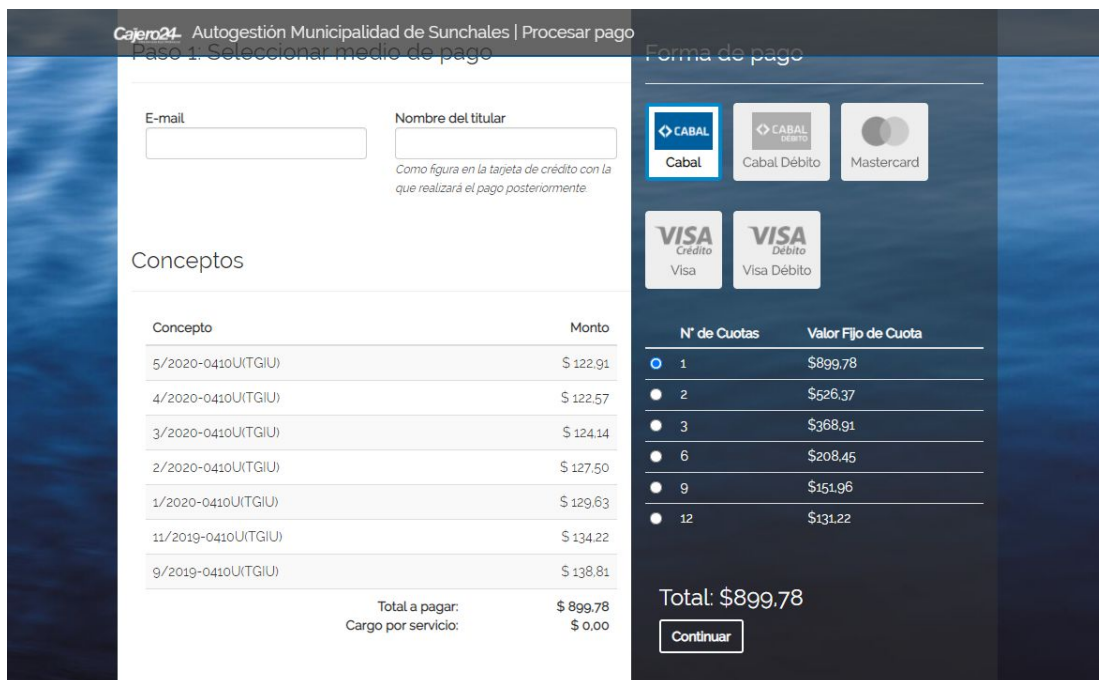


Fig.17 - Redireccionamiento a Sitio web de Pago ONLine (en este caso Cajero24).

API Consulta de Padrones/Cuotas

Detalle

Interfaz de consulta tipo REST de Cuotas, filtradas por Padrón/Año/Estudio Contable.
Al enviar el requerimiento GET/POST/PUT se obtiene una respuesta JSON con el resultado del mismo, en este caso un arreglo de Cuotas.

Solicitud: GET <https://muniNN.boletaweb.com.ar/api/v1/cuotas/9999/2020/>
Respuesta:

```
{
  "data": [
    {
      "id_cuota": 164648,
      "cuota": 9,
      "anio": 2020,
      "tributo_nombre": "DRI",
      "tributo_abreviatura": "DRI",
      "estado_nombre": "NORMAL",
      "id_responsable": 1051,
      "id_unidad": 331,
      "saldo": "1800.00",
      "vencimiento": "2020-10-15",
      "id_padron": "1961",
      "padron": "000/000",
      "fechapago": null,
      "estado": 0,
      "segundo_vencimiento": "2020-11-15",
      "nombre": null,
      "nombre_boleta": null,
      "nrodocu": null,
      "sexo": null,
      "calle": null,
      "numero": null,
      "piso": null,
      "depto": null,
      "localidad": null,
      "codseg": null,
      "fecha_audit": "2020-05-24T21:18:35.897831Z",
      "tributo": 6
    },
    {
      "id_cuota": 164639,
      "cuota": 9,
      "anio": 2020,
      "tributo_nombre": "Tasa General de Inmuebles Urbana",
      "tributo_abreviatura": "TGIU",
      "estado_nombre": "NORMAL",
      "id_responsable": 1168,
      "id_unidad": 92,
    }
  ]
}
```

Fig.18 -Ejemplo de Petición/Respuesta de la API.

Consulta de DDJJ Realizadas según Contribuyente/Padrón (para el Municipio)

Detalle

Listado de Períodos liquidados por cada contribuyente, detallando sus bases imponibles, alícuotas, mínimos y adicionales según sea el caso.
Sirve para realizar la DDJJ Anual y visualizar lo informado por cada contribuyente.

Año	Mes	Fecha Carga	Der Año	Tasa Salud PNM	Tipo Adic.	Monto Adic.	Retenciones	Mínimo Global	Recargos	Total
2016	10	13/09/2016	\$0	\$0		\$0	\$0	\$100,00	\$0	\$100,00
2016	11	25/09/2016	\$0,00	\$0,00	Adicional 10%	\$179,01	\$-562,00	\$100,00	\$13,957,02	\$62,251,19
2019	1	27/08/2019	\$100,00	\$0,00	Adicional 10%	\$123,46	\$0,00	\$120,00	\$19,712,72	\$19,213,91
2019	7	12/12/2019	\$0,00	\$0,00	Adicional 10%	\$305,916,13	\$0,00	\$120,00	\$439,169,01	\$4,524,187,00
2019	8	17/12/2019	\$0	\$0	Adicional 10%	\$0	\$0	\$120,00	\$0	\$141,191,00
2019	9	16/12/2019	\$0,00	\$0,00		\$0,00	\$0,00	\$120,00	\$13,40	\$713,40
2016	10	13/10/2016	\$0,00	\$0,00		\$0,00	\$0,00	\$100,00	\$16,00	\$116,00
2019	11	09/04/2020	\$0,00	\$0,00	Adicional 10%	\$65,75	\$0,00	\$100,00	\$72,33	\$795,62
2016	8	13/04/2016	\$0,00	\$1,556,36	Sin Adicional	\$0,00	\$0,00	\$100,00	\$78,320,04	\$293,508,53
2020	1	26/03/2020	\$0,00	\$0,00	Adicional 10%	\$26,10	\$0,00	\$100,00	\$3,21	\$806,94
2020	3	02/05/2020	\$0,00	\$0,00		\$0,00	\$0,00	\$100,00	\$16,65	\$518,65
2020	2	26/03/2020	\$0	\$0	Adicional 2%	\$0	\$0	\$100,00	\$0	\$256,00
2019	12	15/04/2019	\$1,000,00	\$0,00	Adicional 10%	\$115,50	\$0,00	\$100,00	\$147,23	\$2,412,23
2020	3	28/04/2020	\$0,00	\$0,00	Adicional 2%	\$4,00	\$0,00	\$100,00	\$13,55	\$103,55
2020	3	25/04/2020	\$0,00	\$0,00		\$0,00	\$0,00	\$100,00	\$26,91	\$712,41
2020	3	01/05/2020	\$0,00	\$0,00		\$0,00	\$0,00	\$100,00	\$0,04	\$1,04
2020	12	05/10/2020	\$20,00	\$0,00		\$0,00	\$0,00	\$100,00	\$0,00	\$1,485,00

Fig.19 - Consulta de DDJJ generadas.

Padrón	Año	Mes	Contribuyente	F.Carga	Der Neto	Tasa PNM	Adicional	Monto Adic.	Retenciones	Mín. Global	Recargos	Total																																																
000000	2.020	1		26/03/2020	\$0,00	\$0,00	Adicional 10%	\$26,10	\$0,00	\$120,00	\$3,21	\$106,94																																																
<table border="1"> <thead> <tr> <th>Actividad</th> <th>Base</th> <th>Alícuota 00%</th> <th>Impuesto</th> </tr> </thead> <tbody> <tr><td>(*) 171 - VENTA DE ARTICULOS PARA EL HOGAR</td><td>\$25.000,00</td><td>0,01</td><td>\$250,00</td></tr> <tr><td>15 - ARTICULOS DE COTILLON</td><td>\$62.000,00</td><td>0,01</td><td>\$620,00</td></tr> <tr><td>57 - FERRERIA</td><td>\$0,00</td><td>0,01</td><td>\$0,00</td></tr> <tr><td>100 - PINTURERIA</td><td>\$0,00</td><td>0,01</td><td>\$0,00</td></tr> <tr><td>105 - PINTURERIA</td><td>\$0,00</td><td>0,01</td><td>\$0,00</td></tr> <tr><td>105 - PINTURERIA</td><td>\$0,00</td><td>0,01</td><td>\$0,00</td></tr> <tr><td>107 - PLUMBERO</td><td>\$0,00</td><td>0,01</td><td>\$0,00</td></tr> <tr><td>109 - PROVISION DE AGUA POTABLE</td><td>\$0,00</td><td>0,01</td><td>\$0,00</td></tr> <tr><td>152 - TAPISERIA</td><td>\$0,00</td><td>0,01</td><td>\$0,00</td></tr> <tr><td>177 - VENTA DE CARBON</td><td>\$0,00</td><td>0,01</td><td>\$0,00</td></tr> <tr><td>178 - VENTA DE CIGARRILLOS</td><td>\$0,00</td><td>0,01</td><td>\$0,00</td></tr> </tbody> </table>													Actividad	Base	Alícuota 00%	Impuesto	(*) 171 - VENTA DE ARTICULOS PARA EL HOGAR	\$25.000,00	0,01	\$250,00	15 - ARTICULOS DE COTILLON	\$62.000,00	0,01	\$620,00	57 - FERRERIA	\$0,00	0,01	\$0,00	100 - PINTURERIA	\$0,00	0,01	\$0,00	105 - PINTURERIA	\$0,00	0,01	\$0,00	105 - PINTURERIA	\$0,00	0,01	\$0,00	107 - PLUMBERO	\$0,00	0,01	\$0,00	109 - PROVISION DE AGUA POTABLE	\$0,00	0,01	\$0,00	152 - TAPISERIA	\$0,00	0,01	\$0,00	177 - VENTA DE CARBON	\$0,00	0,01	\$0,00	178 - VENTA DE CIGARRILLOS	\$0,00	0,01	\$0,00
Actividad	Base	Alícuota 00%	Impuesto																																																									
(*) 171 - VENTA DE ARTICULOS PARA EL HOGAR	\$25.000,00	0,01	\$250,00																																																									
15 - ARTICULOS DE COTILLON	\$62.000,00	0,01	\$620,00																																																									
57 - FERRERIA	\$0,00	0,01	\$0,00																																																									
100 - PINTURERIA	\$0,00	0,01	\$0,00																																																									
105 - PINTURERIA	\$0,00	0,01	\$0,00																																																									
105 - PINTURERIA	\$0,00	0,01	\$0,00																																																									
107 - PLUMBERO	\$0,00	0,01	\$0,00																																																									
109 - PROVISION DE AGUA POTABLE	\$0,00	0,01	\$0,00																																																									
152 - TAPISERIA	\$0,00	0,01	\$0,00																																																									
177 - VENTA DE CARBON	\$0,00	0,01	\$0,00																																																									
178 - VENTA DE CIGARRILLOS	\$0,00	0,01	\$0,00																																																									
000000	2.020	3		26/03/2020	\$0,00	\$0,00		\$0,00	\$0,00	\$100,00	\$191,25	\$101,25																																																
<table border="1"> <thead> <tr> <th>Actividad</th> <th>Base</th> <th>Alícuota 00%</th> <th>Impuesto</th> </tr> </thead> <tbody> <tr><td>(*) 171 - VENTA DE ARTICULOS PARA EL HOGAR</td><td>\$50.000,00</td><td>0,01</td><td>\$500,00</td></tr> <tr><td>57 - FERRERIA</td><td>\$25.000,00</td><td>0,01</td><td>\$250,00</td></tr> <tr><td>100 - PINTURERIA</td><td>\$0,00</td><td>0,01</td><td>\$0,00</td></tr> <tr><td>177 - VENTA DE CARBON</td><td>\$0,00</td><td>0,01</td><td>\$0,00</td></tr> <tr><td>178 - VENTA DE CIGARRILLOS</td><td>\$0,00</td><td>0,01</td><td>\$0,00</td></tr> </tbody> </table>													Actividad	Base	Alícuota 00%	Impuesto	(*) 171 - VENTA DE ARTICULOS PARA EL HOGAR	\$50.000,00	0,01	\$500,00	57 - FERRERIA	\$25.000,00	0,01	\$250,00	100 - PINTURERIA	\$0,00	0,01	\$0,00	177 - VENTA DE CARBON	\$0,00	0,01	\$0,00	178 - VENTA DE CIGARRILLOS	\$0,00	0,01	\$0,00																								
Actividad	Base	Alícuota 00%	Impuesto																																																									
(*) 171 - VENTA DE ARTICULOS PARA EL HOGAR	\$50.000,00	0,01	\$500,00																																																									
57 - FERRERIA	\$25.000,00	0,01	\$250,00																																																									
100 - PINTURERIA	\$0,00	0,01	\$0,00																																																									
177 - VENTA DE CARBON	\$0,00	0,01	\$0,00																																																									
178 - VENTA DE CIGARRILLOS	\$0,00	0,01	\$0,00																																																									
000000	2.020	2		26/03/2020	\$0	\$0	Adicional 2%	\$0	\$0	\$100,00	\$0	\$ 256,00																																																
<table border="1"> <thead> <tr> <th>Actividad</th> <th>Base</th> <th>Alícuota 00%</th> <th>Impuesto</th> </tr> </thead> <tbody> <tr><td>(*) 171 - VENTA DE ARTICULOS PARA EL HOGAR</td><td>\$25.000,00</td><td>0,01</td><td>\$250,00</td></tr> <tr><td>15 - ARTICULOS DE COTILLON</td><td>\$324.324.324,00</td><td>0,01</td><td>\$3.243.243,24</td></tr> <tr><td>57 - FERRERIA</td><td>\$0,00</td><td>0,01</td><td>\$0,00</td></tr> <tr><td>100 - PINTURERIA</td><td>\$0,00</td><td>0,01</td><td>\$0,00</td></tr> <tr><td>105 - PINTURERIA</td><td>\$0,00</td><td>0,01</td><td>\$0,00</td></tr> <tr><td>105 - PINTURERIA</td><td>\$0,00</td><td>0,01</td><td>\$0,00</td></tr> </tbody> </table>													Actividad	Base	Alícuota 00%	Impuesto	(*) 171 - VENTA DE ARTICULOS PARA EL HOGAR	\$25.000,00	0,01	\$250,00	15 - ARTICULOS DE COTILLON	\$324.324.324,00	0,01	\$3.243.243,24	57 - FERRERIA	\$0,00	0,01	\$0,00	100 - PINTURERIA	\$0,00	0,01	\$0,00	105 - PINTURERIA	\$0,00	0,01	\$0,00	105 - PINTURERIA	\$0,00	0,01	\$0,00																				
Actividad	Base	Alícuota 00%	Impuesto																																																									
(*) 171 - VENTA DE ARTICULOS PARA EL HOGAR	\$25.000,00	0,01	\$250,00																																																									
15 - ARTICULOS DE COTILLON	\$324.324.324,00	0,01	\$3.243.243,24																																																									
57 - FERRERIA	\$0,00	0,01	\$0,00																																																									
100 - PINTURERIA	\$0,00	0,01	\$0,00																																																									
105 - PINTURERIA	\$0,00	0,01	\$0,00																																																									
105 - PINTURERIA	\$0,00	0,01	\$0,00																																																									

Fig.20 - Consulta de DDJJ y sus detalles.

Suscripción/Desuscripción de Contribuyentes a “Boleta Digital”

Detalle

Cada contribuyente tiene la posibilidad de suscribirse a la boleta digital, generando un ahorro en papel y logística para el Municipio.
El contribuyente obtiene así una bonificación en el importe de su cuota.

Año	Cuota	Estado	Vencim.	2º Vencim.	F. Pago	Saldo a cobrar
2020	5	VENCIDO	10/05/2020	10/05/2020		\$111,54
2020	4	VENCIDO	15/06/2020	15/06/2020		\$111,54
2020	3	VENCIDO	25/04/2020	25/05/2020		\$111,54
2020	2	VENCIDO	10/03/2020	10/04/2020		\$111,54
2020	1	VENCIDO	10/03/2020	10/03/2020		\$111,54
2019	11	VENCIDO	10/12/2019	10/01/2020		\$111,54
2019	9	VENCIDO	10/10/2019	10/10/2019		\$111,54
2019	7	VENCIDO	10/08/2019	10/09/2019		\$111,54
2019	5	VENCIDO	24/06/2019	24/07/2019		\$200,00
2019	4	VENCIDO	24/05/2019	24/06/2019		\$200,00
2019	3	VENCIDO	24/04/2019	24/05/2019		\$200,00
2019	2	VENCIDO	24/03/2019	24/04/2019		\$200,00
2019	1	VENCIDO	24/02/2019	24/03/2019		\$200,00
2019	1	VENCIDO	10/03/2019	10/03/2019		\$111,54
2018	12	VENCIDO	24/01/2019	24/02/2019		\$200,00
2018	11	VENCIDO	10/12/2018	10/01/2019		\$111,54



Fig.21 - Alta/Baja Suscripción a “Boleta Digital”.

Capítulo 5 - Herramientas

El presente Capítulo tiene como objetivo describir las tecnologías y herramientas utilizadas a lo largo del desarrollo del proyecto.

5.1. Herramientas de Gestión del Proyecto

Para gestionar recursos, historias de usuario, diseño de diagramas, etc fueron utilizadas las siguientes herramientas:

- **JIRA/ASANA:** Utilizados para gestionar incidencias y errores, casos de prueba y planificar Sprints. Sirvió de apoyo para la gestión de requisitos y seguimiento del estado de desarrollo del proyecto. Las historias de usuario fueron convertidas en listas de requerimientos, las cuales se tradujeron en tareas para el equipo de desarrollo.
- **MS Visio / Visual Paradigm:** Utilizados para el diseño de diagramas de Entidad-Relación, Diagrama de Modelado de Tablas, Esquemas de Interacción de Componentes, Mapa del Sitio, etc.

5.2. Herramientas y Tecnologías

En este apartado se describen todas las herramientas y tecnologías utilizadas durante el desarrollo del proyecto. Se divide en herramientas (entornos y aplicaciones), lenguajes y librerías utilizadas.

HERRAMIENTAS UTILIZADAS	
IDE de Desarrollo Python:	SublimeText 3 / VS Code
Gestión de Peticiones:	Postman ⁽¹⁾
Integración Continua:	Travis CI ⁽²⁾
Sistema de Control de Versiones:	GIT 2.11 ⁽³⁾
Aislamiento/Despliegue:	Docker ⁽⁴⁾ / VirtualEnv

- (1) **Postman:** Es un cliente REST que permite la realización de peticiones a una API o Webservice de manera rápida y eficiente. Es de gran utilidad para debuggear las aplicaciones de una manera más sencilla. Por cada petición ofrece una respuesta en diversos formatos y la posibilidad de capturar las Cookies y credenciales de Chrome para realizar peticiones y pruebas de desarrollo.
- (2) **Travis CI:** Servicio de Integración continua distribuido, se encuentra integrado con GitHub. Permite una sencilla configuración e implementación de los proyectos almacenados en repositorios de GitHub y la ejecución de comandos especificados en el archivo de configuración cuando se realiza un Commit y Push. Normalmente estos comandos son construir la aplicación y ejecutar los Tests.
- (3) **GIT:** Es un sistema de control de versiones que permite hacer un seguimiento de los cambios realizados en los archivos de un ordenador y coordinar dichos cambios con los que puedan estar realizando otros usuarios sobre los mismos archivos en sus ordenadores. Git hace énfasis en la velocidad e integridad de los datos y en el apoyo para la distribución de flujos de trabajo no lineales.
- (4) **Docker:** Es una tecnología de virtualización que provee una capa extra de abstracción y automatización a través del uso de “contenedores”. Con ello se evita el uso de grandes máquinas virtuales y se permite la ejecución de instancias de Linux independientes en cada contenedor. Docker automatiza las tareas de configuración y despliegue.

LENGUAJES Y LIBRERÍAS	
BACKEND	FRONTEND
Python ⁽⁵⁾ 2.7.29	Bootstrap 3.5.4
Django ⁽⁶⁾ 1.11.2	JQuery 1.10

Django Rest Framework ⁽⁷⁾ 3.6.3 - JWT	CSS3
Django Extensions	Vanilla JS
Local Flavour	AJAX
Django Crispy Forms	HTML5
Django Secure	
Python Decouple	

- (5) **Python:** Lenguaje de alto nivel creado por Guido Van Rossum en 1991. Es un lenguaje interpretado que hace énfasis en una sintaxis que resulte de fácil lectura e interpretación. Aboga por un código limpio y que permita expresar conceptos complejos en pocas líneas de programación.
- (6) **Django:** Framework (conjunto de librerías y herramientas) Open Source escrito en Python que permite un desarrollo rápido y limpio de aplicaciones WEB. Hace uso de una variante del patrón de diseño MVC (Modelo-Vista-Controlador) que se llama MVT (Modelo-Vista-Template). Hace uso de la reutilización de componentes y del principio DRY (Don't Repeat Yourself).
- (7) **Django REST:** Kit de herramientas que permite la introducción de permisos de autenticación, versionado, CRUD (Create/Update/Delete), JWT (Json Web Token), serialización del modelo de objetos a JSON y multitud de características API de una manera sencilla e intuitiva.

Capítulo 6 - Implementación

El Capítulo describe aspectos relacionados con la implementación del sistema, teniendo en cuenta las funcionalidades desarrolladas, las páginas y accesos de usuarios, la estructura del código fuente y especificando el desarrollo tanto del lado del cliente como del lado del servidor con las tecnologías planteadas y su integración.

6.1. Control de Accesos

Las sesiones son el mecanismo que utiliza Django para guardar registro del "estado" entre el sitio y un navegador en particular. Nos permiten almacenar información arbitraria por navegador, y tener esta información disponible para el sitio cuando el mismo se conecta. Cada pieza individual de información asociada con una sesión se conoce como "clave", que se usa tanto para guardar como para recuperar la información.

El mecanismo que utiliza Django contiene un "*id*" de sesión específica para identificar cada navegador y su sesión asociada con el sitio. La información real de la sesión se guarda por defecto en la base de datos del sitio (esto es más seguro que guardar la información en una cookie, donde es más vulnerable para los usuarios maliciosos). Se puede configurar Django para guardar la información de sesión en otros lugares (caché, archivos, cookies "seguras"), pero la opción por defecto es una buena opción y relativamente segura. Esto nos proporciona un sistema de autenticación y autorización ("permisos"), construido sobre dicho framework de sesión, lo cual permite verificar credenciales de usuario y definir qué acciones pueden realizar. El framework incluye modelos para usuarios, procedimientos y vistas que designan si un usuario puede realizar una tarea y/o para restringir el contenido.

Para el Proyecto, se creó una tabla de "*Perfil*" para cada usuario el cual tiene una relación 1 a 1 con la tabla *Users* manejada por el sistema de Sesiones de Django

[10]. En dicho perfil se almacena si es un usuario de un estudio contable, un contribuyente o el propio Municipio.

Una vez ingresado se le asignan los permisos correspondientes y se verifican en cada una de las acciones a tomar por el usuario (edición, impresión, liquidación, pagos, visualización, etc.).

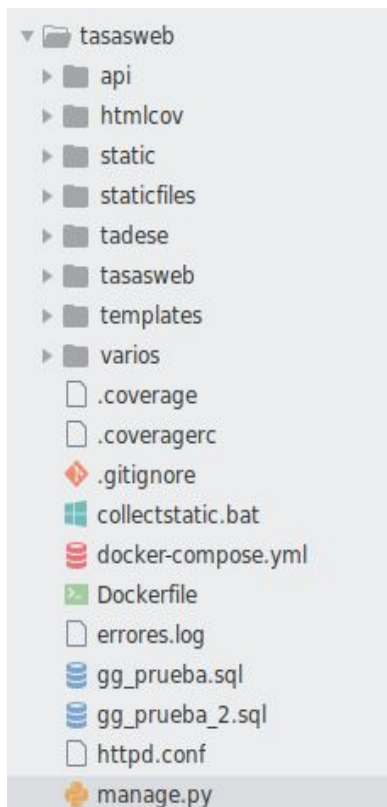
Esta metodología prohíbe que algún usuario realice una acción indebida sobre los datos de otro y/o pueda visualizar información sensible.

El esquema de permisos queda determinado de la siguiente manera:

Tipo Usuario	Visualización	Acciones
Contribuyente	<ul style="list-style-type: none"> ● Padrones/Cuotas propias. 	<ul style="list-style-type: none"> ● Imprimir Cuotas. ● Generar Liquidaciones. ● Pagar Cuotas. ● Liquidar Drel, ● Editar Cuota DRel. ● Rectificar Cuota DRel. ● Modificar Bases Imponibles DRel.
Estudio Contable	<ul style="list-style-type: none"> ● Padrones a su cargo. ● Información del Estudio. 	<ul style="list-style-type: none"> ● (Idem Contribuyente) ● Editar Datos Propios.
Municipio	<ul style="list-style-type: none"> ● Listado Padrones Generales. ● DDJJ Realizadas por Contrib/Estudios. ● Datos Estudios Contables. 	<ul style="list-style-type: none"> ● Editar Datos Estudios. ● Imprimir Listados.

6.2. Estructuración del Proyecto

Para la implementación se estructuró el proyecto conforme a los lineamientos del Framework Django, el cual contempla una estructura de organización predeterminada. A continuación mostramos cómo queda la estructura de directorios de un proyecto base y qué contiene cada una:



- **api**: Aplicación que tiene como fin todo lo referido a la API, urls, vistas y serializadores (modelado de tablas a formato json).
 - **htmlcov**: Archivos html resultantes de ejecutar el comando “coverage”, los cuales nos brindan un informe del estado y el porcentaje funcionalidades cubiertas por tests en el sistema.
 - **static/staticfiles**: En este directorio están almacenados todos los archivos estáticos del proyecto, como lo son los archivos CSS, Javascript, imágenes, entre otros.
 - **tadese**: Aplicación principal del sistema. Contiene la definición del modelo, urls, vistas, tests y funciones auxiliares utilizadas en todo el sistema.
-
- **tasasweb**: Contiene los archivos de configuración general del sistema, urls y vistas de logueo.
 - **templates**: En el directorio de templates están almacenados todos los archivos .HTML, esta es una de las carpetas por excelencia del desarrollador del lado del cliente conocido como Frontend.
 - **varios**: Archivos de configuración y temporales del proyecto.

6.3. Interacción de Componentes y Tecnologías

Se muestra un esquema resumiendo cómo interactúan los componentes desarrollados separando en cada caso el circuito de solicitud de logueo, consulta a la API y sincronización.

6.3.1. Proceso de Sincronización

El Sincronizador (sistema desarrollado en Delphi 7, provisto por la empresa Grupo Guadalupe) es el encargado de ejecutar cada N minutos el proceso de sincronización entre la Base de Datos local Firebird y la Base de Datos ONLine MySQL. Dicho sistema posee un archivo de configuración en donde se detallan parámetros tales como ruta de la base de datos local, servidor remoto, horario de sincronización, etc.

Cuando se ejecuta el proceso, el sincronizador verifica la tabla de logs pendientes a ser subidos hacia la base de datos remota y genera scripts de actualización en formato JSON, los cuales son transferidos al servidor y encolados para su posterior procesamiento. Una vez terminada dicha tarea el contador se reinicia y queda a la espera de nuevas novedades.

6.3.2. Proceso de Interacción con el Sistema

Al ingresar a "<https://muniNN.boletaweb.com.ar>" actúa el ruteo MVC [14] del proyecto, devolviendo la vista correspondiente y validando el perfil del usuario. Una vez renderizada la vista en el cliente, se carga tanto el HTML propio de la vista, los CSS (estilos), y el JavaScript que se encargará de realizar las peticiones al servidor para traer los datos necesarios. Una vez cargado los archivos estáticos (HTML y JS), el usuario está en condiciones de interactuar con el sistema, ya sea en el logueo o solicitando datos al mismo.

6.3.3. Proceso de Interacción con la API

La API desarrollada tiene diferentes “puntos de ingreso” según sea la solicitud (ver sección “*Extensibilidad del Sistema*”).

El acceso a la misma se realiza mediante una petición GET/POST bajo el Protocolo HTTPS v1.1. Un ejemplo del mismo sería solicitar las cuotas de un padrón y año específico:

<https://muniNN.boletaweb.com.ar/api/v1/cuotas/PPPP/AAAA/>

en donde PPPP es el id_padron y AAAA es el año solicitado.

La respuesta es un archivo JSON con el resultado de la operación solicitada.

Capítulo 7 - Testing y Control de Versión

Si bien el presente proyecto no se centró en aspectos de control de calidad (QC) o aseguramiento de calidad (QA), el capítulo pretende mostrar las prácticas básicas utilizadas en etapas de Testing y Mantenimiento para poder lograr una mejora del producto software desarrollado y así aportar a la calidad del mismo. Asimismo se describe los mecanismos utilizados para el versionado del código fuente durante la etapa de desarrollo.

7.1. Testing

El Testing [13] pertenece a las actividades de verificación y validación, las cuales pretenden asegurar que el software respeta su especificación y satisface las necesidades de los usuarios. El Testing o pruebas de software, es una actividad desarrollada para evaluar la calidad del producto, y para mejorarlo al identificar defectos y problemas.

Consiste en la verificación dinámica del comportamiento de un sistema sobre un conjunto de casos de prueba en relación con el comportamiento esperado. Es una técnica dinámica, el programa se verifica poniéndolo en ejecución de la forma más parecida posible a cómo ejecutará cuando esté en producción, esto se contrapone a las técnicas estáticas las cuales se basan en analizar el código fuente (ver Anexo *Tipos de Testing*).

7.2. Metodología Utilizada

Para el Proyecto se realizaron pruebas de Unidad (Unit Testing) y pruebas Funcionales (Functional Testing). posteriormente integradas en versiones de prueba de la rama Testing del mismo (corridas bajo el servidor de Desarrollo).

Las Pruebas fueron separadas en los siguientes módulos:

- **URLs:** Pruebas sobre las Rutas de acceso al sistema, tanto del Core como de la API.
- **VIEWS:** Pruebas sobre Vistas, cada función y/o procedimiento del Core y API.
- **MODELOS:** Pruebas sobre el ORM del Core y Serializers de la API.
- **Pruebas Funcionales:** Tales como Logueo, impresión de PDFs, Envío de Emails, etc.

Se utilizó *UnitTest* de Python para las pruebas de Unidad y *Selenium* (con el módulo del navegador web Chromium) para las pruebas Funcionales.

Selenium es una API utilizada para crear pruebas funcionales y de aceptación. Abre una ventana del navegador (por ejemplo Google Chrome) y lee su contenido para testear y simular la interacción del Usuario con la aplicación, permite programar funciones de llenado de formularios, ejecutar acciones y leer porciones de la página para luego evaluar los resultados, automatizar pruebas de logueo de usuarios, ejecución de búsquedas y/o de formularios.

El objetivo fue testear cada aspecto y funcionalidad del sistema, para ello se utilizó el módulo *Coverage* de Python el cual indica el porcentaje del sistema “cubierto” por dichas pruebas y las porciones de código que no lo están (siendo una herramienta que agiliza el proceso de testing).

Se han realizado tests funcionales con usuarios reales, permitiendo de esta manera comprobar la usabilidad y obtener un feedback directo de posibles usuarios finales.

7.3. Integración Continua (CI)

La integración continua es una práctica de desarrollo de software donde los miembros de un equipo realizan integraciones con frecuencia, generalmente cada persona integra sus modificaciones al menos una vez al día, pudiendo haber múltiples integraciones en un día. Esto permite detectar errores de integración tan pronto como sea posible. La integración puede incluir la compilación y ejecución de pruebas de todo un proyecto.

En proyectos como este, en donde se adoptan metodologías ágiles, la integración continua es uno de los pilares de la agilidad, asegurando el funcionamiento del sistema en cada integración. El proceso utilizado para la integración continua en nuestro caso fue, cada cierto tiempo (horas) hacer una actualización de los fuentes (update) del control de versiones Git, realizar un merge en caso de haber conflicto con la versión local, compilar y ejecutar pruebas.

De la misma manera cuando se realizan cambios locales se realiza el proceso para subir al servidor de control de versiones.

En nuestro caso utilizamos **Travis CI**, el cual permite conectar un repositorio de Github y probar después de cada push realizado, regenerando automáticamente el proyecto.

El flujo de trabajo implementado es el siguiente:

1. Realizar un Push al repositorio de GitHub;
2. Se dispara un proceso interno en la plataforma;
3. La rama solicitada es actualizada, se crea la app y se corren los tests pertinentes;
4. Los resultados son enviados por correo (según configuración);

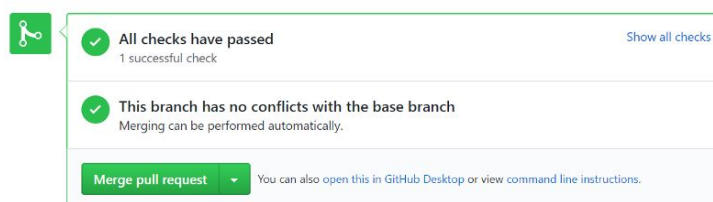


Fig.22 - Ejemplo de Mensaje resultante de Travis CI.

Si todos los tests son satisfactorios significa que el código puede ser integrado sin problemas en la rama principal del proyecto.

7.4. Políticas de Publicación

Una vez que se pasan las pruebas (tanto de código como de integración), se integran los cambios de la rama de Testing a la rama de Máster y se incorpora la nueva funcionalidad al sistema en Producción (comando *merge*).

Cabe destacar que las publicaciones del sitio son realizadas fuera del horario de trabajo para no interferir con la utilización del mismo. Las mismas no se realizan con una frecuencia fija predefinida, más bien depende de los cambios realizados y la necesidad de actualizar el sitio.

7.5. Control de Versiones (GIT)

Para el versionado del código fuente se utilizó Git. El mismo se presenta como un sistema distribuido, en el que todos los nodos manejan la información en su totalidad y por lo tanto pueden actuar de cliente o servidor en cualquier momento, es decir, se elimina el concepto de «centralizado». Esto se logra gracias a que cada vez que se sincronizan los cambios con el repositorio remoto Git, se guarda una copia entera de los datos con toda la estructura y los archivos necesarios. Así ya no es necesario recurrir a Internet para consultar los cambios históricos sobre un archivo o para ver quién fue la última persona que lo editó, todo se hace directamente sobre la copia local y luego, cuando se considere oportuno, se puede enviar esos cambios hacia el repositorio remoto.

Es indispensable utilizar una herramienta de control de versiones en cualquier desarrollo de software, ya sea un proyecto pequeño, mediano o de gran envergadura. Estos sistemas facilitan la administración de las distintas versiones del producto desarrollado, y crean un entorno de trabajo colaborativo, aumentando la productividad del equipo a la hora de integrar versiones y cambios en el código fuente.

Para el desarrollo del sistema se utilizó **GitHub Desktop** como gestor de versionado GIT, ya que agiliza la ejecución de comandos tales como STASH, PULL,

PUSH, COMMIT, etc, así como el manejo de las diferentes ramas/versiones del proyecto.

En la configuración se definen los siguientes archivos:

.gitignore: encargado de detallar todos los archivos y directorios que van a quedar fuera del versionado, esto es, archivos de claves privadas, repositorios privados y lista de requerimientos de programas (requirements.txt).

settings.ini : Definición de todas las claves y variables de seguridad del sistema. Queda excluido del repositorio ONLine (por cuestiones de seguridad) y será consultada por el módulo “**python-decouple**” una vez que arranca el servidor de desarrollo de Django (el mismo se encargará de completar las variables referenciadas en el sistema según su requerimiento).

Para el versionado del código del Proyecto en cuestión se determinaron las siguientes ramas de desarrollo:

- **master**: Es la rama principal, también conocida como “producción”. Allí se encuentran los archivos finales que serán parte de la versión final y utilizable del sistema.
- **prueba**: Es la rama de prueba o desarrollo donde inicialmente se agregan modificaciones, correcciones y nuevas funcionalidades al sistema. Son subidas al servidor de producción pero en una aplicación separada a la rama “master”. Luego de pasar todas las pruebas se incorpora a la rama principal (mediante el comando “*merge*”).

Capítulo 8 - Seguridad

En este capítulo se describen los posibles puntos de ingreso inseguros, las contra-medidas existentes en el Framework Django y medidas de seguridad implementadas en el sistema.

8.1. Seguridad

El Framework nos brinda protección nativa (parte del núcleo de Django)[16] contra ataques del tipo:

- **XSS (Cross Site Scripting):** XSS permite a un usuario inyectar código a ejecutar del lado del cliente en los navegadores de otros usuarios. Esto generalmente se logra guardando código malicioso en la base de datos en donde será devuelto y mostrado a otros usuarios, o haciendo que los usuarios clickeen en un link que provocará la ejecución de cierto código JavaScript del atacante en sus navegadores. Sin embargo los ataques XSS pueden ser originados desde cualquier origen de datos no confiable, tales como cookies o servicios web (generalmente cuando los datos no son validados antes de ser incluidos en la página).

Usando los templates de Django se tiene protección contra la mayoría de estos tipos de ataque. Sin embargo, es importante entender qué protección nos brinda y sus limitaciones. Django se encarga de “limpiar” ciertos caracteres en sus templates que son perjudiciales en particular para el código HTML a ser interpretado.

- **CSRF Protection:** El paquete `django.contrib.csrf` provee protección contra *Cross-site request forgery* (CSRF) (falsificación de peticiones inter-sitio).

CSRF, también conocido como *"session riding"* (montado de sesiones), es un exploit de seguridad en sitios Web. Se presenta cuando un sitio Web malicioso induce a un usuario a cargar sin saberlo una URL desde un sitio al

cual dicho usuario ya se ha autenticado, por lo tanto saca ventaja de su estado autenticado.

- **SQL Injection:** La *inyección de SQL* es un exploit común en el cual un atacante altera los parámetros de la página (tales como datos de GET/POST o URLs) para insertar fragmentos arbitrarios de SQL que una aplicación Web ingenua ejecuta directamente en su base de datos. Es probablemente la más peligrosa y, desafortunadamente, una de las más comunes vulnerabilidades existentes.

Esta vulnerabilidad se presenta más comúnmente cuando se está construyendo SQL "a mano" a partir de datos ingresados por el usuario. Aunque este problema es insidioso y a veces difícil de detectar la solución es simple: *nunca* confíes en datos provistos por el usuario y *siempre escapa* el mismo cuando lo conviertes en SQL. La API de base de datos de Django escapa automáticamente todos los parámetros especiales SQL, de acuerdo a las convenciones de uso de comillas del servidor de base de datos que estés usando (por ejemplo, PostgreSQL o MySQL).

- **Session Forging/Hijacking:** No se trata de un ataque específico, sino una clase general de ataques sobre los datos de sesión de un usuario. Django incluye protección contra un ataque de sesiones de fuerza bruta. Los identificadores de sesión se almacenan como hashes (en vez de números secuenciales) previniendo un ataque por fuerza bruta, y un usuario siempre obtendrá un nuevo identificador de sesión si intenta usar uno no existente, lo cual evita la session fixation.
- **Clickjacking protection:** Clickjacking es un tipo de ataque en donde un sitio malicioso contiene a otro sitio dentro de un frame, con el fin de engañar al usuario y obligarlo a realizar acciones no deseadas sobre el sitio original. Django nos brinda protección contra este tipo de ataques llamado "clickjacking" en la forma de X-Frame-Options middleware, lo cual previene que un sitio pueda ser renderizado en un frame. Es posible deshabilitar esta protección.

Al haber utilizado una API con ingreso por **fuera** del circuito de seguridad del núcleo del Framework (sesiones de usuario) se implementaron técnicas de validación para brindar garantías con respecto a quien está solicitando cada recurso y si tiene permiso de acceso o no al mismo. Utilizando el módulo JWT(Json Web Token) se garantiza que cada acceso haya sido apropiadamente validado. JWT es un estándar abierto basado en JSON para la creación de Tokens de acceso. El procedimiento utilizado por JWT está desarrollado en el apartado “*Extensibilidad del Sistema*”.

Con respecto a la seguridad de las Bases de Datos, al contratar a Webfaction como proveedor del servicio de DBMS, se garantiza una disponibilidad de servicio de casi el 99,99% así como servidores con los últimos parches de seguridad al día de la fecha. Para garantizar la recuperación rápida de la BD, se realiza un backup todos los días a las 2 am, el cual registra cualquier cambio realizado en el día. El backup es creado utilizando el comando “*mysqldump*”, siendo ejecutado el mismo en un script de *crond* (comando de administración y ejecución de tareas diarias en el sistema operativo CentOS), el cual almacena la base de datos en un archivo de tipo script sql y luego lo comprime con el comando *gzip*.

```
0 2 * * * mysqldump -u <user> -p<passwd> <db> | gzip > $HOME/backups/db.sql.gz
```

8.1.1. Seguridad en el Protocolo

HTTPS (HyperText Transfer Protocol Secure) es el protocolo utilizado en la transmisión de los datos, añade una capa de seguridad extra a HTTP al integrar la encriptación de los datos a través de SSL/TLS). Bajo HTTPS toda la información que fluye entre el cliente y el servidor viaja cifrada a través de protocolos SSL/TLS, esto previene que intermediarios accedan a información privada contenida dentro de cada mensaje (por ej. contraseñas, números de tarjetas de crédito, etc). Además, en la mayoría de casos al cifrado lo acompaña un certificado emitido por una autoridad para corroborar que los datos de la web pertenecen a una persona o empresa real.

Según sea el caso dichos certificados pueden ser creados y renovados de manera automática mediante el proveedor del servicio de hosting.

Para los dominios utilizados en el proyecto se utilizaron certificados Lets Encrypt (autoridad dedicada a proveer certificados de manera libre y gratuita) los cuales son renovados automáticamente por Webfaction cada 3 meses.

Capítulo 9 - Extensibilidad

En este Capítulo se describe la extensibilidad del sistema, es decir, funcionalidades, extensiones o proyectos futuros a partir de este proyecto, cuestiones que no formaron parte del alcance inicial.

9.1. Extensibilidad del Sistema

Se desarrolló una API (Application Programming Interface) para poder interactuar con el sistema sin necesidad de cambios en el código, permitiendo a Clientes externos incorporar las nuevas funcionalidades de manera casi inmediata. Dicha interfaz es parte del módulo de Autogestión y fue completamente desarrollada en Django incorporando el módulo Django REST y utilizando tecnología JWT (Json Web Token).

El funcionamiento es el siguiente:



Figura 23 - Diagrama de Interacción de la API.

- El Cliente/App externo solicita un dato a la API, accediendo a una ruta determinada mediante una llamada GET/POST/PUT/PATCH (HTTPS v1.1).

Por ej.: <https://muniNNN.boletaweb.com.ar/api/v1/boletas/NNN/>

(permite solicitar todas aquellas boletas del padrón NNN)

- La API verifica mediante un código único (Token) enviado en la cabecera si el solicitante tiene permitido el acceso al recurso y si todavía es válido (tiene un tiempo de validez establecido), de lo contrario le solicita que se loguee para validar y generar dicho Token nuevamente.
- Una vez validada la identidad del solicitante, se procesa el requerimiento y se retorna el recurso, o en su defecto una respuesta indicando el éxito o fracaso de la operación. Una respuesta válida podría ser un recurso JSON con el listado de boletas (discriminadas como un conjunto de “campo”:”valor”) de un padrón determinado.

Dicho esquema permite controlar el nivel de acceso a los recursos de manera sencilla y transparente, así como también la escalabilidad en la funcionalidad de dicha API.

Como posibles extensiones o funcionalidades a futuro de nuestro sistema utilizando la API pueden plantearse dos grandes caminos:

- **API Consultas/Gestión:** Se puede utilizar la API de Gestión/Consulta para interactuar con otras tecnologías tales como Apps. de Celular, Webapps, etc.

La comunicación se realiza mediante el intercambio de datos estandarizados (JSON) y normas de seguridad establecidas (JW Token). Este camino abre el juego para que el Municipio pueda expandir el abanico de servicios al Contribuyente, estableciendo una mayor presencia e interacción con los mismos.

- **Generación de Tickets de pago ONLINE:** Pueden asociarse según los requerimientos de cada sistema de Pago, es decir, haciendo que la API sea configurable/parametrizable. El Municipio podrá gestionar convenios con las distintas entidades de Pago electrónico existentes.

Con la incorporación de la API se pueden ir agregando al sistema ONLine módulos tales como Reimpresión de Recibos de Sueldo, Sistema de Reclamos, Consulta de Expedientes, Solicitud de Turnos ONLine, etc.

Conclusión

El uso y aplicación de TICs en los procesos de un municipio es un componente fundamental para su ingreso en la Sociedad de la Información. El municipio, como la unidad del Estado con contacto directo con el ciudadano, no sólo afecta de este modo sus propios procesos, sino que también colabora en forma indirecta con la tecnificación de los ciudadanos. La informatización de un municipio requiere de un plan de proyecto detallado que incluya un análisis de la forma de trabajo manual para desarrollar un proceso informático que se adapte a la realidad de uso. Tampoco debe limitarse a informatizar los procesos existentes: con un análisis claro de ventajas y desventajas deben rediseñarse los procesos para ajustarlos a las necesidades actuales, con especial foco en el servicio al ciudadano.

Pero fundamentalmente el éxito de estos proyectos requiere de personal y autoridades educadas en la Sociedad de la Información, de modo que conozcan sus fortalezas y también sus debilidades, y no perciban que la informática amenaza su continuidad laboral. Sin líderes de TICs en los Municipios, la informatización no pasará de ser más que un proceso manual mejorado por las computadoras. Con líderes, usuarios y ciudadanos que posean una visión integral del beneficio de las TICs habremos construido una base muy sólida donde asentar una política de e-gobierno.

Habiendo finalizado el desarrollo y la posterior implementación exitosa en los múltiples municipios llegamos a la conclusión que la adopción de una metodología híbrida (XP y Scrum) y la arquitectura elegida fueron acertadas.

Gracias a las historias de usuarios, las cuales se utilizaron para la recolección de información; y junto con la constante comunicación con los clientes, se generó la retroalimentación adecuada para llevar adelante los Sprint planeados, dando como resultado un sistema confiable y funcional. La constante refactorización del código, el diseño simple, la integración continua y las pruebas constantes permitieron generar un sistema fácil de mantener y en constante actualización.

En la actualidad, el sistema sigue funcionando con las mismas tecnologías y sumando nuevas funcionalidades adaptándose a las necesidades requeridas por los

clientes. El éxito e impacto del sistema le permitió a Grupo Guadalupe SRL encarar nuevos proyectos, brindando a los municipios un abanico de aplicaciones a implementar que van desde la Gestión de Turnos para trámites presenciales, Consultas de Pagos orientado a Proveedores y Gestión de Recibo de Sueldos y Vacaciones para el Personal Municipal.

En el proyecto no tuvimos mayores problemas para llegar a dichos objetivos, esto es, gracias a la buena predisposición del personal municipal y al conocimiento previo de los asesores y encargados de cada área. Inicialmente existieron algunos inconvenientes con el relevamiento de los requerimientos, más que nada por inexperiencia previa, pero luego se fueron optimizando los procesos, todo esto gracias a la retroalimentación obtenida en las evaluaciones de cada etapa del relevamiento.

Podemos afirmar que por más que la metodología elegida sea la adecuada, el éxito del proyecto depende en gran parte del ambiente de trabajo, sus integrantes y la claridad del cliente para definir los objetivos. Son pilares fundamentales para poder llevar adelante un proyecto exitoso.

Por otro lado, como profesionales, el proyecto nos aportó tanto conocimientos y habilidades sobre nuevas tecnologías y herramientas de trabajo como así también la posibilidad de poner en práctica los conocimientos adquiridos durante el cursado de la carrera.

Referencias Bibliográficas

- [1] Canós, Letelier y Penadés: Metodologías Ágiles en el Desarrollo de Software. DSIC -Universidad Politécnica de Valencia (2003).
[En línea]
<http://aleteya.cs.buap.mx/~jlavalle/papers/agileMethodology/TodoAgil.pdf>
- [2] Metodologías ágiles de gestión de proyectos (Scrum, DSDM, Extreme Programming – XP) (2008).
[En línea]
<https://www.marblestation.com/?p=661>
- [3] Pérez, Oiver Andrés: Cuatro enfoques metodológicos para el desarrollo de Software RUP – MSF – XP - SCRUM (2011).
[En línea]
https://apps.utel.edu.mx/recursos/files/r161r/w20493w/Cuatro_enfoques_metodologicos_para_el_desarrollo_d.pdf
- [4] Palacio, Juan: Flexibilidad con Scrum (2008).
[En línea]
<http://www.safecreative.org/work/0710210187520>
- [5] Pressman: Ingeniería de Software: Un enfoque práctico 6ta Ed. Madrid, McGraw-Hill (2005).
- [6] Kendall y Kendall: Análisis y Diseño de Sistemas 8va Ed. México, Pearson Education (2011).
- [7] Bass, Clement y Kazman: Software Architecture in Practice Third Edition. Addison Wesley (1998)
- [8] Patrones de Arquitectura.
[En línea]
https://es.wikipedia.org/wiki/Patrones_de_arquitectura

- [9] Transferencia de Estado Representacional REST.
[En línea]
https://es.wikipedia.org/wiki/Transferencia_de_Estado_Representacional
- [10] Django documentation.
[En línea]
<https://docs.djangoproject.com/en/3.1/>
- [11] Gomez, Diego: Integración Continua (2008).
[En línea]
<https://dosideas.com/noticias/metodologias/309-integracion-continua>
- [12] Gothelf y Seiden: Learn UX: Applying Lean Principles to Improve User Experiences. O'Reilly Media Inc (2013).
- [13] Cristiá, Maximiliano: Introducción al Testing de Software. Ingeniería de Software, Universidad Nacional de Rosario (2015).
[En línea]
<https://www.fceia.unr.edu.ar/ingsoft/testing-intro-a.pdf>
- [14] Modelo–vista–controlador.
[En línea]
<https://es.wikipedia.org/wiki/Modelo–vista–controlador>
- [15] Sarco, Jose Pabo: ISTQB – Cap 2 – Testing a Través del Ciclo de Vida del Software - 2 (2012).
[En línea]
<https://josepablosarco.wordpress.com/2012/05/25/istqb-cap-2-testing-a-traves-del-ciclo-de-vida-del-software-ii/>
- [16] Django Web application Security.
[En línea]
https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/web_application_security

- [17] Refactorización.

[En línea]

<https://es.wikipedia.org/wiki/Refactorización> [En línea]

Anexos

A. Modelo MVT de Django

Django fue diseñado para promover el acoplamiento débil y la estricta separación entre las piezas de una aplicación. Siguiendo dicha filosofía, es fácil hacer cambios en un lugar particular de la aplicación sin afectar otras piezas. En las funciones de vista, por ejemplo, discutimos la importancia de separar la lógica de negocios de la lógica de presentación usando un sistema de plantillas. Con la capa de la base de datos, aplicamos esa misma filosofía para el acceso lógico a los datos.

Estas tres piezas juntas — la lógica de acceso a la base de datos, la lógica de negocios, y la lógica de presentación — comprenden un concepto que a veces es llamado el patrón de arquitectura de software Modelo-Vista-Controlador (MVC) [9].

En este patrón, el "Modelo" hace referencia al acceso a la capa de datos, la "Vista" se refiere a la parte del sistema que selecciona qué mostrar y cómo mostrarlo, y el "Controlador" implica la parte del sistema que decide qué vista usar, dependiendo de la entrada del usuario, accediendo al modelo si es necesario.

Debido a que la "C" es manejada por el mismo framework y la parte más importante se produce en los modelos, las plantillas y las vistas, Django es conocido como un Framework MTV.

En el patrón de diseño MTV,

- **M** significa "Model" (Modelo), la capa de acceso a la base de datos. Esta capa contiene toda la información sobre los datos: cómo acceder a estos, cómo validarlos, cuál es el comportamiento que tiene, y las relaciones entre los datos.
- **T** significa "Template" (Plantilla), la capa de presentación. Esta capa contiene las decisiones relacionadas a la presentación: como algunas cosas son mostradas sobre una página web u otro tipo de documento.

- **V** significa "View" (Vista), la capa de la lógica de negocios. Esta capa contiene la lógica que accede al modelo y la delega a la plantilla apropiada: puedes pensar en esto como un puente entre el modelos y las plantillas.

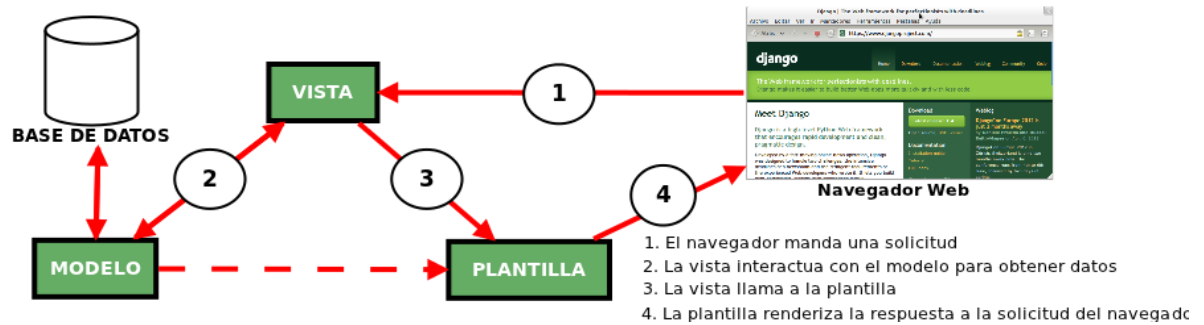


Figura 24 - Modelo MVT de Django.

B. Estructura de una Aplicación DJANGO

La figura siguiente describe los componentes que interactúan en el funcionamiento de una aplicación Django:

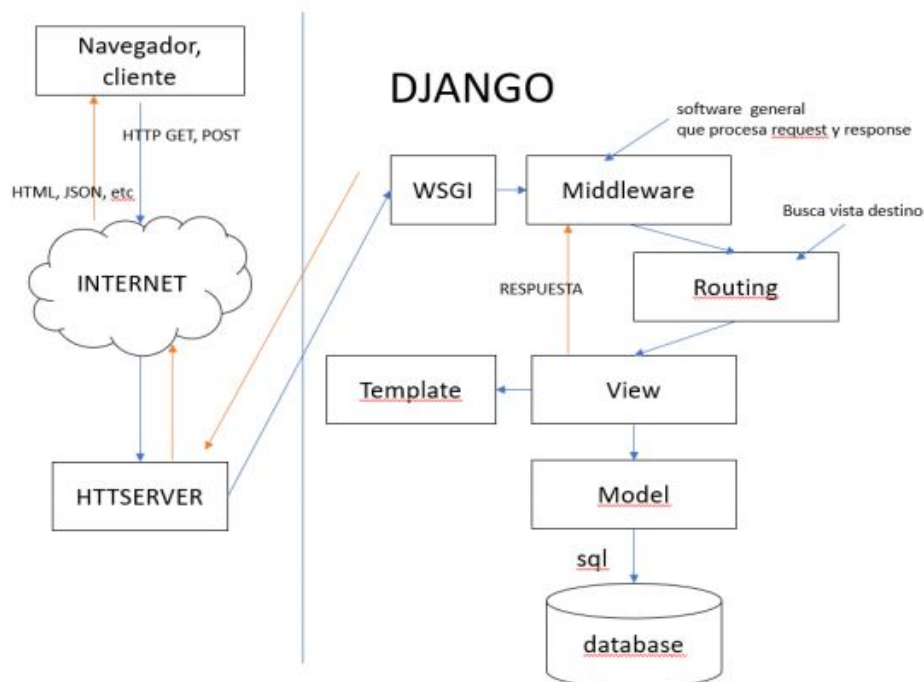


Figura 25 - Interacción de módulos en una aplicación Django.

Navegador/Cliente	Agente que solicita datos al Servidor. En la mayoría de los casos será un navegador, pero puede ser otro agente. Se comunica con el Server mediante el protocolo HTTPs.
Internet	Red de comunicación que permite alcanzar el servidor. Será una red tcp/ip.
HttpServer	Es el server que escucha las peticiones. Entre los más conocidos están Nginx, Apache, IIS.
WSGI	Modelo de interacción entre servidores web y aplicaciones Python.
Middleware	Software python dentro de django, definido en settings y que permite alterar la petición y/o respuesta.
Routing	Módulo de Django que usa las definiciones en el fichero urls.py para seleccionar la vista.
View	Código python que recibe una petición y la procesa, genera la respuesta y la renderiza en los templates.
Model	La vista usa las clases que representan el modelo de datos. Dichas clases del modelo accederán a la base de datos vía SQL.
Template	En conjunción con la vista, describe la salida que será principalmente html. Usará datos proporcionados por la vista que provienen de la base de datos.

C. Tipos de Testing

C.1. Testing Estructural o de Caja Blanca

Las pruebas de caja blanca [10] se centran en los detalles procedimentales del software, por lo que el diseño de los casos de prueba está fuertemente ligado al código fuente. En las pruebas se escogen distintos valores de entrada para examinar cada uno de los posibles flujos de ejecución del programa y asegurarse que se devuelven los valores de salida adecuados. Como los casos de prueba se crean en base a la estructura del código fuente, no es posible generarlos hasta tanto no haya sido terminado el módulo a probar. Y, si la implementación cambia, debido a errores o cambios en las estructuras de datos o algoritmos, puede ser necesario volver a rediseñar los casos de prueba.

C.2. Testing Basado en Modelos o de Caja Negra

Es una técnica de pruebas de software en la cual la funcionalidad se verifica sin tomar en cuenta la estructura interna de código, detalles de implementación o escenarios de ejecución internos en el software. En las pruebas de caja negra [10], nos enfocamos solamente en las entradas y salidas del sistema, sin preocuparnos en tener conocimiento de la estructura interna del programa de software. Para obtener el detalle de cuáles deben ser esas entradas y salidas, nos basamos en los requerimientos de software y especificaciones funcionales.

La distinción entre técnicas de pruebas de caja negra y pruebas de caja blanca es la clasificación clásica de las pruebas de software. Para el proyecto, la mayoría de las pruebas realizadas fueron pruebas de caja negra, que se centran en verificar la funcionalidad del mismo, sin la necesidad de diseñar casos de prueba en base al código, que puede llegar a modificarse.

C.3. Pruebas Funcionales y No Funcionales

Las pruebas funcionales [10] se centran en comprobar que los sistemas desarrollados funcionan acorde a las especificaciones funcionales y requisitos del cliente. Se basan en funciones y su interoperabilidad con sistemas específicos, y puede llevarse a cabo en todos los niveles del Testing. Las pruebas no funcionales incluyen pruebas de rendimiento, carga, estrés, usabilidad, entre otros requerimientos no funcionales definidos para el sistema. Se centran en características que debe cumplir el sistema y puede ejecutarse en todos los niveles de pruebas.

D. Refactoring de código fuente

El refactoring (o refactorización) [11] es una técnica de la ingeniería de software para reestructurar código fuente, alterando su estructura interna, pero sin modificar su comportamiento externo. Esta práctica es muy utilizada en las metodologías ágiles, que tiene como objetivo remover duplicación de código, mejorar su legibilidad, simplificarlo y hacerlo más flexible para facilitar sus posteriores cambios y mantenimiento.

El refactoring es la parte del mantenimiento del código que no arregla errores ni añade funcionalidad. En este proyecto se dio mucha importancia a esta técnica para mejorar la estructura del código y facilitar su mantenimiento. A medida que se van desarrollando nuevas funcionalidades y realizando modificaciones, se realiza el refactoring.

Glosario de Definiciones

Apache: es un servidor web HTTP de código abierto, para plataformas Unix (BSD, GNU/Linux, etc.), Microsoft Windows, Macintosh y otras, que implementa el protocolo HTTP/1.1 y la noción de sitio virtual.

API: es un conjunto de subrutinas, funciones y procedimientos (o métodos, en la programación orientada a objetos). Representa la capacidad de comunicación entre componentes de software.

Cookie: hace referencia a una pequeña información enviada por un sitio web y almacenada en el navegador del usuario.

Coverage: es una medida que indica cuánto código es capaz de cubrir una prueba. El objetivo sería generar pruebas que cubran todas las llamadas de función en el código,

CSS (Cascading Style Sheets): es un lenguaje de diseño gráfico para definir y crear la presentación de un documento estructurado escrito en un lenguaje de marcado.² Es muy usado para establecer el diseño visual de los documentos web, e interfaces de usuario escritas en HTML o XHTML

DDL (Data Definition Language): es un lenguaje de descripción de datos que describe los registros, los campos, y "conjuntos" que conforman el usuario modelo de datos.

Exploit: es un fragmento de software, fragmento de datos o secuencia de comandos o acciones, utilizada con el fin de aprovechar una vulnerabilidad de seguridad de un sistema de información para conseguir un comportamiento no deseado del mismo.

Firebird: es un sistema de administración de base de datos relacional (o RDBMS) (Lenguaje consultas: SQL) de código abierto.

Framework: es una estructura conceptual y tecnológica de asistencia definida, normalmente, con artefactos o módulos concretos de software, que puede servir de base para la organización y desarrollo de software. Típicamente, puede incluir soporte de programas, bibliotecas, y un lenguaje interpretado, entre otras

herramientas, para así ayudar a desarrollar y unir los diferentes componentes de un proyecto.

HTTP: es el protocolo de comunicación que permite las transferencias de información a través de archivos (XHTML, HTML . . .) en la World Wide Web

HTTPS (HyperText Transfer Protocol Secure): es un protocolo de aplicación basado en el protocolo HTTP, destinado a la transferencia segura de datos de hipertexto, es decir, es la versión segura de HTTP.

IaaS (Infrastructure as a Service): se refiere a los servicios en línea que proporcionan un alto-nivel de APIs utilizadas para referenciar indirectamente detalles a bajo nivel de infraestructura como recursos de informática física, ubicación, dato partitioning, scaling, seguridad, copia de seguridad etc.

JSON (JavaScript Object Notation): es un formato de texto sencillo para el intercambio de datos.

Merge: Comando de GIT que Impacta los cambios en la rama en la que te encuentras parado.

MySQL: es un sistema de gestión de bases de datos relacional, considerada como la base de datos de código abierto más popular del mundo.

Pecuniarias: Se utiliza para denominar al deber que tiene una parte de llevar a cabo la entrega de un dinero en efectivo a otra, en señal de prestación.

REST (representational state transfer): es un estilo de arquitectura software para sistemas hipermedia distribuidos como la World Wide Web. Es utilizado en el sentido más amplio para describir cualquier interfaz entre sistemas que utilice directamente HTTP para obtener datos o indicar la ejecución de operaciones sobre los datos, en cualquier formato (XML, JSON, etc).

Scrum: es un marco de trabajo para desarrollo ágil de software en el que se aplican de manera regular un conjunto de buenas prácticas para trabajar colaborativamente en equipo.

Selenium: es un entorno de pruebas de software para aplicaciones basadas en la web, provee una herramienta de grabar/reproducir para crear pruebas sin usar un lenguaje de scripting para pruebas.

Serializers: métodos de codificación de un objeto en un medio de almacenamiento (como puede ser un archivo, o un buffer de memoria) con el fin de transmitirlo a través de una conexión en red como una serie de bytes o en un formato humanamente más legible como XML o JSON, entre otros.

SOAP (Simple Object Access Protocol): es un protocolo estándar que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML.

Sprint: es el período de tiempo en el cual se lleva a cabo el trabajo en sí.

SQL (Structured Query Language): es un lenguaje de dominio específico utilizado en programación, diseñado para administrar, y recuperar información de sistemas de gestión de bases de datos relacionales.

SSL/TLS: es un protocolo de la capa de seguridad de transporte Transport Layer Security (TLS) provee la capacidad de asegurar la comunicación a través de las redes.

TaDeSe: Sistema de Gestion Integrado de Derechos y Servicios, brindado por la empresa Grupo Guadalupe SRL.

Token: es una referencia (es decir, un identificador) que regresa a los datos sensibles a través de un sistema de tokenización (proceso de sustitución de un elemento de datos sensible por un equivalente no sensible).

WSGI (Web Server Gateway Interface): Es una librería escrita en el lenguaje Python para redireccionar peticiones desde el Servidor web hacia los servidores de aplicaciones.

XML (eXtensible Markup Language): es un metalenguaje que permite definir lenguajes de marcas desarrollado por el World Wide Web Consortium (W3C) utilizado para almacenar datos en forma legible.

XP (eXtreme Programming): es una metodología de desarrollo de la ingeniería de software en la que se pone más énfasis en la adaptabilidad que en la previsibilidad.