# Proyecto final de carrera

## "Reimplementación y extensión de una plataforma de denuncia y asesoramiento de delitos informáticos para Latinoamérica"

Armando Andini

Director de Proyecto: Ing. Martín Domínguez

Ingeniería en Sistemas de Información

Universidad Tecnológica Nacional

Facultad Regional Santa Fe

2019

# Indice

# Índice de figuras

## Índice de tablas

# 1. Introducción

Este capítulo tiene por objetivo presentar el proyecto final de carrera, denominado "Reimplementación y extensión de una plataforma de denuncia y asesoramiento de delitos informáticos para Latinoamérica", realizado para la empresa AsegurarTe y Segu-Info. Esta presentación comienza mediante el análisis del problema, los fundamentos de la propuesta, los objetivos perseguidos a lo largo del proyecto, el alcance del mismo y la enumeración de los contenidos del resto del informe.

## 1.1. Delimitación del tema

El proyecto consiste en la reimplementación y adición de funcionalidades a un sistema ya existente, llamado ODILA (www.odila.org), el cual sirve como plataforma de denuncia y asesoramiento sobre delitos informáticos. El mismo es un proyecto sin fines de lucro que busca además de brindar soporte y orientación a las víctimas, elaborar estadísticas y análisis sobre las ocurrencias de estos delitos.

Los usuarios, que son víctimas de delitos informáticos pueden realizar una denuncia a través de una serie de formularios presentados en el sitio web, y luego recibir un informe sobre las legislaciones vigentes en el país de residencia. Además son motivados a realizar las denuncias correspondientes en una serie de centros de denuncia proporcionados por el sitio.

La información recopilada es utilizada luego para elaborar un reporte anual de estadísticas sobre delitos informáticos, cuyas cifras son de difícil obtención y de las cuales no se cuenta con fuentes confiables y precisas.

De las funcionalidades existentes solo se reutilizaron aspectos visuales y de diseño , y las nuevas funcionalidades requeridas se implementaron desde cero. Este proyecto es

impulsado por Segu-Info y AsegurarTe, para recibir reportes de delitos informáticos de todo Latinoamérica. Estos delitos incluyen phishing, cracking, grooming, suplantación de identidad, entre otros.

## 1.2. Objetivo general

Reimplementar la plataforma de denuncia de delitos informáticos de ODILA, utilizando tecnologías modernas de desarrollo Web y almacenamiento en la nube.

Analizar, diseñar y desarrollar nuevas funcionalidades que permitan a los usuarios del sitio denunciar siniestros de índole informático, y recibir asesoramiento automatizado en base al delito sufrido y el país de residencia.

Analizar, diseñar e implementar una plataforma de gestión de información y conocimiento que permita a los administradores del sitio gestionar las legislaciones vigentes referidas a delitos informáticos, y generar reportes automatizados sobre los delitos denunciados.

## 1.3. Objetivos Específicos

- Reimplementar la plataforma existente: Se reimplementará la funcionalidad de denuncia existente, que consta de un formulario de denuncia, cuyas preguntas son configuradas por los administradores del sitio.
- Diseñar e implementar una interfaz de usuario amigable: Los usuarios denunciantes encuentran el formulario de denuncia muy intimidante, por lo que se desarrollará un comportamiento dinámico para el mismo y se incluirán animaciones.
- Desarrollar e implementar un panel de administración integral protegido por un mecanismo de autenticación: Permitirá a los administradores gestionar toda la

información del sitio, incluyendo las legislaciones vigentes para los delitos informáticos tipificados en cada país.

- Diseñar e implementar un mecanismo de respuesta de denuncia: Los usuarios denunciantes recibirán un reporte de asesoramiento en el cual se incluyen las leyes y artículos que los protegen frente a los siniestros especificados, y una lista de centros de denuncia a los que pueden acudir para recibir ayuda.
- Modelar e implementar una herramienta para la creación dinámica de cuestionarios, con múltiples preguntas y distintos tipos de respuestas asociadas (texto, opción simple, opciones múltiples, etc).

## 1.4. Alcance

A partir de las diversas reuniones con los clientes se reconocieron cuáles eran las principales características necesarias para afrontar la problemática existente, y de esta manera se logró tener un mejor conocimiento del alcance de este proyecto.

De forma que, el sistema debe contar con las siguientes funcionalidades (los términos específicos serán explicados en el Glosario, sección 6):

- **Realizar denuncia:** Los usuarios que ingresan al sitio pueden realizar denuncias anónimamente, completando un cuestionario de múltiples pasos, cuyas preguntas son definidas por los administradores del sitio.
- **Generar una respuesta al usuario:** Una vez completado el cuestionario, los usuarios deben recibir una respuesta del sistema, la cual debe presentar las legislaciones vigentes en el país de residencia, sobre los delitos denunciados. Además deben listarse los lugares a los que la víctima puede acudir para realizar una denuncia formal y obtener mayor asesoramiento.
- **Presentar datos de la empresa y de difusión del proyecto:** la aplicación web debe contar con secciones que incluyan imágenes e información acerca de los clientes y el proyecto ODILA.

- **Formulario de contacto:** Permitir a los usuarios contactarse directamente con los administradores a través de un formulario de contacto, en el que pueden ingresar su e-mail y la consulta pertinente.
- **Panel de administración:** Los administradores deben poder acceder a una sección para realizar alta, baja y modificación de todas las entidades del sistema. Además en esta sección deben listarse las denuncias recibidas, y los mensajes de contacto enviados por los usuarios.
- **Generación dinámica del formulario de denuncia:** A través de una herramienta disponible para los administradores, deben poder extender y modificar el formulario de denuncia dinámicamente, agregando o quitando preguntas, opciones, etcétera.
- **Importar datos de archivos CSV:** Los clientes desean poder suministrar datos en masa a través de planillas de cálculo exportadas en formato CSV.

## 1.5. Metodología

Para desarrollar la solución prevista se llevarán a cabo las siguientes etapas.

### 1.5.1 Etapa 1 - Análisis de requerimientos generales:

En primera instancia se realizará un relevamiento de los requerimientos generales y se analizará el sistema con el que los clientes contaban previamente. Es necesario determinar los lenguajes de programación, las librerías y herramientas que se utilizaron. Además se identificarán las carencias del mismo y se determinarán cuáles son las funcionalidades principales que se desean agregar.

Estas actividades se llevarán a cabo mediante:

- Reuniones presenciales con los clientes.
- Reuniones remotas a través de videollamadas.
- Intercambios de correo electrónico.

### 1.5.2 Etapa 2 - Refinamiento de los requerimientos:

En esta etapa se refinan los requerimientos generales, lo que los clientes esperan del nuevo sistema.

Además, se confeccionarán las historias de usuario para determinar los requerimientos a realizar, para luego asignarles prioridades.

### 1.5.3 Etapa 3 - Investigación preliminar:

En primera instancia se recolectará material bibliográfico relacionado con las tecnologías que brindan los servicios necesarios para desarrollar una solución que alcancen los objetivos planteados. Entre las tecnologías necesarias se incluyen HTML, CSS y Javascript, presentes en la mayoría de las aplicaciones web, y el framework Ruby on Rails para implementar la lógica del lado servidor. Se realizará también una evaluación y selección de bases de datos relacionales y de proveedores de servicio *PaaS* para alojar una aplicación en la nube.

### 1.5.4 Etapa 4 - Diseño de la arquitectura del sistema:

La arquitectura que se propone para implementar la nueva solución es la de Modelo - Vista - Controlador, siendo la más utilizada para el desarrollo de aplicaciones web, y en la que están basado la mayoría de los frameworks, incluyendo el que se va a utilizar en este proyecto. Esta arquitectura se destaca por las importantes cualidades de ingeniería de software que propone: mantenibilidad, modularización, bajo nivel de acoplamiento y alta reusabilidad de componentes.

La utilización de un framework integral para aplicaciones web permite desarrollar de manera modular, pero a la vez monolítica, tanto el programa servidor como el cliente. Esto se debe a que las tanto las páginas dinámicas como los estilos y los scripts de navegador son

servidos por el mismo sistema que corre la lógica de negocio y se conecta con los demás componentes como ser la base de datos.

### 1.5.5 Etapa 5 – Desarrollo del Sistema:

La metodología a utilizar para la planificación y ejecución del proyecto será un método propio basado en Kanban. El mismo consiste en un enfoque de cambio incremental y evolutivo. Los requerimientos se representan en tarjetas que se ubican en cuatro posibles estados:

**Backlog**: Tareas ya descriptas y aprobadas por los clientes, listas para ser realizadas.

**En progreso**: Las tareas están en etapa de ejecución.

**En etapa de prueba**: Los cambios fueron aplicados al entorno de pruebas. Los clientes pueden probar las nuevas modificaciones libremente y proveer realimentación, en el caso de ser necesario.

**En producción**: Los cambios involucrados en la tarea fueron validados y forman parte de la versión estable del sistema.

Este enfoque permite gran flexibilidad e interacción por parte de los clientes, ya que los mismos pueden crear nuevas tarjetas, ordenarlas en base a su prioridad, y estar al tanto del estado de cada una.

### 1.5.6 Etapa 6 - Pruebas de usuario:

Una vez terminado el sistema y validado, se comienza con un periodo de prueba del sistema a nivel de usuario, para evaluar el nivel de calidad alcanzado en función de los objetivos planteados. Es probable que en esta etapa surjan requerimientos de ajuste de interfaz de usuario, tanto de estilos como de comportamiento, con el fin de mejorar la experiencia final. Se solicitará a colaboradores voluntarios que realicen múltiples denuncias en el sistema de pruebas para asegurarnos que las combinaciones de preguntas y respuestas funcionen correctamente.

### 1.5.7 Etapa 7 – Comienzo de etapa productiva:

Una vez que se hayan migrado los datos al nuevo sistema, y el cliente considere que está en condiciones de empezar a ser utilizado, se configurará el dominio utilizado actualmente para que comience a direccionar a nuestra nueva aplicación en lugar del sistema antiguo.

# 2. Presentación de la empresa

## 2.1. General

AsegurarTe es una una empresa surgida en el año 2008, dedicada al campo de la Seguridad de la Información, concebida con el objetivo de prestar una gama de servicios que tienen como finalidad garantizar la confidencialidad, integridad y disponibilidad de la información en forma constante, tomando como punto de partida las necesidades de cada cliente.

Inmersos en una realidad donde la masividad de las telecomunicaciones y conexiones a la red traen como consecuencia un aumento de casos de conflicto e incidentes de seguridad, la empresa opta por una alternativa de servicios que le permite ayudar al cliente ante los incidentes y delitos informáticos en donde se vean involucrados y afectados distintos tipos de derechos (propiedad, imagen, reputación, confidencialidad, etc).

AsegurarTe cuenta con el respaldo y la experiencia de haber trabajado en una amplia gama de incidentes de seguridad de la información, por lo que está capacitada para asesorar a las víctimas en cada caso concreto. Este nuevo concepto de investigaciones digitales, son brindados siempre comprometidos con la seriedad, confidencialidad y responsabilidad hacia sus clientes.

### 2.1.2. Integrantes del equipo

El proyecto ODILA es impulsado por miembros de la empresa AsegurarTe y Segu-Info. Los tres miembros principales son Agustín Borghello, licenciado en sistemas de información y director de Segu-Info, Marcelo Temperini, socio de AsegurarTe y doctorando en materia de delitos informáticos, y Maximiliano Macedo, analista en informática aplicada y socio fundador de AsegurarTe.

### 2.1.3. Otros proyectos relacionados

#### Botón de pánico

Desde AsegurarTe se realizó una colaboración con la organización “NiUnaMenos”, que constó en la realización de una tecnología de botón de pánico para las mujeres que sufren abuso doméstico y acoso.

Este botón de pánico está pensado para que funcione en un modo familiar, donde la persona tiene la posibilidad de enviar un alerta a contactos de confianza predefinidos (familiares, vecinos, amigos, etc) que puedan asistirla ante casos de emergencia. Dicha alerta se envía vía SMS, indicando la situación de emergencia y la ubicación donde se encuentra la persona.

Entre las funcionalidades se destaca la posibilidad de accionamiento a través de un botón físico. De esta forma, ante una situación de emergencia, la víctima podrá enviar el alerta de pánico a sus contactos de confianza, solamente con presionar reiteradamente el botón de bloqueo hasta que el celular empieza a vibrar indicando que su alerta se está enviando.

Adicionalmente la aplicación posee un novedoso sistema llamado "Modo Protesta", que está diseñado para utilizarse en manifestaciones masivas en señal de protesta. El modo hace parpadear la pantalla, vibrar y emitir un sonido.

#### Proyecto de ley en contra del cibercrimen

En vistas del incremento exponencial de los casos de Suplantación de Identidad Digital, Segu-Info y AsegurarTe se han unido para generar la Primera Cruzada para tipificar el Delito de Suplantación de Identidad Digital en 2012.

La misma está basada en un Proyecto de Ley presentado ante el Senado de la Nación, en el cuál participaron en su redacción el Abog. Marcelo Temperini y el Lic. Cristian

Borghello, junto al equipo de asesores de los senadores, buscando plasmar en el proyecto una alternativa válida que permita avanzar en la lucha contra el cibercrimen.

Los Proyectos presentados son los Nº S-2257/11 - Captación ilegítima de datos confidenciales/Phishing - y Nº S-1312/12 - Suplantación de Identidad Digital - , ya presentados por mesa de entradas del Senado de la Nación e impulsados por la Senadora Nacional María de los Ángeles Higonet y el Senador Nacional Carlos Verna, habían sido girados a la Comisión de Justicia y Asuntos Penales para ser tratados.

## 2.2. Procesos del sistema ODILA

En esta sección se detallarán los principales procesos llevados a cabo por los usuarios a través del sistema ODILA existente. Más adelante se explicarán las falencias del mismo y las necesidades planteadas por los clientes.

### 2.2.1. Proceso de denuncia

El proceso comienza cuando un usuario cualquiera, no autenticado, ingresa al sitio de ODILA y accede a la pantalla principal. En la misma, el usuario se encuentra con información general sobre el proyecto y sus integrantes. Si el mismo se desplaza verticalmente hacia abajo, o clickea en el enlace “Denunciar”, encuentra directamente un formulario de denuncia que consta de múltiples preguntas y distintos tipos de respuestas posibles. Algunas preguntas aceptan respuestas de tipo selección simple, otras selección múltiple, y otras texto plano.

El usuario se encuentra con todas estas preguntas en un único formulario presentado de forma vertical, con alrededor de 10 preguntas relevantes al incidente sufrido. Al final del mismo, el usuario puede ingresar opcionalmente su correo electrónico, y puede enviar el formulario al sistema de ODILA.

Una vez recibida la denuncia, el sistema presenta al usuario una pantalla con un mensaje de éxito, informando que la misma está siendo procesada y que al finalizar recibirá en su correo electrónico (en caso de haberlo ingresado), la lista de leyes que lo cubren en su país respecto al incidente sufrido, junto con una serie de centros de denuncias a los que puede acudir para formalizar la misma. Adicionalmente se agregan recomendaciones generales a tener en cuenta para evitar ser víctima de futuros incidentes informáticos.

En la base de datos del sistema, la denuncia es almacenada para ser luego procesada y transformada en un reporte que los administradores del sitio realizan de forma anual.

### 2.2.2 Carga de datos sobre delitos, legislaciones y centros de denuncia

Los administradores del sistema periódicamente recopilan información sobre nuevos delitos informáticos existentes, así como también la tipificación de los mismos en los distintos países de Latinoamérica. Esto sucede por ejemplo cuando se sanciona una nueva ley que cubre a los ciudadanos ante la ocurrencia de uno de estos delitos, o cuando se relevan nuevas instituciones donde es posible realizar una denuncia de los mismos.

Para ingresar esta información, actualmente los clientes proceden a cargarla en una planilla de cálculo excel, y luego proveen la misma a los desarrolladores. Estos luego proceden a ingresarla en la base de datos del sistema, a través de algún procedimiento manual.

### 2.2.2. Proceso de generación de reporte

Este proceso de negocio es el de mayor valor para los usuarios administradores del sistema, ya que consiste en la agregación y procesamiento de todas las denuncias recibidas en un período de tiempo (habitualmente un año), para obtener estadísticas sobre los delitos informáticos y combatir el problema de la “cifra negra” de estos delitos.

Actualmente este proceso se realiza en gran parte de forma manual, ya que consiste en la exportación de datos desde una base de datos SQL sobre la que opera el sistema, a una planilla excel para que luego los administradores realicen las estadísticas relacionadas utilizando las herramientas de las planillas de cálculo.

Una vez obtenidas las estadísticas para el período evaluado, se redacta un informe analizando los datos obtenidos junto con las observaciones y conclusiones derivadas del mismo. Este reporte es confeccionado por los administradores del sitio, y se envía a los desarrolladores para que lo agreguen a la sección de "Reportes" del sistema, que es de acceso público para cualquier visitante del sitio.

## 2.3. Inconvenientes y dificultades

Entre los problemas que se han encontrado con la forma actual de llevar a cabo las actividades, podemos destacar lo siguiente:

- El formulario de denuncia es muy extenso, por lo que las personas que ingresan al sitio web pueden verse abrumados por el mismo, disminuyendo su motivación a realizar la denuncia del incidente sufrido, y de recomendar la plataforma a sus contactos. Este problema se hace más grave a medida que se agregan nuevas preguntas al formulario, se agregan más delitos informáticos y se da soporte a más países. Además el estilo del mismo es muy anticuado, dado que se asemeja a los sitios web de hace una década. El mismo presenta una oportunidad de modernización a través del agregado de estilos vigentes en sitios web modernos y de comportamiento dinámico.
- La respuesta del sistema ante una denuncia se recibe solo via e-mail, por lo que el usuario luego de completar el formulario debe acceder a su cliente de correo electrónico para poder visualizar el resultado. Sería una mejor experiencia de usuario si además de enviarse este correo, se presentará el resultado de la denuncia con el mismo contenido en el sitio web.

- Los administradores no tienen acceso a la información de las denuncias realizadas y las estadísticas relacionadas en tiempo real. Cuando desean tener conocimiento de las mismas, deben comunicarse con el equipo de desarrollo del sistema para que le provean esta información, generalmente volcando las tablas de la base de datos en una planilla de cálculo para su posterior procesamiento. Sería muy útil tener una sección para administradores en la que se refleje en tiempo real información relevante como ser la cantidad de denuncias, estadísticas por país, delito, rango de edad, etcétera.
- El proceso de carga de datos tiene una amplia carga manual. Implica que los administradores manejen la información en planillas de cálculo y luego las suministren a los desarrolladores del sistema. Esto presenta varios problemas, como ser que estas planillas se puedan perder, se puedan confundir versiones de las mismas, y hasta se puedan corromper por error humano. Además representa un trabajo repetitivo e innecesario, que puede ser ahorrado si se contara con un panel de administración apropiado, en el que se pueda realizar alta, baja, modificación y listado de todas las entidades del sistema relevantes.

## 2.4. Arquitectura actual

Actualmente la solución con la que cuentan los clientes consiste en tres partes: Un servidor web, un servidor de bases de datos, y el navegador web que cumple el rol de cliente.

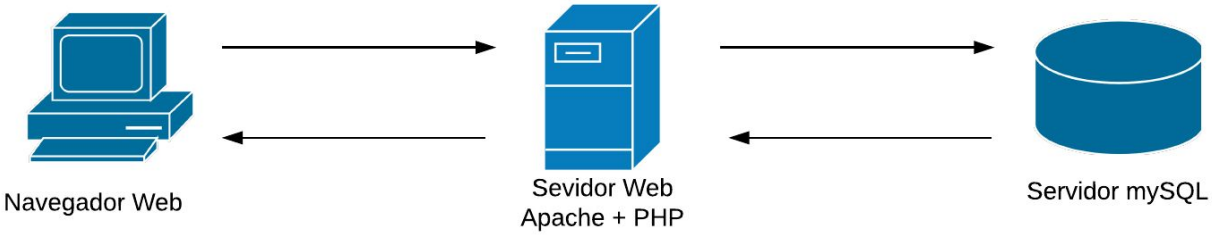



Figura 2.4. Arquitectura actual

### 2.4.1. Servidor de bases de datos

Se cuenta con un gestor de base de datos MySQL 5. MySQL es un sistema de gestión de bases de datos relacional desarrollado bajo licencias de código abierto y está considerada como una de las base datos más populares del mundo.

MySQL es muy utilizado en aplicaciones web, como Joomla, Wordpress, Drupal o phpBB y por herramientas de seguimiento de errores como Bugzilla. Su popularidad como aplicación web está muy ligada a PHP, que a menudo aparece en combinación con MySQL.

### 2.4.2. Servidor Web

Este servidor cuenta con el sistema operativo Ubuntu Server y corre los servicios Apache y PHP.

Apache es un servidor web HTTP de código abierto, para plataformas Unix, Microsoft Windows, Macintosh y otras, que implementa el protocolo HTTP. Apache es usado principalmente para enviar páginas web estáticas y dinámicas en la World Wide Web.

PHP es un lenguaje de programación de propósito general de código del lado del servidor originalmente diseñado para el desarrollo web de contenido dinámico.

### 2.4.3. Cliente Web

Un navegador web es un software, aplicación o programa que permite el acceso a la Web, interpretando la información de distintos tipos de archivos y sitios web para que estos puedan ser visualizados.

La funcionalidad básica de un navegador web es permitir la visualización de documentos de texto, posiblemente con recursos multimedia incrustados. Además, permite visitar páginas web y hacer actividades en ella, es decir, enlazar un sitio con otro, imprimir, enviar y recibir correo, entre otras funcionalidades más.

Los principales navegadores web hoy en día son Google Chrome y Firefox (multiplataforma), en Windows Microsoft Edge, y en iOS Safari.

# 3. Solución

Ante los problemas detectados, detallados en el punto 2.2.2 "Inconvenientes, dificultades", se propuso la implementación de un nuevo Sistema de Información basado en tecnologías modernas que mejore las funcionalidades previamente existentes, y además incluya otras nuevas. Se propuso una solución integral que cubra los requerimientos y proporcione las herramientas necesarias tanto a los administradores del sitio como a los usuarios visitantes del mismo.

Las funcionalidades que proporciona el nuevo sistema son:

- Denunciar un delito informático a través de un formulario dinámico diseñado por los administradores de sitio. El mismo presenta una pregunta a la vez y presenta animaciones para hacer la experiencia de usuario más agradable.
- Respuesta automatizada ante una denuncia, en formato web y vía e-mail, con asesoramiento e información sobre legislaciones vigentes y centros de denuncia.
- Glosario de términos de seguridad informática y recomendaciones generales.
- Formulario de contacto con los administradores del sitio y fundadores del proyecto ODILA.
- Acceso a informes de estadísticas de delitos informáticos de años anteriores
- Panel de administración con acceso mediante usuario y contraseña.
- Panel con información sobre denuncias recientes y cantidad de denuncias recibidas por país y tipo de delito .
- Funcionalidades de alta, baja, modificación y búsqueda por filtrado de entidades del sistema: países, centros de denuncia, delitos informáticos, denuncias realizadas, y legislaciones.
- Diseño del formulario de denuncia dinámico a través de una herramienta de definición de preguntas y respuestas asociadas. Soporta preguntas con respuestas de tipo opción simple, opción múltiple, fecha y texto plano.

- Carga por lotes de legislaciones y centros de denuncia a través de archivos en formato CSV.

## 3.1. Arquitectura

### 3.1.1. Infraestructura

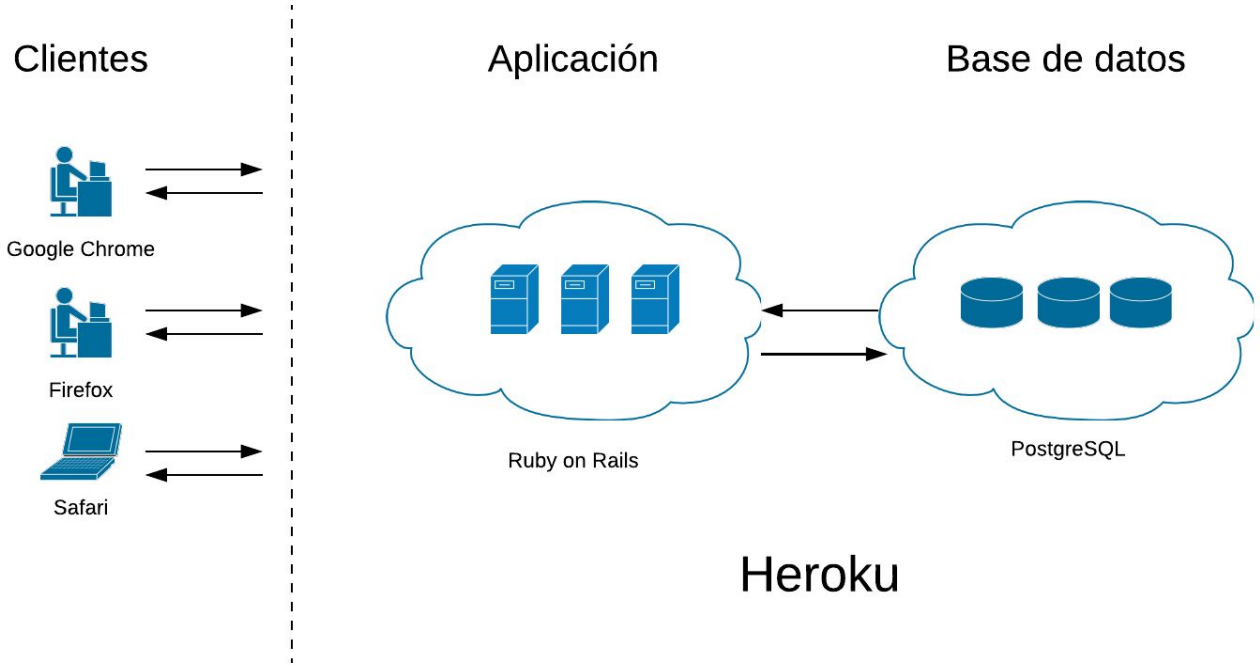



Figura 3.1.1. Arquitectura propuesta

La solución propuesta incluye la utilización de tecnologías modernas para el desarrollo web, y las mismas se integran particularmente bien en entornos que corren en la nube. Por eso, la arquitectura propuesta es la de una aplicación que corra en una plataforma de Cloud Computing de estilo PaaS (Platform as a Service). Esto permite gran flexibilidad ya que no es necesario configurar servidores ni comprar hardware, solo hay que diseñar una

aplicación que cumpla ciertos estándares cloud (12 Factors) y luego se podrá escalar horizontalmente a través de la provisión de nuevas instancias en dicha plataforma.

El servicio de plataforma como servicio elegido es Heroku, dado que ya se cuenta con experiencia previa con el mismo y demostró ser eficaz y robusto. En el mismo se correrá la aplicación del lado servidor, y a la vez se provisionará con una instancia de base de datos PostgreSQL. Esta instancia también es provista por Heroku en forma de *addon* o complemento. Lo que permite esto es formar un ecosistema *cloud*, en el que la aplicación y la base de datos corren en instancias separadas, pero interconectadas por URIs, lo que permite que se puedan escalar independientemente una de otra, dependiendo de la demanda y el recurso que genere cuello de botella. Internamente ambas instancias corren dentro de la infraestructura de Heroku, por lo que los tiempos de conexión y consulta entre ambas partes es mínimo.

La parte de la aplicación cliente será servido por la misma aplicación en forma de páginas web dinámicas. Esto permite desarrollar una solución monolítica en la que la interfaz de usuario y el código de aplicación comparten la misma base de código, por lo que los tiempos de desarrollo se ven minimizados. Los archivos de la parte cliente o Frontend incluyen archivos html, css y javascript, y son servidos por la instancia de aplicación al navegador web cliente.

Esta arquitectura de servicios permite abstraerse totalmente del hardware y sistemas operativos que corren cada parte del sistema, tanto clientes como servidores. Por ejemplo, los distintos usuarios administradores y denunciantes pueden utilizar navegadores y sistemas operativos distintos, pero para el diseño de la aplicación esto es transparente.

### 3.1.2. Arquitectura del software

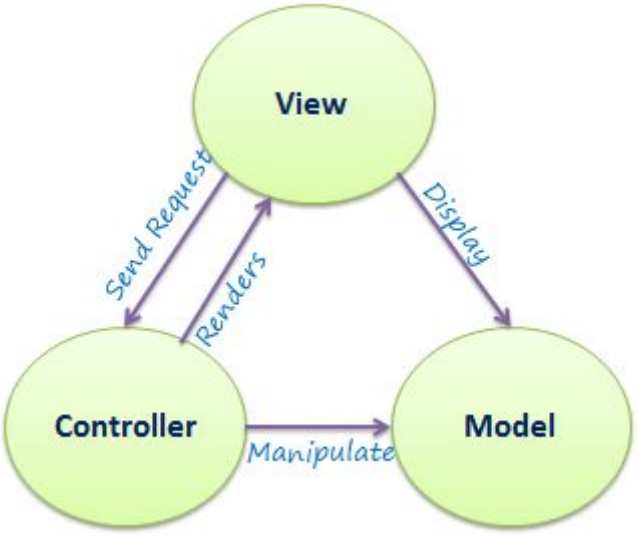



Figura 3.1.2. Arquitectura MVC

La arquitectura de software utilizada por el framework en el cual se basó el desarrollo de este proyecto es la de modelo-vista-controlador (MVC) que desde hace tiempo ya es estándar para aplicaciones web. Esta arquitectura permite una clara separación de conceptos y responsabilidades en el diseño y codificación del software.

**Capa de vistas**

Esta capa es la responsable de representar los datos y dar acceso a las funcionalidades a través de una interfaz gráfica. Incluye el diseño de las páginas del sitio, las hojas de estilo y el comportamiento dinámico de la aplicación cliente. Esta capa no debe incluir lógica de negocio ni de acceso a datos, pero puede incluir lógica de representación de datos (transformaciones, traducciones, representaciones, etc).

**Capa de controladores**

Los componentes en esta capa se encargan de validar las solicitudes realizadas por los clientes web, a través de sus interfaces gráficas. Las responsabilidades principales son las de autenticar usuarios, validar los permisos para la acción solicitada, validar los parámetros recibidos e invocar a la capa de modelo para obtener datos o modificar los mismos. No debe incluirse lógica de negocio en esta capa pero pueden realizarse ciertas validaciones, reglas de filtrado y acceso a servicios externos.

**Capa de modelos**

Es el núcleo de la lógica de negocio y las entidades del sistema. Aquí se representan las entidades en forma de clases con sus atributos y métodos. Además se incluye en esta capa el acceso a los datos persistentes (en este caso una base de datos SQL) y la exposición de una interfaz programática para el acceso y manipulación de los mismos, sin necesidad de escribir consultas en lenguaje SQL u otros por el estilo.

Los componentes de esta capa además se encargan de realizar validaciones a nivel de aplicación de las entidades representadas, por lo que muchas restricciones pueden ubicarse en este nivel y no en las tablas relacionales, lo que permite cambiar el medio de persistencia de datos sin mayores dificultades.

### 3.1.3. Modelo de datos

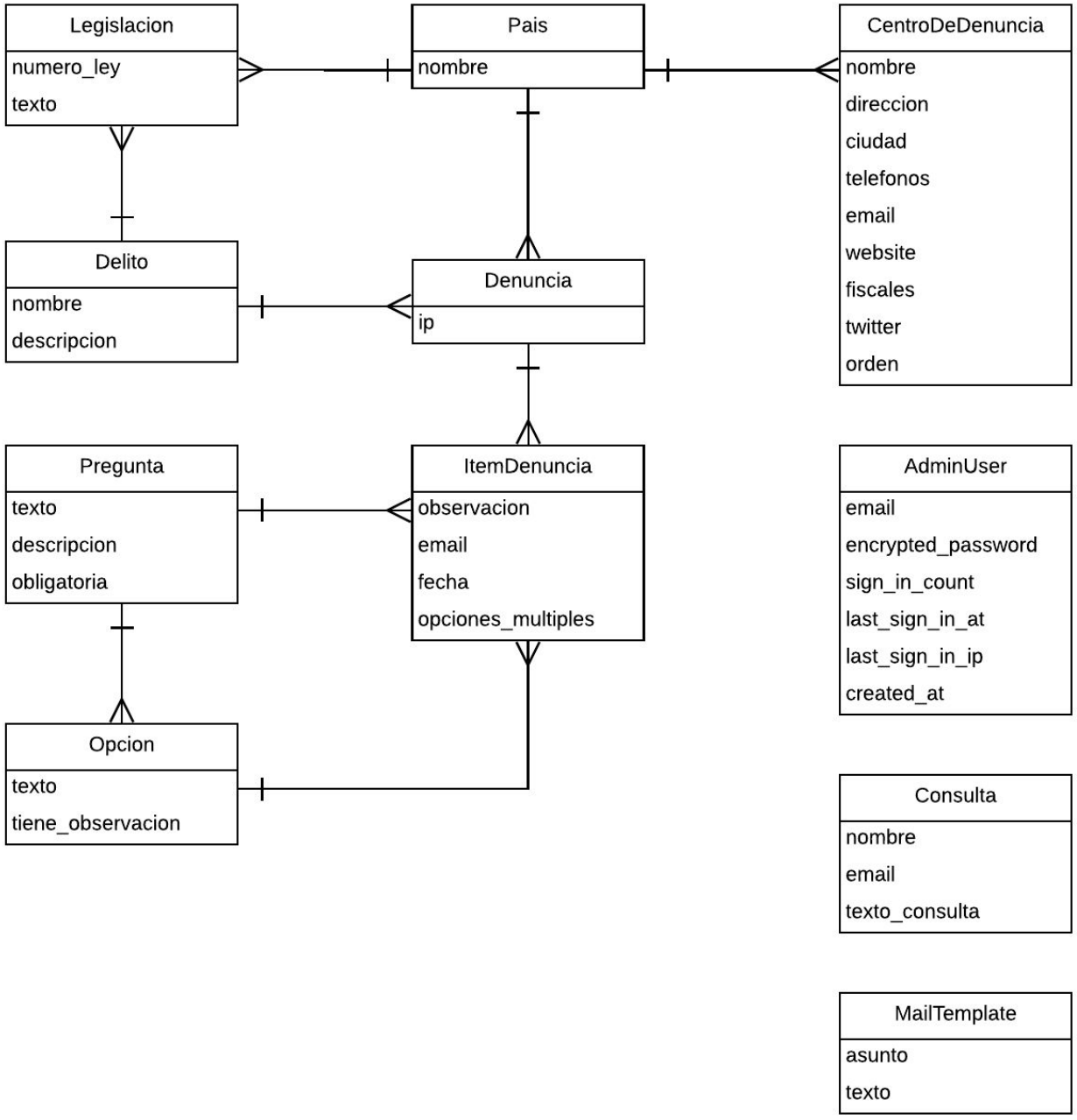



Figura 3.1.3. Modelo de datos

## 3.2. Tecnologías utilizadas para el desarrollo

En este capítulo se detallarán las herramientas y tecnologías que fueron utilizadas para desarrollar los servicios necesarios para alcanzar los objetivos planteados, agrupadas en temas relacionados.

### 3.2.1. Lenguajes de programación utilizados

#### 3.2.1.1 Javascript

Este lenguaje fue elegido para agregar comportamiento dinámico en la interfaz de usuario, por ejemplo para realizar acciones basadas en eventos de click, navegación, animaciones, etc.

JavaScript (JS) es un lenguaje de programación interpretado. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico.

Se utiliza principalmente en su forma del lado del cliente (client-side), implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas, aunque existe una forma de JavaScript del lado del servidor (Server-side JavaScript o SSJS).

JavaScript se diseñó con una sintaxis similar al C, aunque adopta nombres y convenciones del lenguaje de programación Java. Sin embargo Java y JavaScript no están relacionados y tienen semánticas y propósitos diferentes.

Todos los navegadores modernos interpretan el código JavaScript integrado en las páginas web. Para interactuar con una página web se provee al lenguaje JavaScript de una implementación del Document Object Model (DOM).

Tradicionalmente se venía utilizando en páginas web HTML para realizar operaciones y únicamente en el marco de la aplicación cliente, sin acceso a funciones del servidor.

JavaScript se interpreta en el agente de usuario, al mismo tiempo que las sentencias van descargándose junto con el código HTML.

#### 3.2.1.2 HTML5

Al igual que JavaScript, este lenguaje fue elegido para programar la aplicación web del lado cliente.

HTML5 (HyperText Markup Language, versión 5) es la quinta revisión importante del lenguaje básico de la World Wide Web, HTML. HTML5 especifica dos variantes de sintaxis para HTML: un «clásico» HTML (text/html), la variante conocida como HTML5 y una variante XHTML conocida como sintaxis XHTML5 que deberá ser servida como XML 1.2. Esta es la primera vez que HTML y XHTML se han desarrollado en paralelo.

Novedades:

- Incorpora etiquetas (canvas 2D y 3D, audio, vídeo) con códecs para mostrar los contenidos multimedia.
- Etiquetas para manejar grandes conjuntos de datos: Datagrid, Details, Menú y Command. Permiten generar tablas dinámicas que pueden filtrar, ordenar y ocultar contenido en cliente.
- Mejoras en los formularios. Nuevos tipos de datos (eMail, number, url, datetime) y facilidades para validar el contenido sin Javascript.
- Visores: MathML (fórmulas matemáticas) y SVG (gráficos vectoriales). En general se deja abierto a poder interpretar otros lenguajes XML.
- Drag & Drop. Nueva funcionalidad para arrastrar objetos como imágenes.

#### 3.2.1.3 Css

Las hojas de estilo en cascada o (Cascading Style Sheets, o sus siglas CSS) hacen referencia a un lenguaje de hojas de estilos usado para describir la presentación semántica (el

aspecto y formato) de un documento escrito en lenguaje de marcas. Su aplicación más común es dar estilo a páginas webs escritas en lenguaje HTML y XHTML.

La información de estilo puede ser adjuntada como un documento separado o en el mismo documento HTML.

CSS tiene una sintaxis muy sencilla, que usa unas cuantas palabras clave tomadas del inglés para especificar los nombres de varias propiedades de estilo.

#### 3.2.1.4 Ruby on Rails

Ruby es un lenguaje multiparadigma, utilizado para correr la lógica de aplicación en el servidor web. Utilizado junto con el framework para desarrollo web Rails, permite desarrollar una solución integral a cualquier tipo de aplicación o servicio web. Este conjunto (denominado Ruby on Rails) es considerado una de las herramientas más veloces en el mercado para desarrollar nuevos productos, debido a la cantidad de herramientas que proporciona, y su filosofía de convención por sobre la configuración.

Rails es un framework basado en arquitectura Modelo-Vista-Controlador, y está diseñado siguiendo todos los principios especificados en el Manifiesto Ágil, escrito por pioneros del desarrollo web entre los que figuran Kent Beck, Martin Fowler, Robert Martin, Dave Thomas, entre otros.

El uso de prácticas y patrones de diseño de programación orientada a objetos, Ruby permite escribir código de aplicación extremadamente expresivo y compacto, minimizando la repetición de código y reglas de negocio. Además cuenta con funcionalidades de metaprogramación, lo que consiste básicamente en escribir código que produce código. Esto permite diseñar abstracciones sobre las cuales se pueden basar los modelos de aplicación, y escribiendo pocas líneas de código se pueden obtener clases, objetos y métodos definidos automáticamente.

Entre los productos más exitosos construidos con Ruby on Rails se encuentran Twitter, AirBnB, Coachsurfing, GitHub, Groupon, Kickstarter, entre otros.

#### 3.2.1.5 SQL

El lenguaje de consulta estructurado o SQL (por sus siglas en inglés Structured Query Language) es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones en ellas. Una de sus características es el manejo del álgebra y el cálculo relacional que permiten efectuar consultas con el fin de recuperar de forma sencilla información de interés de bases de datos, así como hacer cambios en ella.

SQL es un lenguaje de acceso a bases de datos que explota la flexibilidad y potencia de los sistemas relacionales y permite así gran variedad de operaciones.

Es un lenguaje declarativo de "alto nivel" o "de no procedimiento" que, gracias a su fuerte base teórica y su orientación al manejo de conjuntos de registros (y no a registros individuales) permite una alta productividad en codificación y la orientación a objetos. De esta forma, una sola sentencia puede equivaler a uno o más programas que se utilizarían en un lenguaje de bajo nivel orientado a registros.

#### 3.2.1.7 UML

Lenguaje Unificado de Modelado (LUM o UML, por sus siglas en inglés, Unified Modeling Language) es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad; está respaldado por el OMG (Object Management Group). Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocio, funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y compuestos reciclados.

Es importante remarcar que UML es un "lenguaje de modelado" para especificar o para describir métodos o procesos. Se utiliza para definir un sistema, para detallar los artefactos en el sistema y para documentar y construir. En otras palabras, es el lenguaje en el que está descrito el modelo.

UML cuenta con varios tipos de diagramas, los cuales muestran diferentes aspectos de las entidades representadas.

### 3.2.2. APIs, librerías y plugins web

En esta sección se describen brevemente las APIs, librerías y plugins utilizados para el desarrollo del sistema.

#### 3.2.2.1 Librería jQuery

JQuery es una librería de JavaScript, que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones (FLV) y agregar interacción con la técnica AJAX a páginas web.

JQuery es software libre y de código abierto, posee un doble licenciamiento bajo la Licencia MIT y la Licencia Pública General de GNU v2, permitiendo su uso en proyectos libres y privativos. jQuery, al igual que otras bibliotecas, ofrece una serie de funcionalidades basadas en JavaScript que de otra manera requerirían de mucho más código, es decir, con las funciones propias de esta biblioteca se logran grandes resultados en menos tiempo y espacio.

#### 3.2.2.2 Twitter Bootstrap

Twitter Bootstrap ha sido escogido para la interfaz de la aplicación debido a su simpleza, su variedad de elementos y su facilidad de adaptación y uso.

Twitter Bootstrap es una colección de herramientas de software libre para la creación de sitios y aplicaciones web. Contiene plantillas de diseño basadas en HTML y CSS con tipografías, formularios, botones, gráficos, barras de navegación y demás componentes de interfaz, así como extensiones opcionales de JavaScript.

Bootstrap tiene soporte para HTML5 y CSS 3, y es compatible con la mayoría de los navegadores web.

Desde la versión 2.0 también soporta diseños sensibles (conocido como *responsive design)*. Esto significa que el diseño gráfico de la página se ajusta dinámicamente, tomando

en cuenta las características del dispositivo usado (computadoras, tabletas, teléfonos móviles).

Bootstrap es de código abierto y está disponible en GitHub. Los desarrolladores están motivados a participar en el proyecto y a hacer sus propias contribuciones a la plataforma.

Para usar Bootstrap en una página HTML, el desarrollador solo debe descargar la hoja de estilo Bootstrap CSS y enlazarla en el archivo HTML. Si el desarrollador también quiere usar los componentes de JavaScript, éstos deben estar referenciados junto con la librería jQuery en el documento HTML.

#### 3.2.2.3 Simple Form

Simple Form es una *gema* (librería escrita en Ruby) que simplifica mucho el trabajo de escribir formularios en lenguaje HTML, y se integra directamente con el framework Rails.

Su principal funcionalidad es la de generación de elementos de formulario como ser etiquetas, campos de texto, botones, botones de caja de selección múltiple, botones de radio de selección simple, entre otros. Estos elementos se generan programáticamente llamando a las APIs de la librería con código Ruby embebido en las vistas de la aplicación. La misma logra su objetivo a través de la detección inteligente de tipos de campos de un modelo (asociado a una tabla en la base de datos) y la generación automática de los campos asociados. Por ejemplo, si se trabaja sobre un modelo que contiene una clave foránea a la tabla País, la herramienta generará automáticamente una caja de selección con todos los países existentes.

Esta herramienta es de código libre y se encuentra bajo la licencia MIT. Al día de hoy es la más utilizada para la creación de formularios en el ecosistema Ruby on Rails.

#### 3.2.2.4 Devise

Devise es una librería de autenticación diseñada para acoplarse directamente con una aplicación basada en Rails. La misma provee de un sistema de registro de usuario, inicio de

sesión, recuperación y cambio de contraseña, e-mails de confirmación, entre otras funcionalidades.

Al delegar esta tarea presente en casi todas las aplicaciones web a una librería ya testeada y optimizada por miles de usuarios, se asegura de que el sistema resultante esté libre de bugs y vulnerabilidades. En otras épocas el manejo del registro de usuarios e inicio de sesión se hacía a medida de cada sistema, llevando a posibles casos de vulnerabilidades por inyección de código SQL, robo de *cookies*, entre otros. Utilizando este tipo de librerías se evitan todos estos problemas, y se reduce el tiempo de desarrollo del proyecto.

La librería ya incluye lógica, vistas, plantillas de e-mail y métodos de encriptación de contraseña por defecto. Aun así, todos estos componentes pueden ser extendidos para reemplazar aquellos que se quieran personalizar. El caso más típico es el de estilización y formateo del formulario de registro e inicio de sesión.

#### 3.2.2.5 CanCanCan

Esta *gema* provee los servicios de autorización de usuario. A diferencia de la librería antes mencionada, esta no provee servicios de autenticación y registro de usuario, sino que se integra con la misma para proveer funcionalidades de definición de roles y permisos.

Permite al desarrollador definir un esquema de roles, y asignar permisos a cada uno sobre los distintos recursos y entidades del sistema. En el caso de este proyecto se utilizará para distinguir entre usuarios visitantes del sitio y usuarios administradores. Los últimos tienen acceso privilegiado al panel de administración a través del cual pueden manejar todas las entidades del sistema.

Al igual que todas las herramientas utilizadas para el desarrollo de este proyecto, CanCanCan es de código abierto y está protegido bajo la licencia MIT.

#### 3.2.2.6 ActiveAdmin

Esta librería se integra perfectamente con el framework Rails, y facilita la generación de paneles de administración reduciendo el código de aplicación y configuración necesarios para la implementación de esta funcionalidad.

La misma cuenta con porciones de código y vistas predefinidos que facilitan la tarea de generación de funcionalidades de alta, baja, modificación y listado de entidades. De esta manera el desarrollador se ahorra mucho tiempo en la tarea repetitiva de implementar estas acciones para cada modelo.

Similar a otras librerías utilizadas, ActiveAdmin se basa en un conjunto de componentes generados por defecto, y permite su extensión a través de la definición de subclases y vistas personalizadas. En este proyecto se hace uso extensivo de esta librería, tanto de sus componentes generados por defecto, como de componentes propios integrados a la herramienta.

### 3.2.3. Base de datos

La base de datos elegida para el nuevo sistema es PostgreSQL, un sistema de gestión de bases de datos relacional orientado a objetos y libre, publicado bajo la licencia PostgreSQL similar a la BSD o la MIT.

Como muchos otros proyectos de código libre, el desarrollo de PostgreSQL no es manejado por una empresa o persona, sino que es dirigido por una comunidad de desarrolladores que trabajan de forma desinteresada. Dicha comunidad es denominada el PGDG (PostgreSQL Global Development Group).

La elección de esta base de datos se debe a que se cuenta con experiencia previa trabajando con la misma, y que es la más recomendada para trabajar con la pila de tecnologías elegidas (principalmente Heroku y Rails).

Las principales prestaciones de este motor de base de datos son las siguientes:

- Alta concurrencia: mediante un sistema denominado MVCC (Acceso concurrente multiversión, por sus siglas en inglés) PostgreSQL permite que mientras un proceso escribe en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos.
- Amplia variedad de tipos nativos: soporta números de precisión arbitraria, textos de largo indefinido, listas, direcciones IP, entre otros.
- Claves foráneas: esta funcionalidad, presente en la mayoría de las bases de datos relacionales, permite lograr la integridad relacional en la base de datos cuando se asocian múltiples tablas a través de claves.
- Disparadores o *triggers*: acciones específicas que se realizan de acuerdo a un evento, cuando éste ocurra dentro de la base de datos.
- Transacciones: conjunto de sentencias SQL que se ejecutan de forma atómica, dejando a la base de datos siempre en un estado consistente.
- Funciones: bloques de código que se ejecutan en el servidor de base de datos. Pueden ser escritos en varios lenguajes, desde las operaciones básicas de programación, hasta las complejidades de la programación orientada a objetos o la programación funcional.

### 3.2.4. Herramientas de desarrollo

**Ubuntu Linux**

Ubuntu es un sistema operativo de código abierto, basado en la distribución Debian. El mismo tiene disponibles versiones para escritorio y para servidores, siendo hoy en día una de las distribuciones más populares para ambos casos de uso. El mismo está desarrollado por una comunidad de desarrolladores e impulsado y mantenido por la empresa Canonical.

Este sistema operativo se utilizó tanto para el desarrollo del sistema (ambiente de desarrollo), como para su ejecución en el servidor provisto por Heroku (ambiente de producción). Esto ayuda a minimizar las posibles incompatibilidades que puedan existir entre

paquetes, librerías y ejecutables cuando se desarrolla un sistema en un determinado sistema operativo y luego se despliega sobre un servidor que corre otro distinto.

**Vim**

Vim es un editor de texto de propósito general, que se puede utilizar para escribir código fuente de cualquier lenguaje de programación. A través de la instalación de complementos, permite el soporte de sintaxis de estos lenguajes, así como también las funcionalidades típicas encontradas en un entorno de desarrollo integrado (IDE). Estas funcionalidades van desde el autocompletado del código, búsqueda y reemplazo de símbolos, análisis estático del código, ejecución automática de pruebas unitarias, entre otros.

A diferencia de un IDE tradicional, Vim requiere ser configurado y extendido manualmente para obtener dichas funcionalidades, pero esto a su vez brinda una enorme flexibilidad ya que con una simple herramienta se pueden cubrir todas las necesidades de escritura de código fuente, configuraciones, documentación, etc.

Vim es el sucesor del editor Vi, creado en 1976. Por ello cuenta con 42 años de desarrollo y optimización por su comunidad de desarrolladores, y está publicado bajo la licencia GPL.

**Heroku Toolbelt**

Los desarrolladores de la plataforma Heroku brindan a sus usuarios una herramienta de línea de comandos llamada Toolbelt. La misma es utilizada a través de una terminal, y permite administrar y configurar íntegramente las instancias de servidores de aplicación alojados en el mismo. Dentro de las principales funcionalidades se encuentran:

- Crear nuevas aplicaciones: Esto proveerá un servidor virtual asignado para ejecutar una aplicación servidor. Cuenta con detección automática de tecnologías y servicios, por lo que ya incluyen preinstalados los paquetes necesarios para cada lenguaje (PHP, Ruby, Javascript, Java, etc).
- Administrar complementos: A cada aplicación se le pueden adicionar servicios como ser bases de datos, servidores de caché, herramientas de logging, sistemas de

mensajería, entre otros. Estos vienen en planes gratuitos y planes pagos, dependiendo de la calidad de servicio requerida.

- Detener, reiniciar, y escalar aplicaciones: A través de un simple comando se pueden escalar horizontalmente la cantidad de servidores en los que corre una aplicación. Esto permite por ejemplo, pasar de tener dos instancias a tener cinco al instante, para hacer frente a una demanda poco usual originada por algún evento cíclico (Black Friday, Navidad, etc). El cobro se realiza por hora, por lo que se pueden aumentar las instancias y luego reducirlas cuando ya no se necesitan más.
- Configurar variables de entorno
- Crear, descargar y restablecer Backups de bases de datos

**Rubocop**

Rubocop es una herramienta de análisis y formateo de código, basada en la guía de estilo oficial del lenguaje Ruby. El mismo cuenta con más de cien reglas que se pueden activar o desactivar individualmente, referidas a cómo deben escribirse los archivos de código fuente. Las principales reglas tienen que ver con la longitud máxima de caracteres en una línea de código, cantidad de sentencias por método, cantidad total de líneas de cada clase, alineación de parámetros, convenciones de nombres de variables y métodos, entre otras.

Se considera buena práctica configurar estas reglas, y solo permitir el ingreso de código a la base cuando no se viole ninguna de las mismas.

Además del análisis del código, tiene una funcionalidad de autocorrección, con la que la mayoría de los errores de estilo son corregidos automáticamente. Aun así, algunas reglas son imposibles de corregir por el programa y requieren la intervención del programador.

**Rspec**

Este es el framework de testing más sofisticado que existe actualmente para la pila de tecnologías Ruby/Rails, y es el más usado en aplicaciones comerciales. El mismo está diseñado para desarrollar aplicaciones utilizando el patrón de diseño basado en

comportamiento (BDD - Behaviour Driven Design). Para esto provee una API muy extensa para la definición de casos de prueba automatizados.

Esta herramienta permite la definición de casos de prueba para interfaz, controladores, modelos e integración (conocidos como end-to-end). Además de proveer clases, métodos y un DSL (Domain Specific Language) para la definición de los casos de prueba, cuenta con herramientas de línea de comandos para la ejecución de la suite de pruebas completa, ejecución de un tipo particular de pruebas (unitarias, de integración, etc), monitorización y listado de los casos de prueba más lentos, reporte de porcentaje de cobertura de pruebas, etc.

Una utilización apropiada de esta herramienta asegura la calidad en el producto de software generado, una gran tolerancia a las fallas, y flexibilidad a la hora de hacer refactorización del código fuente.

### 3.2.5. Sistema de control de versiones

**Git**

El sistema de control de versiones utilizado es Git. El mismo es gratuito, de código abierto, y apto para utilizar tanto en pequeños proyectos como en sistemas a gran escala. A diferencia de otros sistemas más tradicionales como SVN o Mercurial, Git es descentralizado. Esto significa que cada nodo posee una réplica completa del repositorio de código, con todo su historial de commits, versiones, etc. El nodo central es elegido sólo por convención, y suele ser alojado en un servidor que provee este servicio, como ser Github, Gitlab o BitBucket. Esto significa que dicho nodo se utilizará como referencia para enviar y obtener los últimos cambios en el código, pero en realidad el repositorio no tiene ninguna diferencia con el resto de los nodos. Por ejemplo, un desarrollador podría obtener los últimos cambios en el código a través de LAN solicitándolos a un host de un compañero de trabajo.

Hoy en día Git es el sistema de control de versiones más utilizado en sistemas modernos, dado que supera en todos los aspectos a las herramientas centralizadas. Las principales funcionalidades y ventajas son:

- Soporte de múltiples ramas independientes: Facilita que múltiples equipos de trabajo puedan enfocarse en funcionalidades diferentes y trabajarlas independientemente una de otras, partiendo de la rama base (denominada *master* por convención). Luego los mismos pueden solicitar la fusión a la rama maestra.
- Liviano y rápido: Al realizar todas las operaciones localmente en vez de interactuar con un servidor, Git es extremadamente rápido. Los Benchmarks realizados reportan que en las operaciones cotidianas de control de versiones, Git puede ser entre 4 y 100 veces más rápido que SVN, dependiendo del tipo de operación.
- Integridad y seguridad: Cada archivo y cada *commit* realizado en Git contiene una suma de verificación SHA-1 de 160 bits, por lo que es tecnológicamente imposible que alguien corrompa un repositorio y modifique el código fuente en el contenido.

## 3.3. Aplicación

Como se ha mencionado anteriormente, el sistema desarrollado consta de una aplicación web, cuya interfaz con los usuarios es un navegador web. La misma cuenta con distintas funcionalidades, algunas accesibles a los visitantes del sitio y otras a los administradores del sistema. En las secciones subsiguientes se explicará cómo se despliega el sistema en el entorno productivo, y cuales son las funcionalidades existentes,

### 3.3.1. Despliegue y configuración

En esta sección se detallarán los pasos a seguir para configurar y desplegar el sistema en la plataforma heroku, empezando de cero. Las condiciones previas son contar con acceso al repositorio de Github donde se aloja el código fuente del sistema, contar con una cuenta en Heroku, y los valores de las variables de entorno que se van a configurar en el mismo.

**Git Clone**

El primer paso es clonar el repositorio de git alojado en Github. Para esto debemos contar con la interfaz de línea de comandos de git instalada en nuestro sistema. Una vez que contemos con el URL de nuestro repositorio, procedemos a invocar el comando con la sintaxis *git clone URL*.

```
~/webapps $ git clone git@github.com:antico5/odila.git
Cloning into 'odila'...
remote: Counting objects: 1094, done.
remote: Total 1094 (delta 0), reused 0 (delta 0), pack-reused 1094
Receiving objects: 100% (1094/1094), 4.47 MiB | 2.15 MiB/s, done.
Resolving deltas: 100% (612/612), done.
```

Figura 3.3.1a. Git Clone

**Heroku app create**

Una vez tengamos el código fuente en nuestro sistema local, procederemos a crear una nueva aplicación en la plataforma Heroku. Esto nos proveerá de una instancia (llamadas *dynos*) a la que podremos desplegar el código fuente.

El primer paso es instalar el *toolbelt* de Heroku, que es un conjunto de herramientas de línea de comandos para administrar los servicios del mismo. Una vez instalado el toolbelt, podremos ejecutar *heroku login* para iniciar sesión en el sistema.

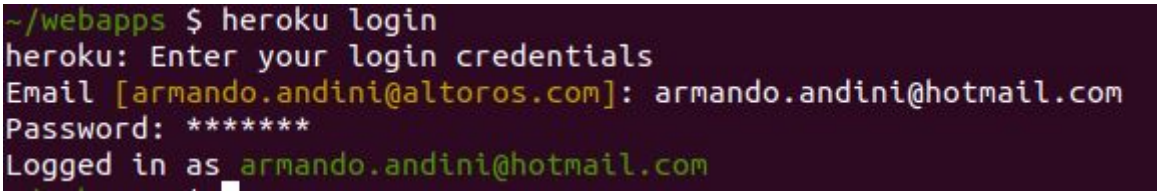



Figura 3.3.1b. Heroku login

Una vez autenticados en el sistema, procedemos a crear una nueva aplicación, con el comando *heroku apps:create nombre_aplicación*. En este caso asignaremos el nombre odila_test a la aplicación.

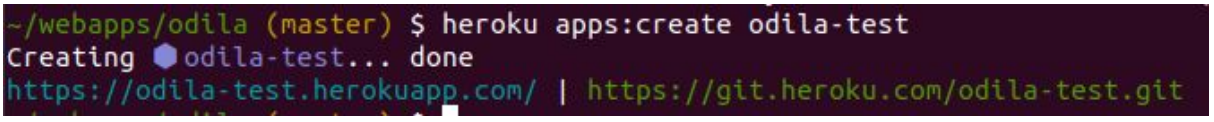



Figura 3.3.1c. Heroku app create

Con esto ya tendremos creada una instancia gratuita para ejecutar nuestra aplicación, accesible a través de la URL [https://odila-test.herokuapp.com/](https://odila-test.herokuapp.com/).

**Procfile**

El siguiente paso es crear un archivo en el cual se especificará cuáles son los procesos que debe correr el servidor, con sus nombres y parámetros. Esto es necesario ya que por lo general se ejecutan otros procesos paralelamente al servidor web (por ejemplo, *workers* o

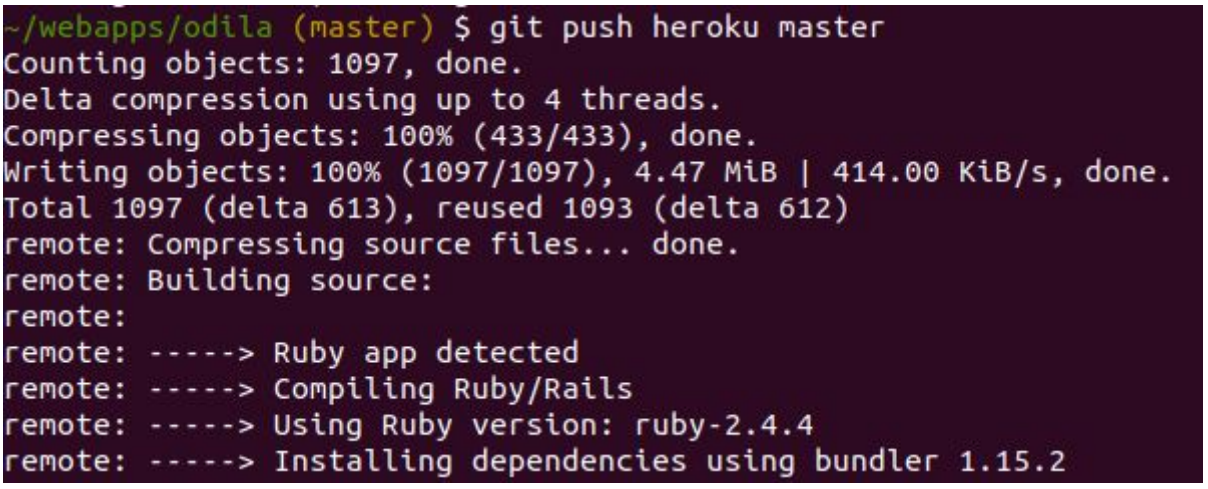
colas de trabajo). Para la implementación de este sistema solo es necesario crear una entrada para el servidor web. El contenido del archivo es el siguiente:

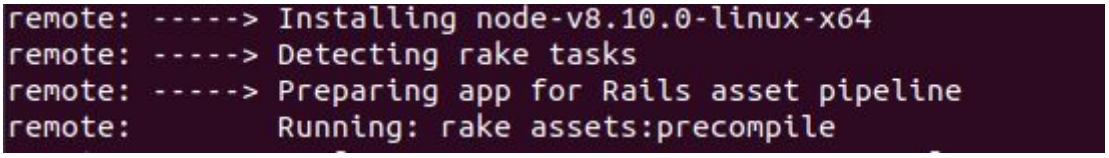
*web: bundle exec puma -e production*

**Git push**

El siguiente paso es realizar una operación *git push* del repositorio local al repositorio remoto provisto por heroku. Como se mencionó anteriormente, Git es un sistema de control de versiones distribuido, por lo que cada nodo cuenta con el repositorio completo y realiza operaciones de *pull (*obtener cambios remotos) y *push* (enviar cambios a otro repositorio) de forma independiente.

El sistema heroku detectará esta operación de *push* e instalará todos los paquetes y dependencias necesarias, luego correrá una serie de tareas de precompilación, para que el sistema pueda ejecutarse en modo producción de forma óptima. Si todas las tareas se ejecutan exitosamente, la aplicación ya estará disponible para accederla por URL pública.



```
remote: -----> Discovering process types
remote:        Procfile declares types     -> web
remote:        Default types for buildpack -> console, rake, worker
remote:
remote: -----> Compressing...
remote:        Done: 42.3M
remote: -----> Launching...
remote:        Released v5
remote:        https://odila-test.herokuapp.com/ deployed to Heroku
remote:
remote: Verifying deploy... done.
To https://git.heroku.com/odila-test.git
 * [new branch]      master -> master
```

Figura 3.3.1d. Heroku deployment

**Configuración de variables de entorno**

Este es el último paso, ya que el sistema se encuentra ejecutándose en la plataforma Heroku, solo falta configurar las variables de entorno necesarias para el normal funcionamiento del mismo. Ciertos valores de configuración como ser cuentas de e-mail, claves secretas de APIs, URLs de bases de datos, entre otros, no son incluídos en el repositorio de Git, dado que si el mismo es de acceso público (en el caso de proyectos open source), o se encuentra comprometido, esta información sensible puede caer en manos de usuarios indeseados. Por esto mismo, en el código fuente se referencian variables de entorno del sistema, por lo que su valor puede ser configurado independientemente en cada plataforma de ejecución.

Heroku nos permite realizar esta operación a través del comando *heroku config*. Más específicamente, para establecer valores para dichas variables de entorno hay que ejecutar el comando *heroku config:set VARIABLE=valor*.

Una vez concluidos estos pasos, el sistema se encontrará desplegado en su totalidad, y se encontrará en etapa de producción.

```
~/webapps/odila (master) $ heroku config:set ADMIN_ROUTE=de004086d3a8c1f1844f706
Setting ADMIN_ROUTE and restarting ⬢ odila-test... done, v6
ADMIN_ROUTE: de004086d3a8c1f1844f706
~/webapps/odila (master) $ heroku config:set MAILER_DOMAIN=gmail.com
Setting MAILER_DOMAIN and restarting ⬢ odila-test... done, v7
MAILER_DOMAIN: gmail.com
~/webapps/odila (master) $ heroku config:set MAILER_PASSWORD=**********
Setting MAILER_PASSWORD and restarting ⬢ odila-test... done, v8
MAILER_PASSWORD: **********
~/webapps/odila (master) $ heroku config:set MAILER_USER=********
Setting MAILER_USER and restarting ⬢ odila-test... done, v9
MAILER_USER: ********
```

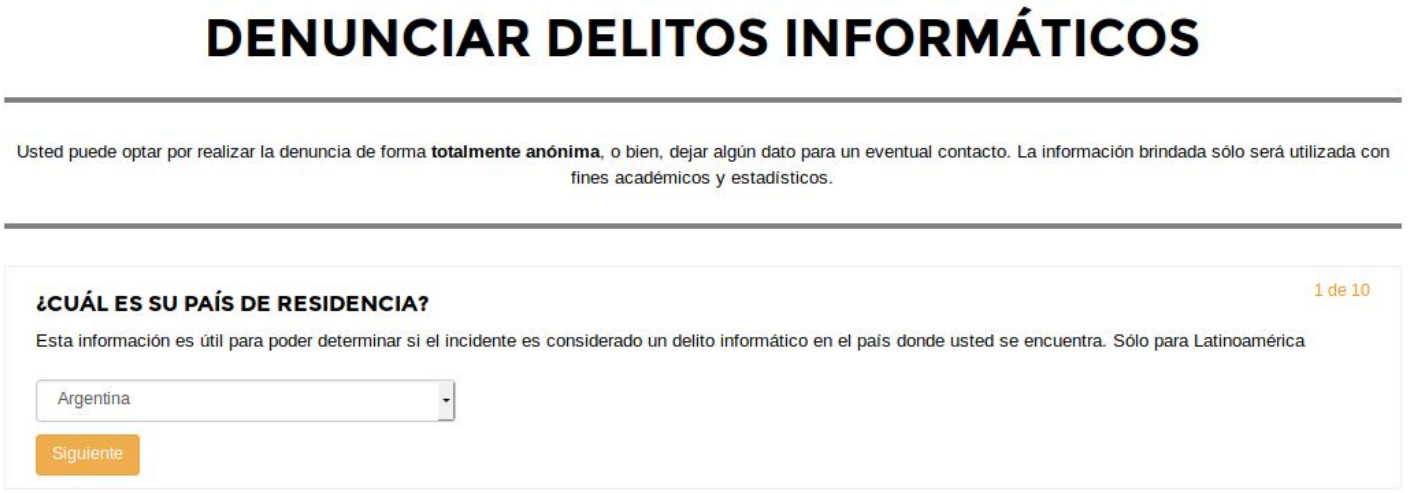
Figura 3.3.1e. Heroku config set

### 3.3.2. Funcionalidades

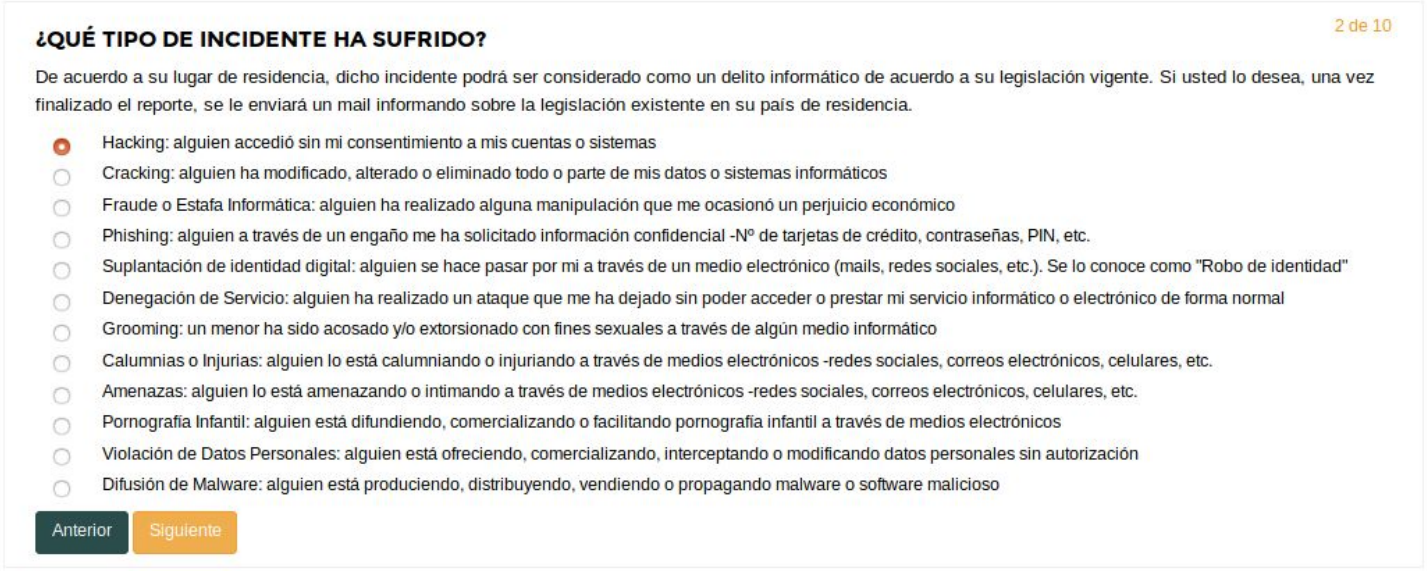
#### 3.3.2.1 Realizar denuncia

Esta funcionalidad es accesible desde la pantalla principal, haciendo click en el enlace “Denuncia”, o desplazándose hacia abajo con la rueda del mouse. Se presenta el formulario de denuncia con las preguntas básicas más las agregadas por los administradores del sitio.

Las primeras preguntas son las más relevantes, el país de origen y el incidente sufrido.

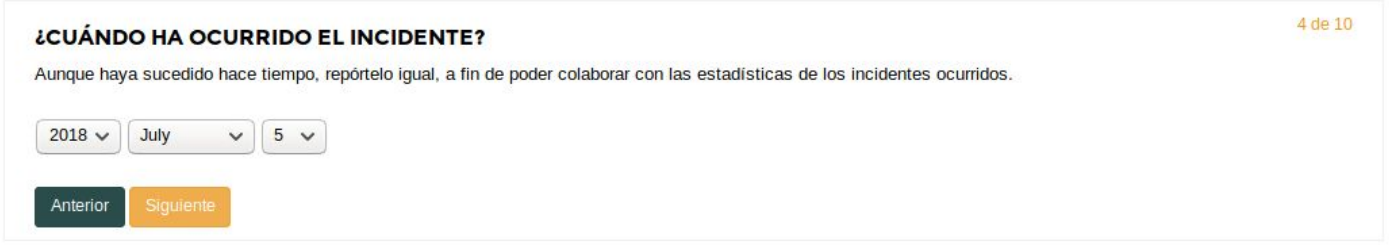


Figuras 3.3.2.1. Realizar denuncia





Como se puede observar en la imagen anterior, algunas preguntas son opcionales y presentan el boton “Saltear” para omitir la pregunta.



5 de 10

**¿HA DENUNCIADO EL DELITO ANTE LOS ORGANISMOS COMPETENTES EN SU PAÍS?**

Ej: Denuncia policial, denuncia ante la fiscalía de turno, etc.

- ○ Si, ya denuncie y la investigación está en curso
- ○ Si, denuncie pero la investigación no avanzó
- ○ Sí, denuncié y ya se ha condenado al o los culpables
- ◉ No, no denuncié porque... (ver siguiente pregunta)

Anterior Siguiente

6 de 10

**EN EL CASO QUE NO HAYA REALIZADO LA DENUNCIA, ¿SE DEBE A ALGUNA DE LAS SIGUIENTES CAUSAS?**

Puede elegir una o más causas.

- ☐ Ya denuncié
- ☐ No creo en la Policía ni en la Justicia Penal
- ☑ No me considero víctima de un delito
- ☐ No quiero difundir públicamente el incidente (pérdida de confidencialidad)
- ☐ Tengo temor de futuras represalias de parte del autor
- ☑ No creo que la investigación tenga éxito
- ☐ No estoy seguro de haber sido víctima de un delito penal
- ☐ En parte me siento culpable por el incidente
- ☐ No creo que la denuncia sea útil, porque el sistema penal no es apto para combatir el cibercrimen
- ☐ Otros

Anterior Siguiente

7 de 10

**EDAD**

Esta información es opcional, pero si usted colabora nos ayudará a comprender cuáles son los sectores sociales más atacados por los ciberdelincuentes.

- ○ Menores de 21 años
- ◉ Entre 22 y 35 años
- ○ Entre 36 y 45 años
- ○ Más de 45 años

Anterior Siguiente Saltear

8 de 10

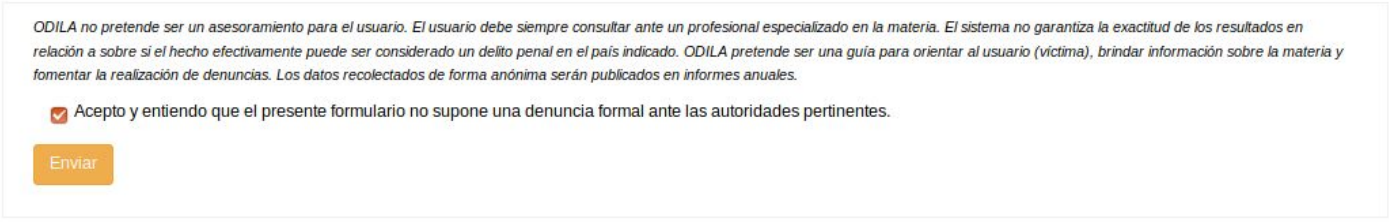
**GÉNERO**

Esta información es opcional, pero si usted colabora nos ayudará a comprender cuáles son los sectores sociales más atacados por los ciberdelincuentes.

- ◉ Masculino
- ○ Femenino

Anterior Siguiente Saltear






#### 3.3.2.2 Glosario de seguridad informática

Esta sección consiste en una lista de definiciones de conceptos relacionados a la seguridad informática. La investigación sobre los mismos y sus correspondientes definiciones fueron proporcionados por los clientes, que son especialistas en el área. Esta sección incluye los siguientes términos: *amenaza, calumnia, cibercrimen, ciberdelito, cifra negra, cracking, dato personal, delito informático, denuncia, estafa, fraude, grooming, hacking, identidad, injuria, malware, pederastia, pedofilia, phishing, pornografía infantil, robo de identidad, spam* y *suplantación de identidad.*

### 3.3.2.3 Información del proyecto

En esta sección se muestra información relevante al proyecto ODILA y a sus integrantes. El contenido de esta página es estático y la información fue provista por los clientes del sistema.


Figura 3.3.2.3. Información del proyecto

### 3.3.2.4 Formulario de contacto

Esta funcionalidad es accesible para cualquier usuario que visite la aplicación web. Permite a los mismos realizar una consulta al equipo de ODILA a través de un formulario de

contacto. Los administradores del sitio tienen dos formas de notificarse de las nuevas consultas realizadas en el sistema: a través de un e-mail generado automáticamente, y a través del panel de administración, en la sección correspondiente a consultas. Los mismos pueden proceder a contestar dichas consultas a través de un mensaje de e-mail convencional.

A continuación se presenta la interfaz de consulta. Todos los datos son obligatorios, y se muestran mensajes de error en caso que alguno no esté presente:

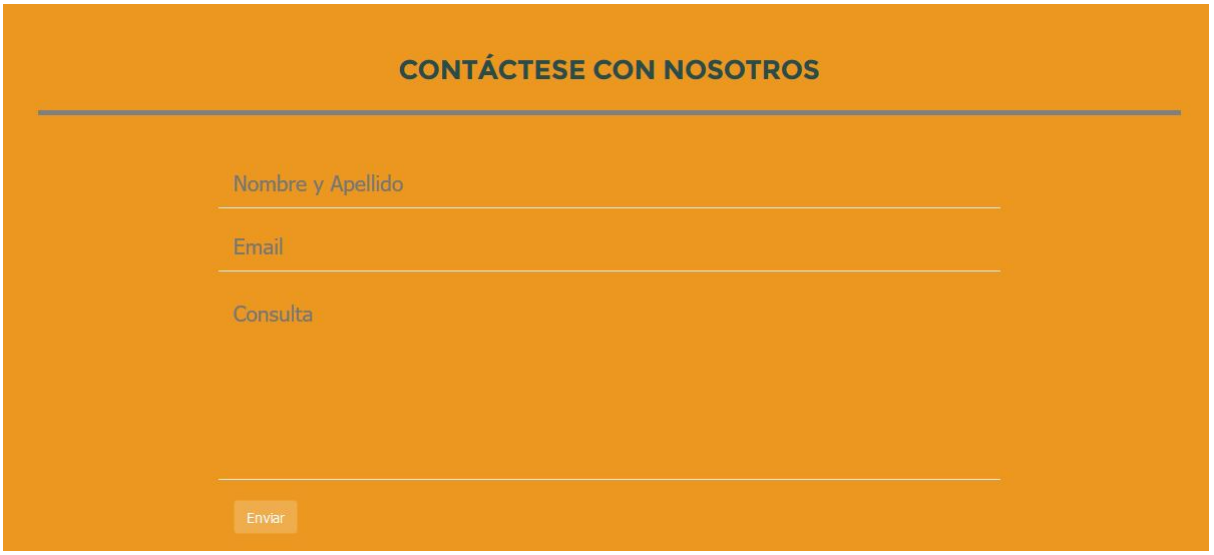
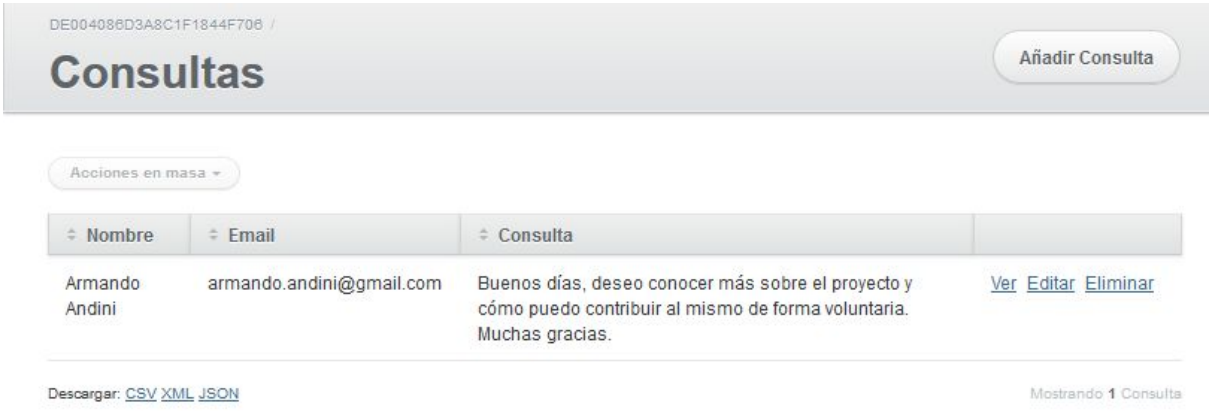



Figura 3.3.2.4. Formulario de contacto

La interfaz del panel de administración para listar las consultas existentes es la siguiente:



### 3.3.2.5 Visualización de reportes anteriores

A través de un menú desplegable en la barra de navegación, los usuarios pueden acceder a reportes de años anteriores elaborados por el equipo del proyecto ODILA. Los mismos se presentan con un lector dinámico de archivos PDF, para poder navegar el contenido sin tener que salir de la web. Además se muestra un enlace para que el que desee descargar los informes lo pueda hacer opcionalmente.

Figura 3.3.2.5. Reportes de años anteriores

### 3.3.2.6 Visualización del resultado de la denuncia

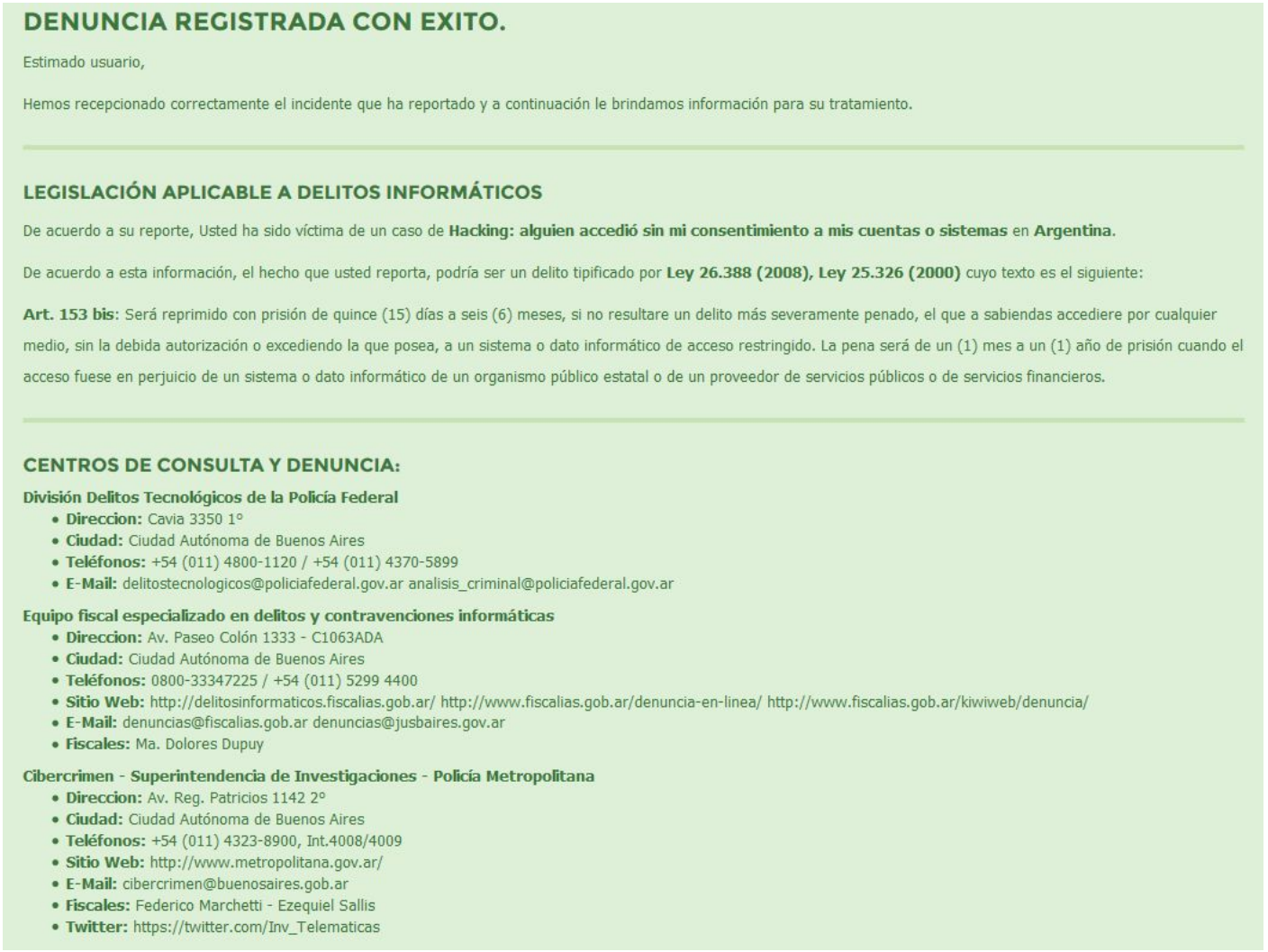
Cuando un usuario completa el formulario de denuncia, debe recibir una respuesta generada por el sistema en la que se brinde asesoramiento sobre qué leyes y artículos protegen al mismo de la injuria sufrida, y en qué lugares puede realizar una denuncia formal sobre el evento. Para generar esta respuesta, el sistema consume datos de diversas fuentes, entre las principales se encuentran:

- Tabla de registros de delitos informáticos
- Tabla de Países soportados
- Tabla de Legislaciones, que vincula un registro de tipo Delito, con uno del tipo País, e incluye información sobre los números de leyes y artículos que tipifican dicho delito
- Tabla de centros de denuncias, que son vinculados a los registros de País

Estas tablas son consultadas enviando parámetros obtenidos de las selecciones que el usuario realizó en el formulario de denuncia. Las preguntas que tienen relevancia para la generación de este resultado son las dos primeras: el país de origen de la víctima y el tipo de delito sufrido.

Estos datos, consumidos de la instancia de base de datos PostgreSQL, son luego procesados para generar una respuesta al usuario, siempre y cuando se encuentre con información almacenada sobre las legislaciones aplicables.

La respuesta del sistema es la siguiente:



## DENUNCIA REGISTRADA CON EXITO.

Estimado usuario,

Hemos recepcionado correctamente el incidente que ha reportado y a continuación le brindamos información para su tratamiento.

### LEGISLACIÓN APLICABLE A DELITOS INFORMÁTICOS

De acuerdo a su reporte, Usted ha sido víctima de un caso de **Hacking: alguien accedió sin mi consentimiento a mis cuentas o sistemas** en **Argentina**.

De acuerdo a esta información, el hecho que usted reporta, podría ser un delito tipificado por **Ley 26.388 (2008), Ley 25.326 (2000)** cuyo texto es el siguiente:

**Art. 153 bis**: Será reprimido con prisión de quince (15) días a seis (6) meses, si no resultare un delito más severamente penado, el que a sabiendas accediere por cualquier medio, sin la debida autorización o excediendo la que posea, a un sistema o dato informático de acceso restringido. La pena será de un (1) mes a un (1) año de prisión cuando el acceso fuese en perjuicio de un sistema o dato informático de un organismo público estatal o de un proveedor de servicios públicos o de servicios financieros.

### CENTROS DE CONSULTA Y DENUNCIA:

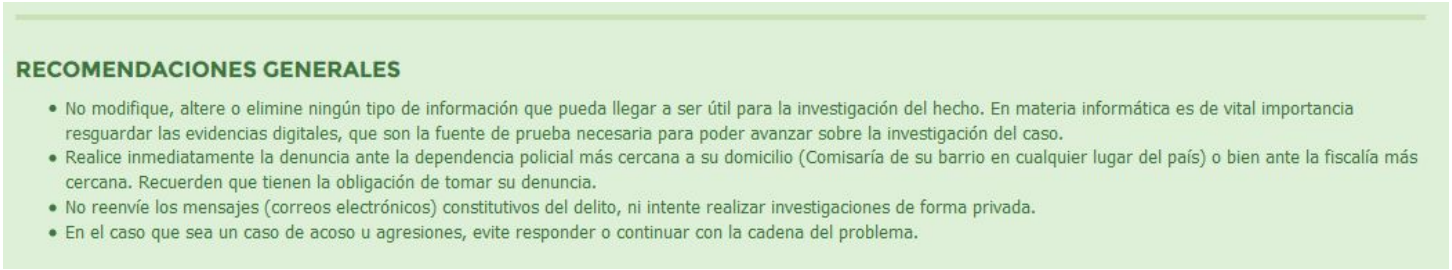
**División Delitos Tecnológicos de la Policía Federal**
- **Direccion:** Cavia 3350 1°
- **Ciudad:** Ciudad Autónoma de Buenos Aires
- **Teléfonos:** +54 (011) 4800-1120 / +54 (011) 4370-5899
- **E-Mail:** delitostecnologicos@policiafederal.gov.ar analisis_criminal@policiafederal.gov.ar

**Equipo fiscal especializado en delitos y contravenciones informáticas**
- **Direccion:** Av. Paseo Colón 1333 - C1063ADA
- **Ciudad:** Ciudad Autónoma de Buenos Aires
- **Teléfonos:** 0800-33347225 / +54 (011) 5299 4400
- **Sitio Web:** http://delitosinformaticos.fiscalias.gob.ar/ http://www.fiscalias.gob.ar/denuncia-en-linea/ http://www.fiscalias.gob.ar/kiwiweb/denuncia/
- **E-Mail:** denuncias@fiscalias.gob.ar denuncias@jusbaires.gov.ar
- **Fiscales:** Ma. Dolores Dupuy

**Cibercrimen - Superintendencia de Investigaciones - Policía Metropolitana**
- **Direccion:** Av. Reg. Patricios 1142 2°
- **Ciudad:** Ciudad Autónoma de Buenos Aires
- **Teléfonos:** +54 (011) 4323-8900, Int.4008/4009
- **Sitio Web:** http://www.metropolitana.gov.ar/
- **E-Mail:** cibercrimen@buenosaires.gob.ar
- **Fiscales:** Federico Marchetti - Ezequiel Sallis
- **Twitter:** https://twitter.com/Inv_Telematicas

Figura 3.3.2.6. Resultado de denuncia

La lista de centros de denuncia continúa, pero fue recortada para economizar espacio. Al final de dicha respuesta, además se presenta una serie de recomendaciones generales para evitar que el usuario vuelva a incurrir en un incidente similar:



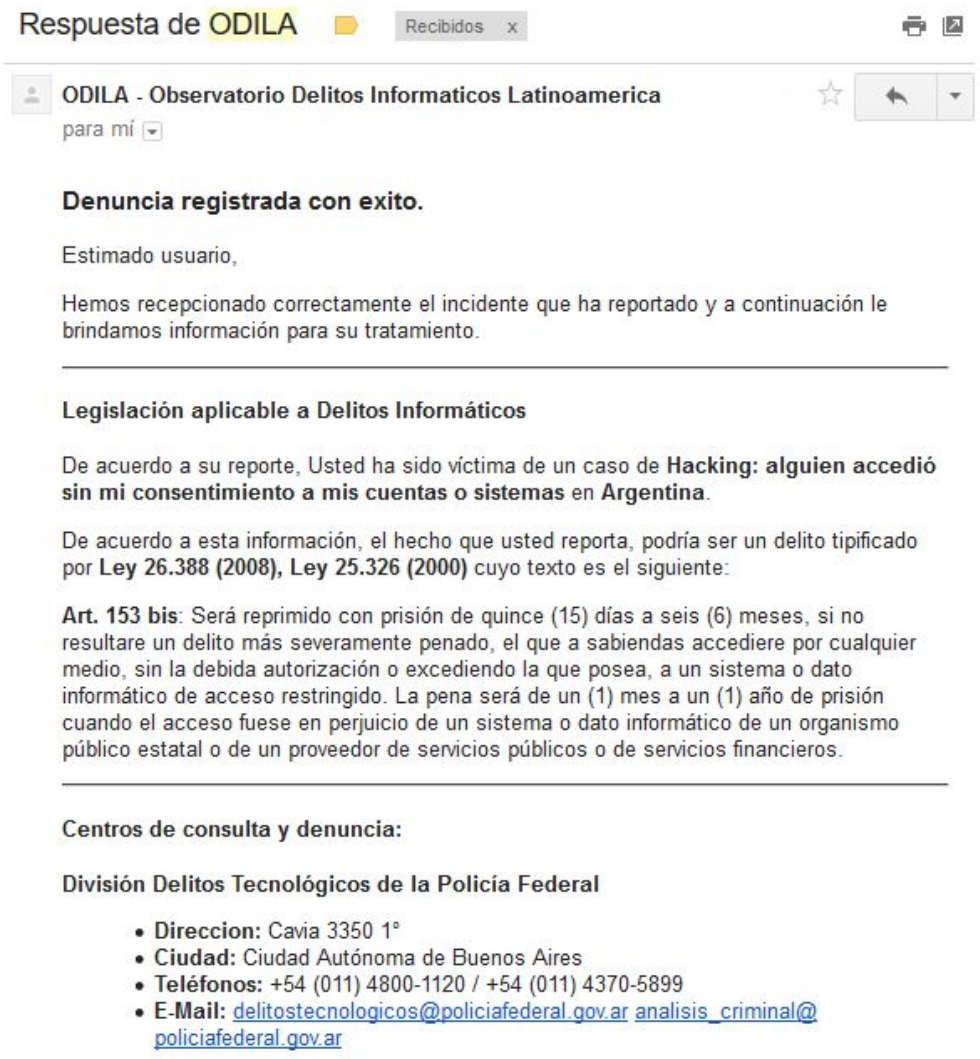
### RECOMENDACIONES GENERALES

- No modifique, altere o elimine ningún tipo de información que pueda llegar a ser útil para la investigación del hecho. En materia informática es de vital importancia resguardar las evidencias digitales, que son la fuente de prueba necesaria para poder avanzar sobre la investigación del caso.
- Realice inmediatamente la denuncia ante la dependencia policial más cercana a su domicilio (Comisaría de su barrio en cualquier lugar del país) o bien ante la fiscalía más cercana. Recuerden que tienen la obligación de tomar su denuncia.
- No reenvíe los mensajes (correos electrónicos) constitutivos del delito, ni intente realizar investigaciones de forma privada.
- En el caso que sea un caso de acoso u agresiones, evite responder o continuar con la cadena del problema.

#### 3.3.2.7 E-mail de respuesta automatizada

Cuando el usuario completa su denuncia, además de recibir una respuesta en el navegador web (como se muestra en el inciso anterior), también recibirá un email en su casilla de correo con el mismo contenido. Esta funcionalidad es necesaria, ya que un usuario puede querer acceder nuevamente a la respuesta luego de cerrar su navegador, o cerrar la pestaña en la que había realizado la denuncia.

Como se puede observar, la respuesta en formato de email presenta un estilo más simple que la respuesta web, pero contiene la misma información:



Respuesta de ODILA Recibidos x

ODILA - Observatorio Delitos Informaticos Latinoamerica
para mí

**Denuncia registrada con exito.**

Estimado usuario,

Hemos recepcionado correctamente el incidente que ha reportado y a continuación le brindamos información para su tratamiento.

---

**Legislación aplicable a Delitos Informáticos**

De acuerdo a su reporte, Usted ha sido víctima de un caso de **Hacking: alguien accedió sin mi consentimiento a mis cuentas o sistemas** en **Argentina**.

De acuerdo a esta información, el hecho que usted reporta, podría ser un delito tipificado por **Ley 26.388 (2008), Ley 25.326 (2000)** cuyo texto es el siguiente:

**Art. 153 bis**: Será reprimido con prisión de quince (15) días a seis (6) meses, si no resultare un delito más severamente penado, el que a sabiendas accediere por cualquier medio, sin la debida autorización o excediendo la que posea, a un sistema o dato informático de acceso restringido. La pena será de un (1) mes a un (1) año de prisión cuando el acceso fuese en perjuicio de un sistema o dato informático de un organismo público estatal o de un proveedor de servicios públicos o de servicios financieros.

---

**Centros de consulta y denuncia:**

**División Delitos Tecnológicos de la Policía Federal**

- **Direccion:** Cavia 3350 1°
- **Ciudad:** Ciudad Autónoma de Buenos Aires
- **Teléfonos:** +54 (011) 4800-1120 / +54 (011) 4370-5899
- **E-Mail:** delitostecnologicos@policiafederal.gov.ar analisis_criminal@policiafederal.gov.ar

Figura 3.3.2.7. Email de respuesta

### 3.3.2.8 Datos e imágenes de difusión

Los clientes del sistema y fundadores del proyecto ODILA necesitan una sección en el sitio en la que se presente información sobre el mismo, y las distintas formas en la que pueden colaborar los usuarios para la difusión del mismo. Es esencial que el proyecto gane popularidad y se haga conocido para que pueda cumplir uno de sus objetivos principales, la elaboración de estadísticas sobre delitos informáticos y combatir la *cifra negra* respecto a los mismos.

En esta sección se incluyen distintas formas en las que un usuario puede colaborar: a través de *sponsorship*, adhesión institucional, y difusión en prensa. Se presentan imágenes en distintos tamaños y formatos para que los usuarios colaboren insertándolos en sus sitios web.

Figura 3.3.2.8. Datos e imágenes de difusión

### 3.3.2.9 Inicio de sesión para administradores

El sistema cuenta con una serie de herramientas de administración, que solo están disponibles para usuarios privilegiados. En primera instancia, los únicos con acceso de administración serán los clientes del sitio, aunque luego los mismos pueden crear cuentas de usuario adicionales en caso de necesitarlo.

El mecanismo de autenticación que se eligió es la clásica combinación de email y contraseña. Al crear un registro de usuario (ya sea programáticamente, o por un usuario administrador ya existente), se debe especificar obligatoriamente una dirección de correo electrónico, una contraseña de mínimo 8 caracteres, y una verificación de contraseña.

En la base de datos, la contraseña es almacenada de forma encriptada, utilizando para ello un algoritmo de encripción no reversible, específicamente el algoritmo *BCrypt*. Al intentar iniciar sesión, el sistema calcula el *hash* correspondiente a la contraseña que ingresó el usuario, y lo compara con el hash almacenado en la base de datos. De esta manera no se almacena la contraseña del usuario en texto plano.

La sesión en el navegador web es mantenida a través de la utilización de *cookies*, esto significa que el servidor puede almacenar una pequeña cantidad de datos en el lado cliente. Esta información de sesión viaja con en todas las peticiones HTTP del cliente al servidor automáticamente, lo que permite mantener una cantidad arbitraria de instancias de aplicación, ya que no se necesita un almacenamiento para sesiones. Para evitar que estos datos sean adulterados (y que por ejemplo, un usuario quiera identificarse con credenciales falsas), la aplicación servidora encripta estos valores utilizando el algoritmo simétrico *Base64,* mezclando los datos en cuestión (email y contraseña), con una clave secreta de 256 bytes, que debe ser almacenada como variable de entorno en cada instancia del servidor. De esta manera, cuando se recibe una petición, la aplicación desencripta el contenido de la *cookie* utilizando esta clave secreta.

La interfaz para iniciar sesión de administrador es la siguiente:

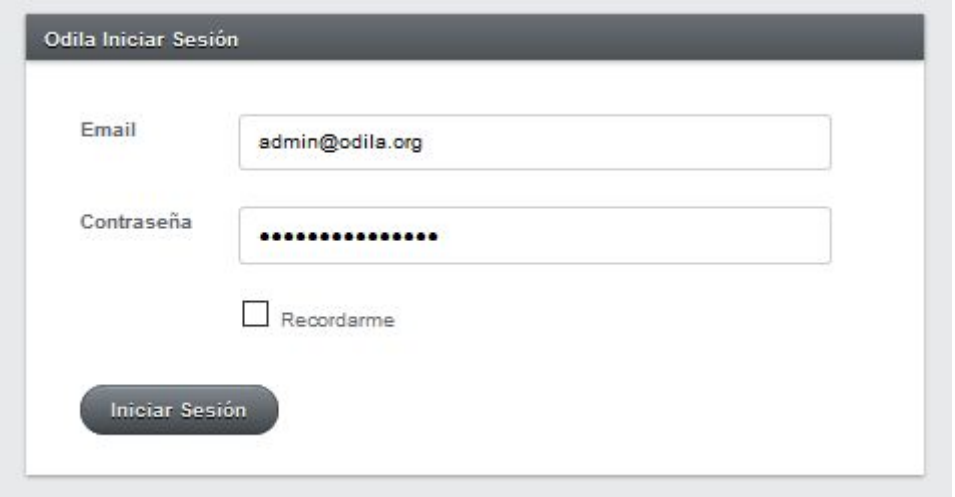



Figura 3.3.2.9. Inicio de sesión de administrador

### 3.3.2.10 Configuración de URL secreta para panel de administración

Los clientes del sistema, al ser expertos en el área de seguridad informática, decidieron agregar aún más medidas de seguridad al sistema de autenticación y panel de administración. Los mismos solicitaron que la ruta de acceso al panel administrativo y a la pantalla de inicio de sesión sea secreta, y configurable a través de una variable de entorno en el servidor.

De forma convencional, la ruta de acceso a este tipo de herramientas es la dirección de host seguida de "/admin" , pero se solicitó que una vez asignado un valor a dicha variable de entorno, por ejemplo "secret_key", el panel de inicio de sesión sea accesible a través de la URL "http://odila.herokuapp.com/secret_key".

Esta funcionalidad se implementó asignando un nombre a la variable de entorno utilizada, "ADMIN_ROUTE", y luego se configuró el archivo de rutas de Rails (*routes.rb)* para montar el motor de administración sobre la ruta especificada en dicha variable.

El procedimiento utilizado para especificar variables de entorno en el servidor de producción se explicó en la sección “Configuración de variables de entorno” en el inciso 3.3.1.

#### 3.3.2.11 Denuncias recientes y denuncias por país

Esta funcionalidad comprende un pequeño reporte con dos secciones principales: En la primera, un listado de las diez denuncias más recientes, de las cuales solo se debe mostrar el país de origen, el tipo de delito sufrido, y la fecha en la que se presentó; en la segunda debe mostrarse la estadística de cuántas denuncias fueron recibidas por país, desde que se puso en marcha el sistema.

Este reporte es presentado cada vez que se inicia sesión como administrador, es decir, es la pantalla de inicio de dicho panel.

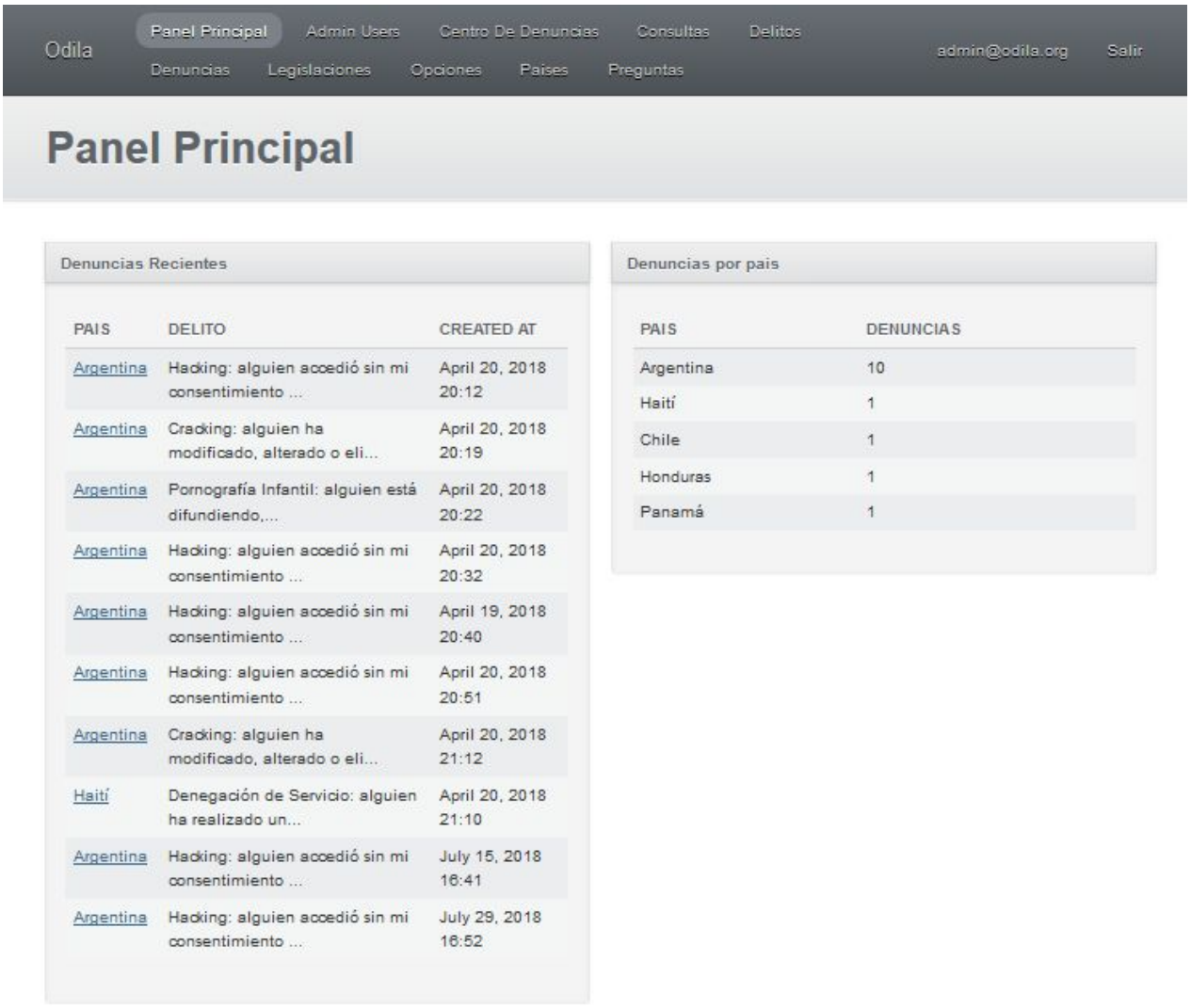



Figura 3.3.2.11. Reporte de denuncias recientes

### 3.3.2.12 ABM y filtrado de centros de denuncia

Esta funcionalidad es accesible a través del panel de administración, en la pestaña de Centros de Denuncia.

La siguiente imagen presenta el listado en forma de tabla de todos los centros de denuncia. En la misma se muestran las columnas País, Nombre y Dirección, así también

como las tres acciones posibles de realizar sobre un registro existente: Ver detalles, Editar y Eliminar.

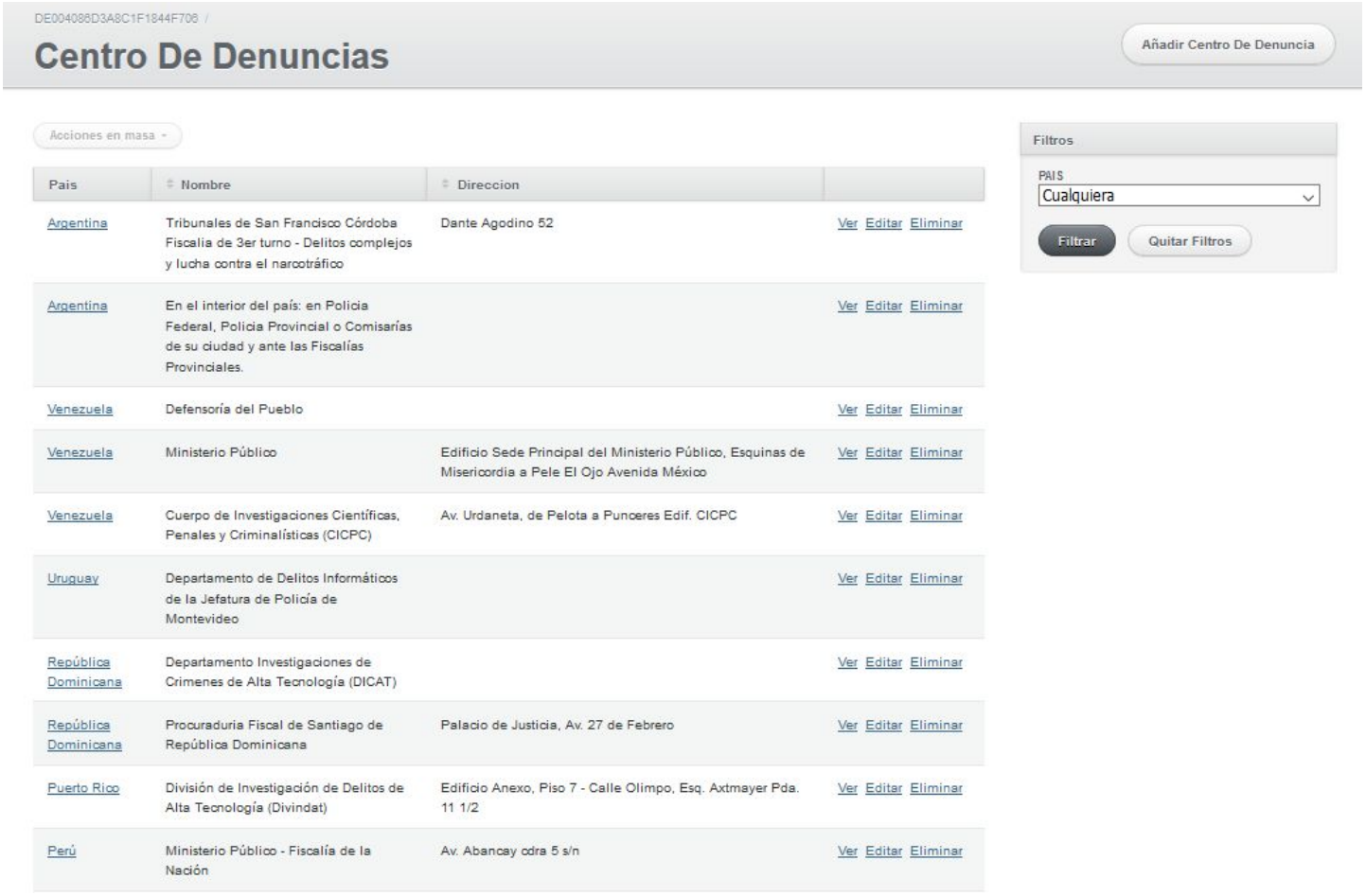



Figura 3.3.2.12a. Listado de centros de denuncia

Esta lista puede ser filtrada seleccionando de una lista el país del centro de denuncia asociado. A continuación se muestra una lista filtrada, seleccionando el país de Venezuela:

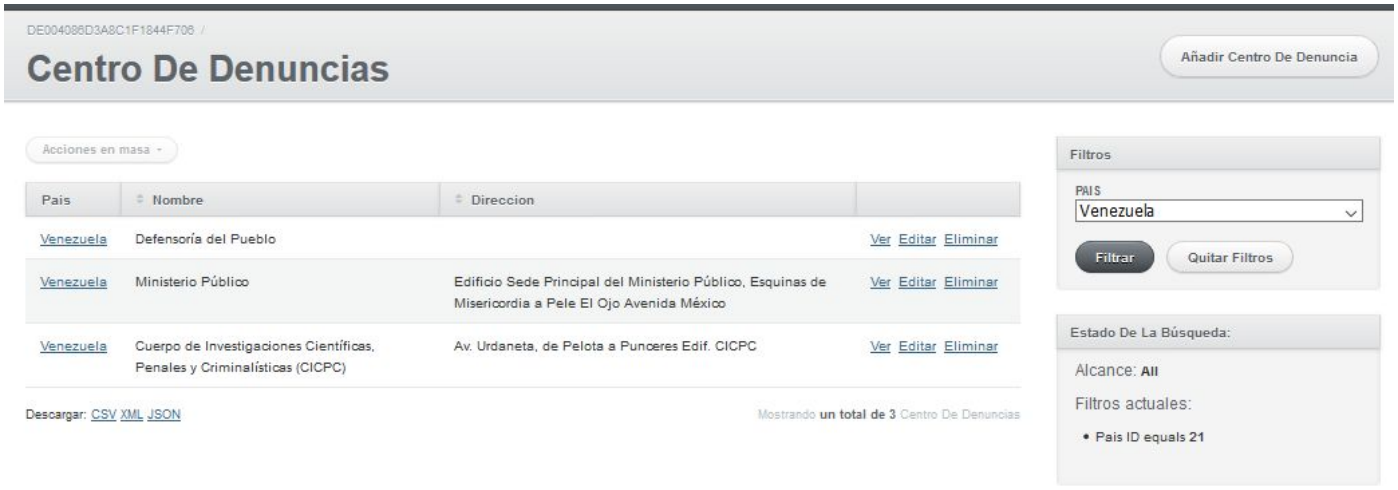



Figura 3.3.2.12b. Filtrado de centros de denuncia

Al hacer clic sobre el botón “Añadir Centro de Denuncia”, se puede acceder al siguiente formulario para dar el alta de un recurso. Este mismo formulario es el utilizado cuando el usuario elige “Editar” algún recurso ya existente, y encuentra los campos de texto con el contenido existente.

Todos los campos del mismo son obligatorios a excepción del sitio web y la cuenta de Twitter asociados a dicho centro. El campo “Orden” permite asociar a cada elemento un número por el cual será ordenado en los resultados de denuncia, y de este modo permitir dar prioridad a algunos registros por sobre otros.

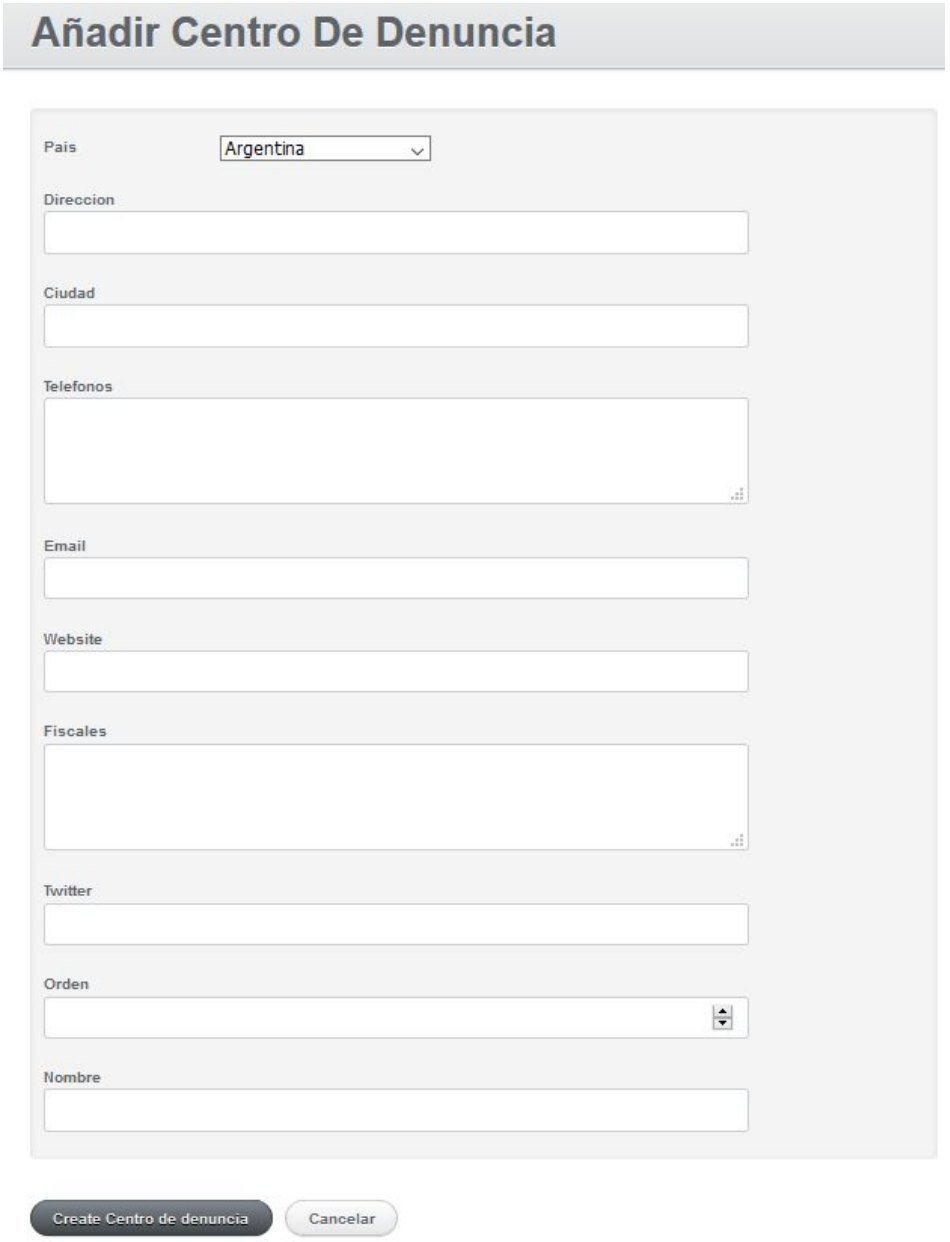



Figura 3.3.2.12c. Alta de centros de denuncia

### 3.3.2.13 ABM de delitos informáticos

Esta funcionalidad permite a los administradores del sitio agregar nuevos delitos a la base de datos. Esto permitirá luego asociar las legislaciones correspondientes, y aparecerá en el formulario de denuncia para que los usuarios puedan elegirlo entre las otras opciones.

Se muestra a continuación el listado de los registros de delitos existentes. Se omite el formulario de alta y edición, ya que es muy simple y sólo se puede ingresar el Nombre del delito en cuestión. Este recurso en particular no cuenta con funcionalidad de filtrado.

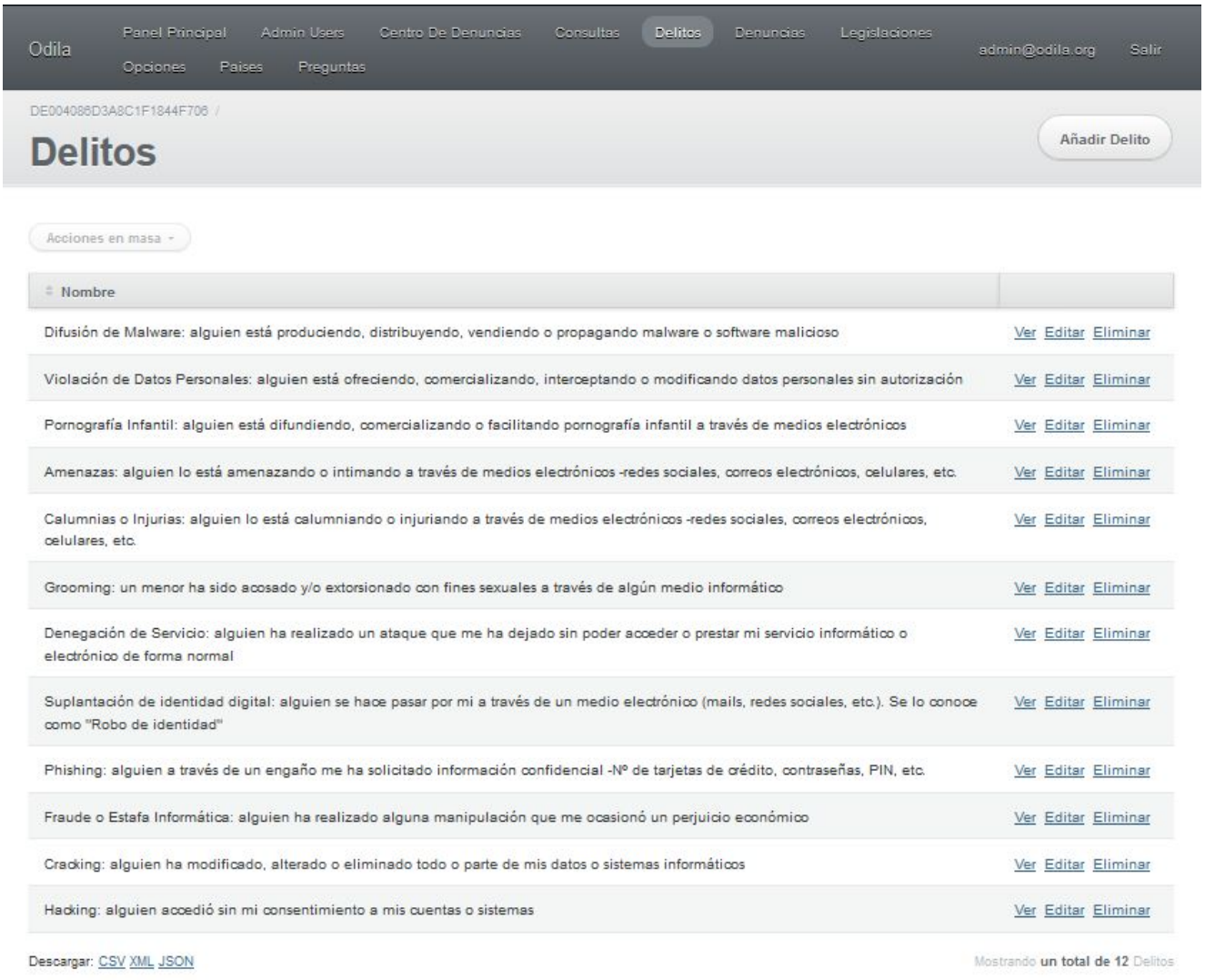


Figura 3.3.2.13. Listado de delitos informáticos

### 3.3.2.14 ABM y filtrado de denuncias realizadas

A continuación se presenta el listado de denuncias existentes, accesible a través de la pestaña “Denuncias” del panel de administración. El mismo muestra los atributos Id, País asociado, Delito asociado y Fecha de realización de las denuncias cargadas en el sistema.

Este listado muestra hasta veinte registros, y cuenta con funcionalidad de paginación. Los resultados son mostrados en orden descendente por fecha de realización (es decir, las últimas denuncias se muestran primeras en la lista).

Los datos mostrados para cada registro son el Id de la denuncia, el país de origen, el delito asociado y la fecha de realización.

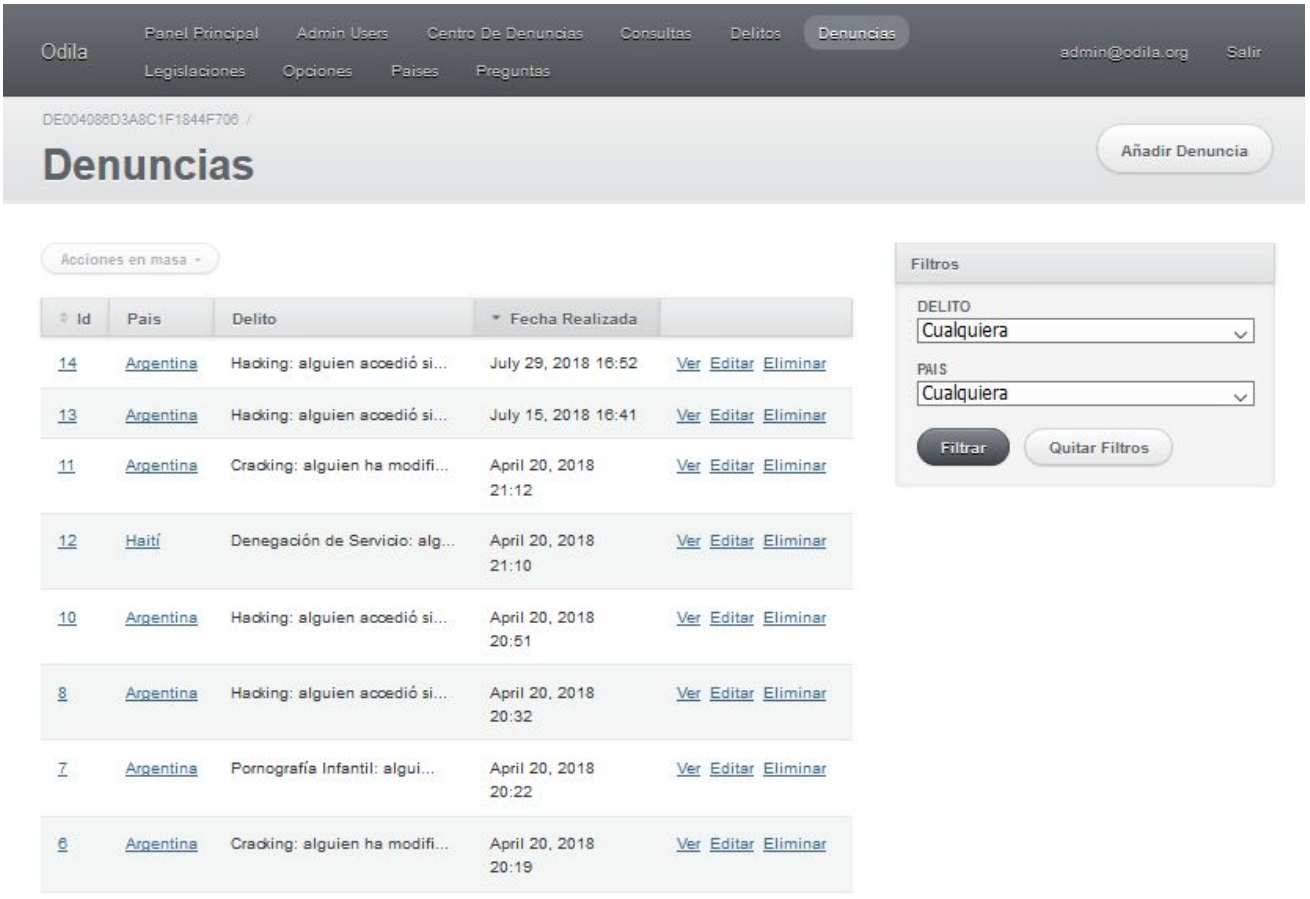


Figura 3.3.2.14a. Listado de denuncias

La siguiente funcionalidad asociada es la de filtrado, permitiendo a los administradores consultar las denuncias existentes asociadas a un delito o país en particular. Los filtros son independientes uno de otro, por lo tanto se puede seleccionar uno, los dos o ninguno. A continuación se muestra un listado filtrado para el delito de Cracking y el país de Argentina:

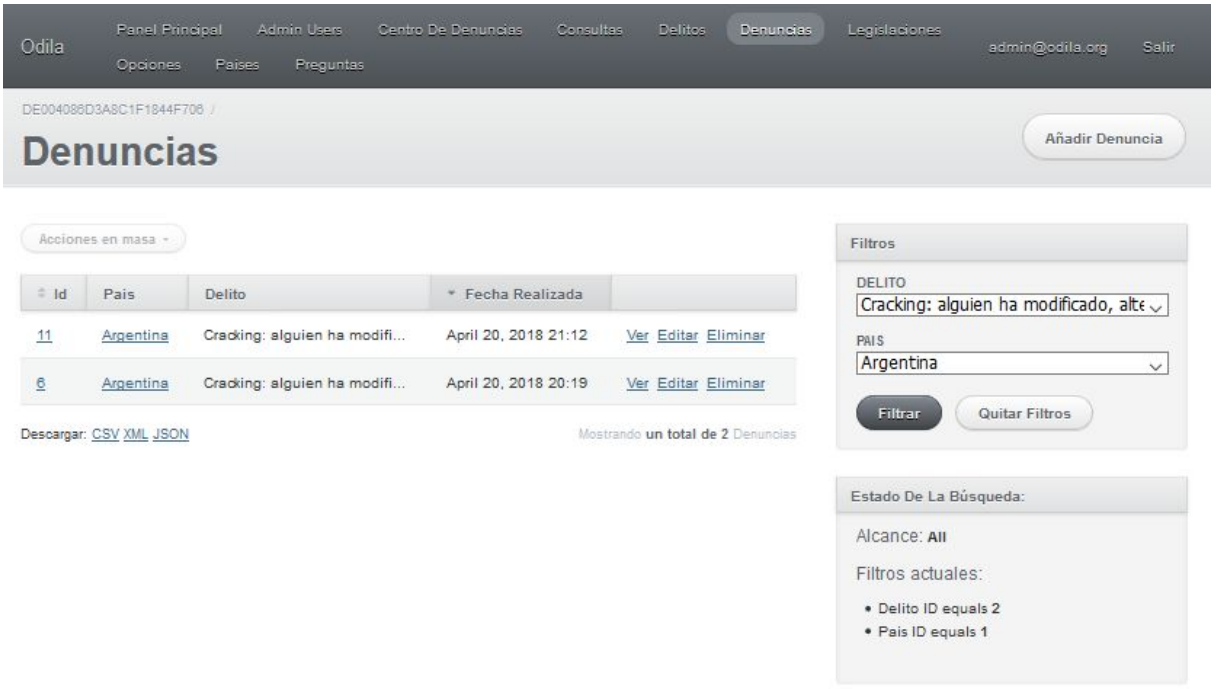



Figura 3.3.2.14b. Filtrado de denuncias

Si se desea ver las respuestas a las preguntas del formulario en cada denuncia, se puede hacer clic sobre la fila correspondiente, y esto nos llevará a la siguiente pantalla. Un dato incluido en esta sección que no figura en el listado es la dirección IP del denunciante:

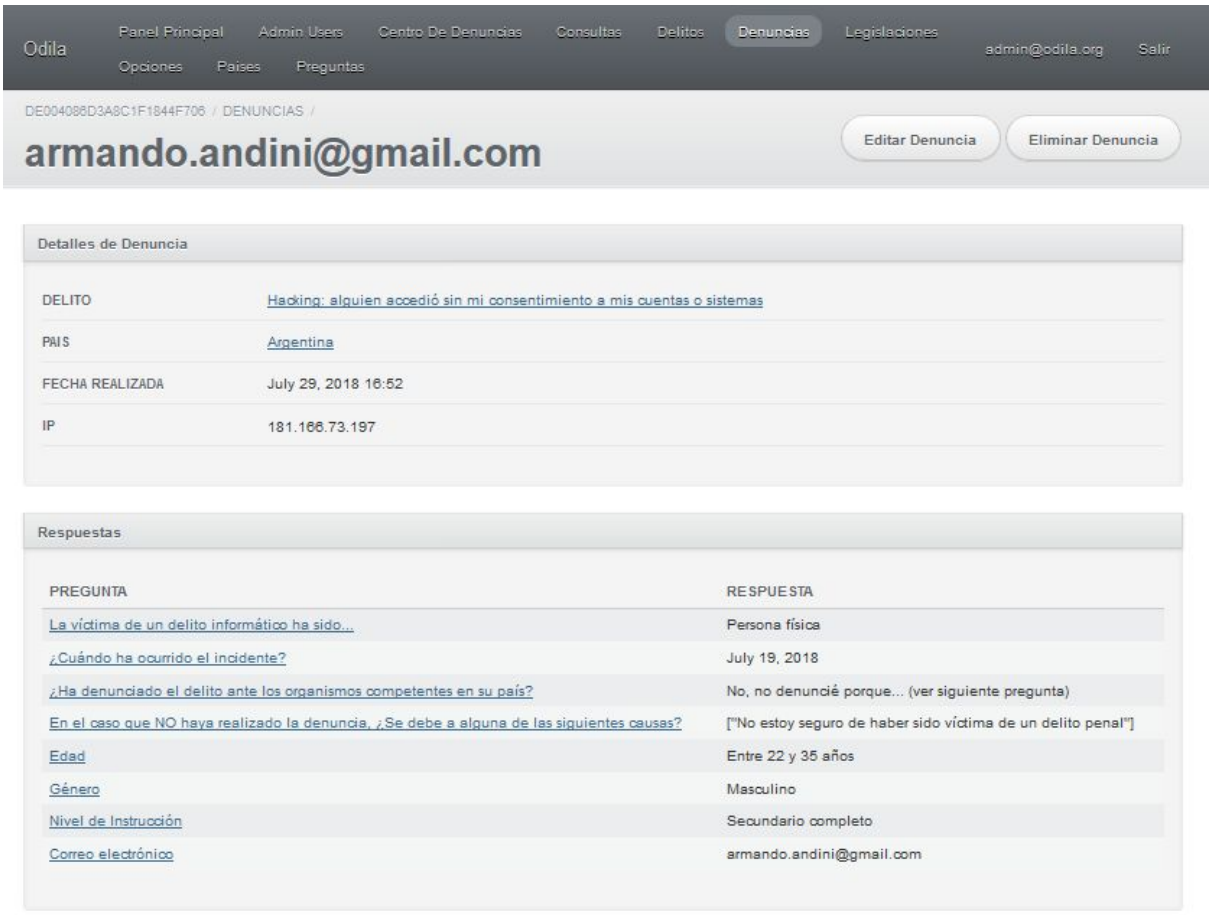



Figura 3.3.2.14c. Detalle de denuncia

### 3.3.2.15 ABM y filtrado de legislaciones por país

Esta sección es una de las herramientas más importantes del sitio, ya que permite la carga de legislaciones asociadas a cada delito y país en particular, según su tipificación legal.

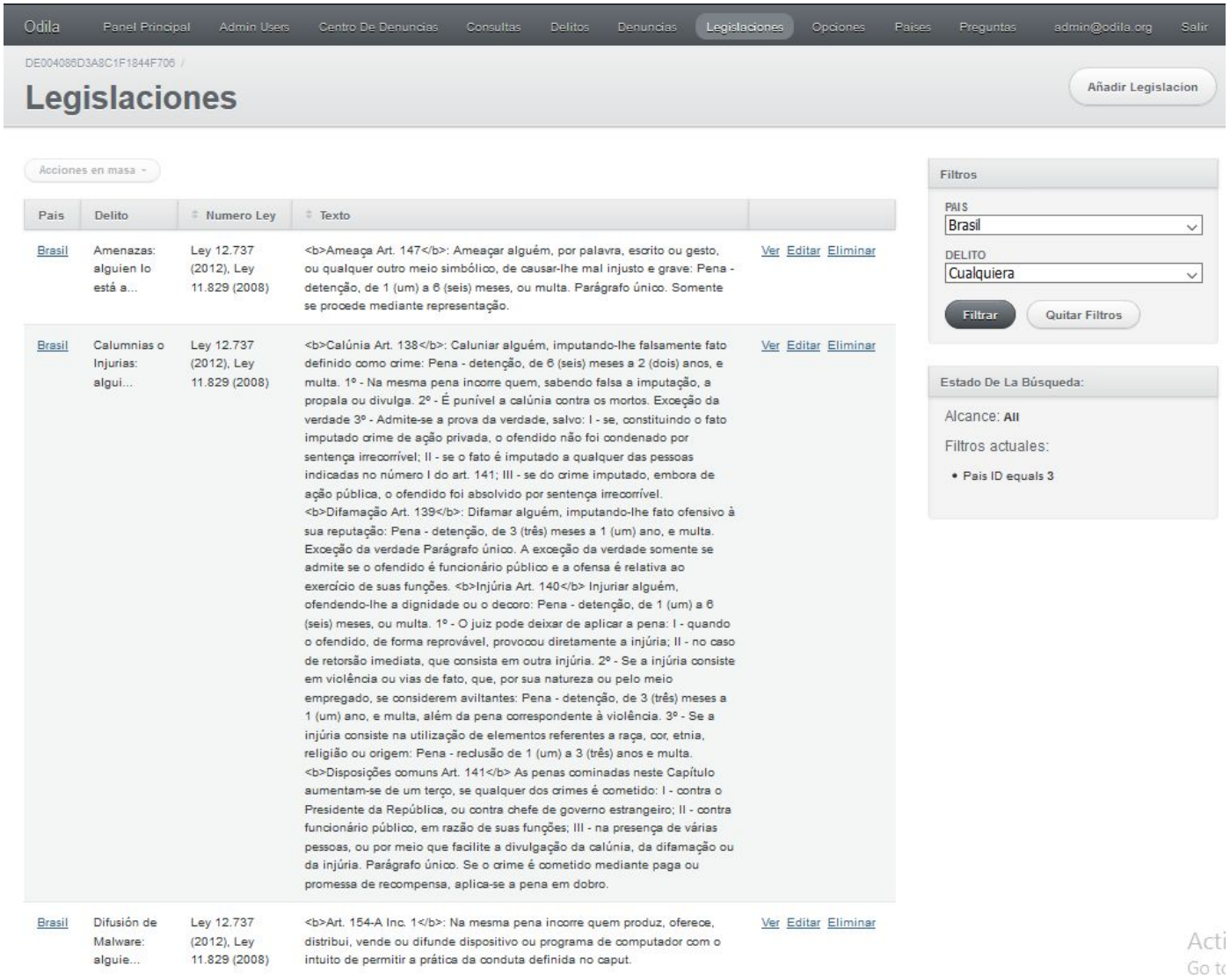



Figura 3.3.2.15a. Listado y filtrado de legislaciones

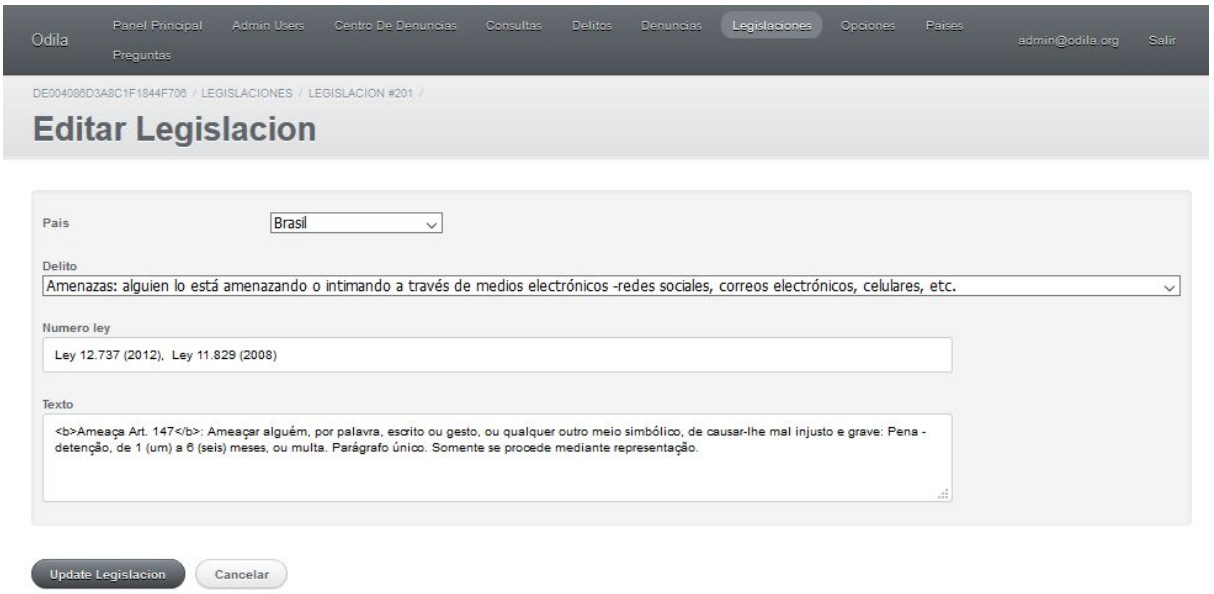



Figura 3.3.2.15b. Editar legislación

### 3.3.2.16 ABM de países

Esta funcionalidad permite el alta, baja, modificación y listado de países. Inicialmente el sistema cuenta con un total de 21 países soportados, cuya lista fue cargada a través de la importación de un archivo de datos. Para todos estos países se encuentran asociadas por lo menos una legislación. Estos países soportados inicialmente son:

- Argentina
- Bolivia
- Brasil
- Chile
- Colombia
- Costa Rica
- Cuba
- Ecuador
- El Salvador
- Guatemala

- Haití
- Honduras
- Mexico
- Nicaragua
- Panamá
- Paraguay
- Perú
- Puerto Rico
- República Dominicana
- Uruguay
- Venezuela

#### 3.3.2.17 Agregar preguntas al formulario de denuncia

Uno de los principales requerimientos para el sistema es el de poder extender el formulario de denuncia sin la necesidad de modificar el código fuente del mismo. Esto supone la existencia de una herramienta que permita agregar preguntas y respuestas asociadas a través de una herramienta de administración. Las mismas luego deberán ser presentadas en el formulario de denuncia para todos los usuarios que visiten el sitio luego de su creación.

El formulario está diseñado de tal manera que las primeras dos preguntas presentadas son predefinidas, es decir no se pueden modificar, y las mismas son: el país de residencia, y el delito del que fue víctima. Luego de estas dos preguntas iniciales, el resto son las de tipo dinámicas, almacenadas en la base de datos y manipulables por los administradores.

A continuación se presenta el formulario de creación de preguntas dinámicas:

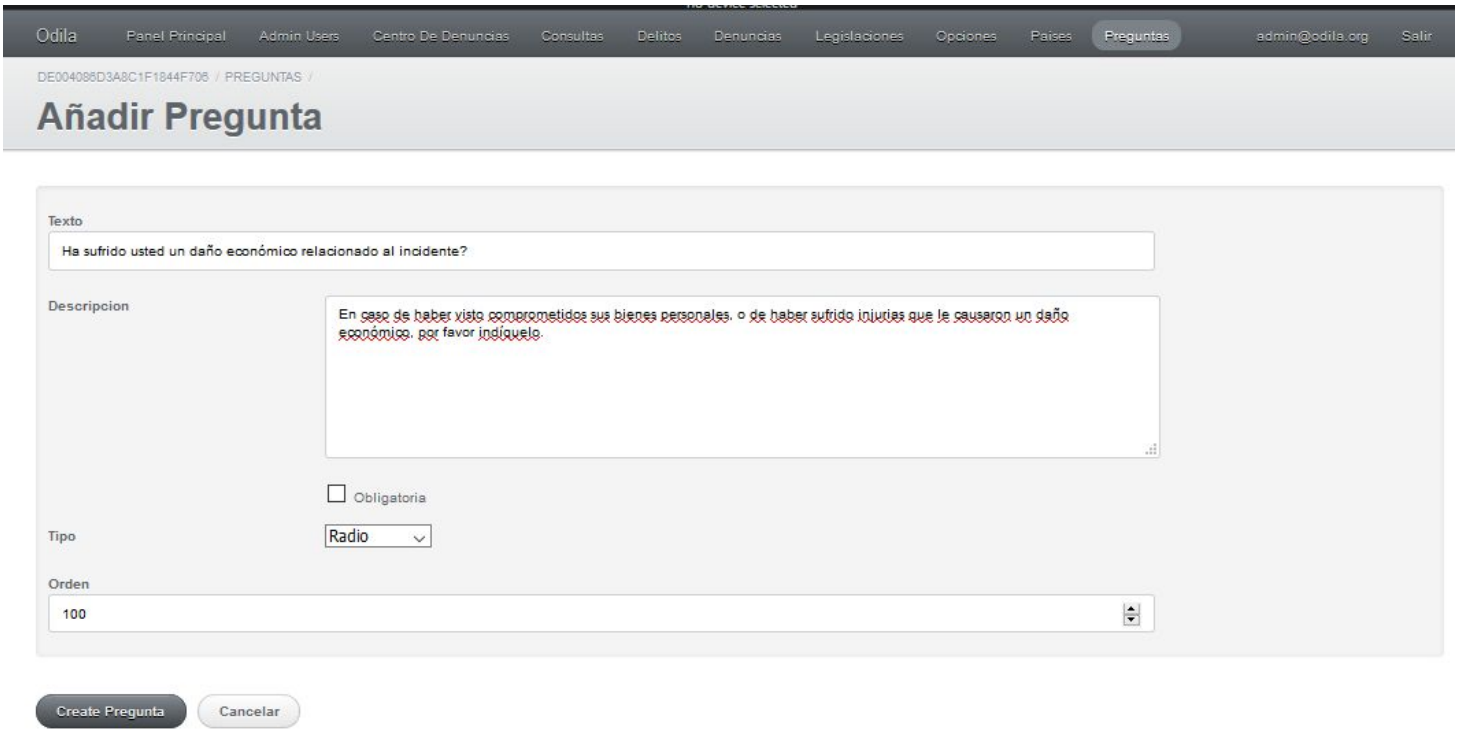



Figura 3.3.2.17. Agregar preguntas al formulario de denuncia

Se debe especificar para cada pregunta un campo texto, qué significa el cuerpo de la pregunta, y una descripción aclarativa para desarrollar más en detalle la misma. Además se puede tildar una pregunta como obligatoria o no. En el caso de serlo, el usuario denunciante no podrá saltear el paso correspondiente en el formulario.

El campo "Tipo" hace referencia a la naturaleza de las respuestas y opciones asociadas a la pregunta. Las posibilidades soportadas son:

- Radio: Las respuestas a esta pregunta son predefinidas y solo se puede elegir una de ellas. Serán presentadas con componentes de botones de radio.
- Checkbox: Las respuestas posibles son predefinidas, pero se pueden elegir cero o más de ellas. En caso de ser obligatoria, por lo menos se deberá tildar una casilla antes de poder continuar con la denuncia. Las opciones serán presentadas con casillas de selección.

- Fecha: La respuesta asociada a esta pregunta es de tipo fecha, y se presentará al usuario con un elemento de selección de tipo “Date Picker”, para facilitar la experiencia de usuario y evitar que tenga que escribir la fecha manualmente y en el formato apropiado.
- Email: La respuesta asociada deberá ser una dirección de correo electrónico. El campo presentado en el formulario tendrá asociadas validaciones para asegurar que sólo se puedan ingresar valores que correspondan a una dirección de e-mail de formato válido.
- SelectBox: La lista de respuestas posibles está predefinida y son presentadas a través de una lista de selección desplegable.

Finalmente se puede asignar un valor de “Orden” a la pregunta, que determinará el orden en que se presentarán las preguntas en el formulario. Las que tengan menor número de orden serán las primeras en ser presentadas al usuario.

#### 3.3.2.18 Asociar respuestas de selección simple

Esta funcionalidad consiste en asociar opciones posibles a preguntas de tipo “Radio” o “SelectBox”, dado que ambas son de selección simple. Se puede acceder a esta función a través del panel de administración, en la pestaña “Opciones”. Para ver las opciones asociadas a una pregunta, se puede aplicar un filtro como se muestra a continuación:


Figura 3.3.2.18a. Lista de opciones asociadas a pregunta

Al hacer click en “Añadir Opción”, se accede al formulario de alta de opción:

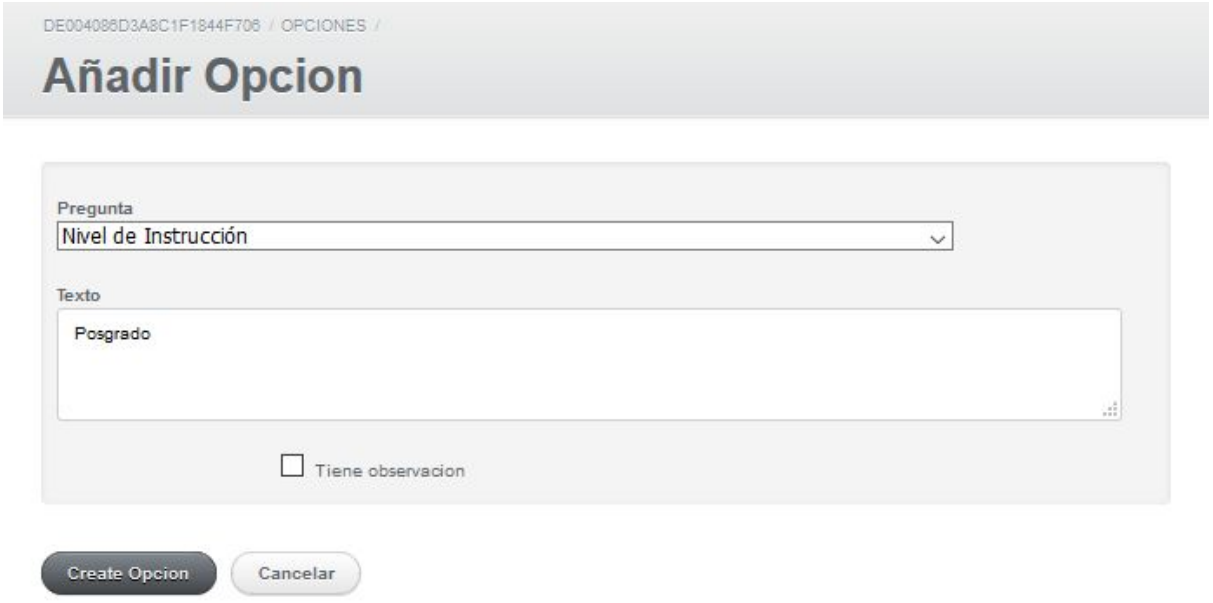


Figura 3.3.2.18b. Asociar nueva opción a pregunta existente

Los campos requeridos para la creación de una opción son: La pregunta asociada, el texto de la opción, y si se puede ingresar una observación en la respuesta del formulario. Esto permite que para ciertas opciones se pueda ingresar texto complementario a la respuesta.

Como se puede observar a continuación, como la opción creada fue asociada a una respuesta de tipo "Radio", dicha opción se presentará en el formulario a los nuevos denunciantes:

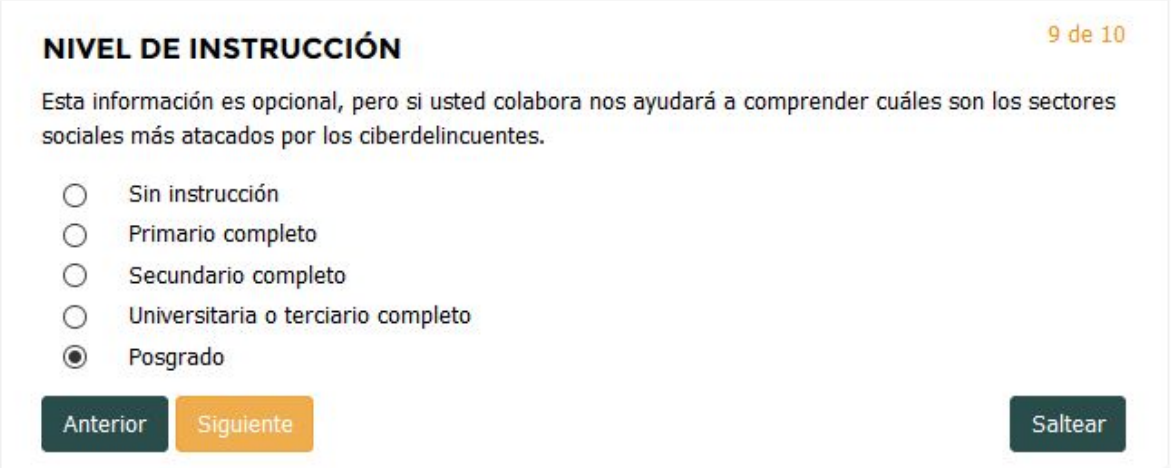



Figura 3.3.2.18c. Opción añadida al formulario

### 3.3.2.19 Asociar respuestas de selección múltiple

El procedimiento para asociar respuestas posibles a preguntas de selección múltiple (de tipo "CheckBox"), es el mismo que para añadir opciones a preguntas de selección simple. La diferencia está en la generación dinámica del formulario y en el almacenamiento de la misma en la base de datos.

Como ejemplo de esta funcionalidad, se añadió a la pregunta "*En el caso que NO haya realizado la denuncia, ¿Se debe a alguna de las siguientes causas?*" la opción

"*Considero que no vale la pena el esfuerzo necesario"*. A continuación se muestra el formulario actualizado:

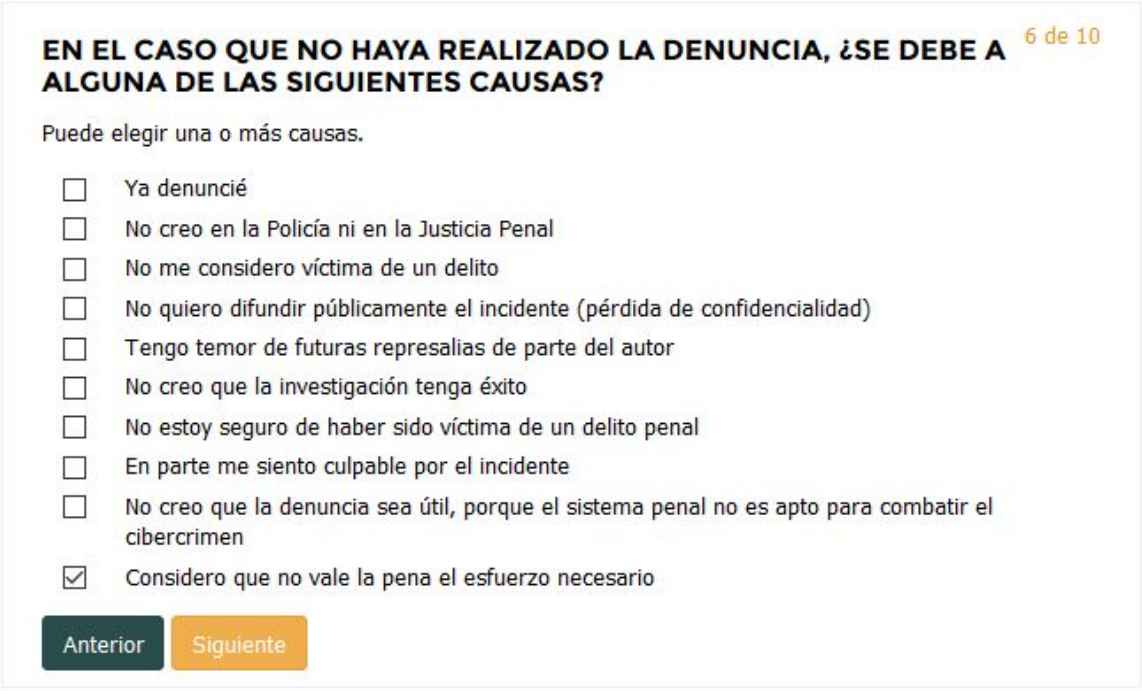



Figura 3.3.2.19. Opción múltiple añadida a formulario

### 3.3.2.20 Asociar respuestas de tipo fecha

Uno de los tipos de pregunta que debía soportar el sistema es el de tipo fecha. Este requerimiento implica que en el formulario se presenten preguntas a las cuales se pueda responder a través de cajas de selección.. De esta manera la experiencia de usuario es más amena dado que no debe ingresar la información en ningún formato predefinido (por ejemplo dd-mm-aa), sino que solo debe seleccionar con el mouse el día, mes y año correspondiente utilizando solamente el *mouse.*

Para crear una pregunta de tipo fecha, se debe indicar el tipo correspondiente en el alta de Pregunta. Se creará una pregunta de ejemplo, con el texto "Cuándo ha actualizado su

antivirus por última vez?". A continuación se muestra el resultado, con la pregunta presentada en el formulario y el elemento de selección de fecha:


Figura 3.3.2.20. Respuesta de tipo fecha agregada al formulario

### 3.3.3 Mejoras, facilidades y nuevas oportunidades

El sistema desarrollado presenta nuevas funcionalidades de gran valor para los clientes, dado que ahora tienen mucho más control sobre la herramienta de denuncia que diseñaron. A través del panel de administración pueden contar con acceso a todos los datos del sistema, así como estadísticas de denuncias recientes, y del total de denuncias agrupadas por país, tipo de delito, etcétera. Esto es una gran ventaja dado que antes el modo de obtener esta información era solicitando una descarga de datos a los desarrolladores del sitio, proporcionado en un archivo excel.

Otra gran utilidad que proporciona el nuevo sistema es la creación de preguntas y respuestas dinámicas, permitiendo la extensión y modificación de los formularios de denuncia según sea necesario. De esta manera, los administradores del sitio pueden alterar la funcionalidad principal del sitio sin necesidad de modificar el código fuente del sistema, ni tener conocimiento de programación.

Una de las oportunidades de mejora identificada durante el proceso de desarrollo fue la de introducir corresponsales o administradores locales por país. Esto implicaría la asignación de roles de administrador a nuevos usuarios, pero con permisos limitados a la administración de recursos asociados a un país en particular. Esto incluiría legislaciones,

centros de denuncia, denuncias existentes, etc. De esta manera, los fundadores del proyecto podrían delegar la expansión del proyecto en otros países asignando colaboradores, y así lograr ampliar la cobertura del mismo.

Otra sugerencia que surgió fue la de permitir modificar partes del contenido estático del sitio a través de alguna herramienta interactiva, puesta a disposición de los usuarios administradores. Esto brindaría aún más flexibilidad al uso de la misma, dado que tampoco se requeriría modificar el código fuente del sistema para modificar secciones como la portada, información de contacto, información del proyecto, datos de difusión, glosario, etc.

Por último, se observó que la visualización de legislaciones asignadas al par país-delito puede mejorarse dado que el listado a través de una tabla no es la forma más práctica para determinar qué combinaciones tienen asignada una legislación y a cuáles les falta. Los clientes propusieron una vista en forma de grilla, en la que un eje corresponda a los delitos y el otro a los países, y se observe en cada celda si dicha combinación está tipificada o no, y que sea fácilmente accesible a la pantalla de alta a través de un click en dicha grilla.

# 4. Metodología

Para abordar este proyecto se ha optado por utilizar la metodología ágil Kanban. Se debe destacar que en realidad no se llevó a cabo en un 100% dicha metodología, sino que fue tomada como base para generar una adaptación de la misma, tomando las prácticas y herramientas que mejor se adapten a los requerimientos de este proyecto.

La elección de una metodología ágil se dio por varios motivos. Uno de ellos es por la incertidumbre siempre presente en los requerimientos, dado que aunque los mismos estuvieran parcialmente definidos de antemano, los detalles iban cambiando constantemente y nuevas ideas iban surgiendo. Otro motivo fue la fuerte participación de los clientes en el proyecto, con quienes se programaban entregas frecuentes e incrementales del sistema desarrollado.

## 4.1. Kanban

Actualmente, el término Kanban ha pasado a formar parte de las metodologías ágiles, cuyo objetivo es gestionar de manera general cómo se van completando las tareas. Kanban es una palabra japonesa que significa “tarjetas visuales”, donde Kan es “visual”, y Ban corresponde a “tarjeta”.

Las principales ventajas de esta metodología es que es muy fácil de utilizar, actualizar y asumir por parte del equipo. Además, destaca por ser una técnica de gestión de las tareas muy visual, que permite ver a golpe de vista el estado de los proyectos, así como también pautar el desarrollo del trabajo de manera efectiva.

La metodología Kanban se basa en los siguientes principios:

- Calidad garantizada: Las tareas realizadas deben cumplir su objetivo y cumplir las expectativas de calidad al primer intento. En Kanban no se prioriza la rapidez, sino la calidad final de las tareas realizadas. Esto se basa en el hecho

que muchas veces cuesta más arreglar un defecto luego de realizado, que hacerlo bien en el primer intento.

- Reducción del desperdicio: Kanban se basa en hacer solamente lo justo y necesario, pero hacerlo bien. Esto supone la reducción de todo aquello que es superficial o secundario. Este principio se alinea con la buena práctica del desarrollo de software de escribir el menor código necesario para cumplir con el objetivo del sistema, evitando el diseño excesivo y la previsión innecesaria de posibles requerimientos futuros.
- Mejora continua. Kanban no es simplemente un método de gestión de proyectos, sino también un sistema de mejora en el desarrollo de proyectos, según los objetivos a alcanzar. En cualquier momento durante el desarrollo se pueden agregar tareas que mejoren o modifiquen las salidas de las anteriores.
- Flexibilidad. El orden de las tareas a realizar se decide partiendo del *backlog* (o tareas pendientes acumuladas), pudiéndose priorizar aquellas tareas entrantes según las necesidades del momento . De esta manera se cuenta con la capacidad de dar respuesta a tareas imprevistas.

### 4.1.1. Roles

Hay que tener en cuenta que Kanban es un método evolutivo que introduce cambios en la organización. Eso significa que no hay roles o prácticas adicionales a introducir en las organizaciones que adopten este método. Los roles y funciones existentes se mantienen en Kanban ya que los flujos de trabajo se investigan y se visualizan para proporcionar control de todo el trabajo, pero que no cambian la forma en que la gente hace su trabajo o cómo interactúan con él. Dicho esto, se pueden identificar dos roles básicos en cualquier proyecto Kanban:

- *Product owner* o *manager:* Se encarga de gestionar la demanda y los requisitos dentro del sistema Kanban, manejando las relaciones con los *stakeholders* y fomentando la transparencia dentro del sistema en torno a la

priorización del trabajo. Entre sus actividades se encuentran la adición de nuevas tareas al backlog, el refinamiento de los requerimientos proporcionados por los clientes, y la priorización de tareas.

- Equipo de desarrollo: Son los encargados de llevar a cabo las tareas del backlog. Este rol incluye tareas de análisis, diseño, desarrollo, pruebas, investigación, entre otras. Estas tareas pueden ser realizadas por la misma persona o distintas, según sus especialidades. A través del trabajo realizado por el equipo, las tareas van avanzando entre las etapas (columnas en Kanban), hasta que llegan a la última etapa de producción, en la que se genera un incremento de valor al producto que se está desarrollando.

### 4.1.2. Proceso

La aplicación del método Kanban implica la generación de un tablero de tareas que permitirá mejorar el flujo de trabajo y alcanzar un ritmo sostenible. Para implantar esta metodología, se deben tener claro los siguientes aspectos:

**Definir el flujo de trabajo de los proyectos**

Para ello, simplemente se debe crear nuestro propio tablero, que deberá ser visible y accesible por parte de todos los miembros del equipo. Cada una de las columnas corresponderá a un estado concreto del flujo de tareas, que nos servirá para saber en qué situación se encuentra cada proyecto. El tablero debe tener tantas columnas como estados por los que pasa una tarea, desde que se inicia hasta que finaliza (por ejemplo lista, en progreso, en revisión, en pruebas, en producción, etc.).

Dicho tablero puede ser específico para un proyecto en concreto o bien genérico. No hay unas fases del ciclo de producción establecidas sino que se definirán según el caso en cuestión, o se establecerá un modelo aplicable genéricamente para cualquier proyecto de la organización.

**Visualizar las fases del ciclo de producción**

Kanban se basa en el principio de desarrollo incremental, dividiendo el trabajo en distintas partes. Esto significa que no hablamos de la tarea en sí, sino que lo dividimos en distintos pasos para agilizar el proceso de producción.

Normalmente cada una de esas partes se escribe en una tarjeta y se agrega al tablero, en la fase que corresponda. Dichas tarjetas contienen la información básica para que el equipo sepa rápidamente la carga total de trabajo que supone: normalmente descripción de la tarea con la estimación de horas. Además, se pueden emplear fotos para asignar responsables así como también usar etiquetas con distintas formas para poner observaciones o indicar bloqueos (cuando una tarea no puede hacerse porque depende de otra).

Al final, el objetivo de la visualización es clarificar al máximo el trabajo a realizar, las tareas asignadas a cada equipo de trabajo, así como también las prioridades y la meta asignada.

**Terminar en vez de comenzar**

Este es el lema principal de la metodología Kanban. De esta manera, se prioriza el trabajo que está en curso en vez de empezar nuevas tareas. Precisamente, uno de los principales aportes de Kanban es que el trabajo en curso debe estar limitado, y existe un número máximo de tareas a realizar en cada fase.

En realidad, se trata de definir el máximo número de tareas que podemos tener en cada una de las fases (por ejemplo tres tareas en la fase de planificación, dos en la fase de desarrollo, una en la fase de pruebas, etc.) y así restringir el trabajo en curso. A esto, se le añade otra idea que, por muy obvia que pueda parecer, la práctica nos demuestra que no es así: no se puede abrir una nueva tarea sin finalizar otra.

De esta manera, se pretende dar respuesta al problema habitual de muchos equipos de tener muchas tareas abiertas pero con una baja tasa de finalización. En Kanban lo ideal es que las tareas que se abran se cierren antes de empezar con la siguiente.

**Control del Flujo**

Con Kanban se trata de mantener a los trabajadores con un flujo de trabajo constante, las tareas más importantes en cola para ser desarrolladas y un seguimiento pasivo para no tener que interrumpir al trabajador en cada momento.

Asimismo, dicha metodología de trabajo nos permite hacer un seguimiento del trabajo realizado, almacenando la información que nos proporcionan las tarjetas. En base a esto se puede determinar la velocidad del proyecto, el cumplimiento de estimaciones, la productividad de cada integrante, entre otras métricas.

#### 4.1.2.1. Aspectos de KanBan no utilizados

La metodología aplicada se diferencia de la definición estricta de KanBan ya que si bien se realizan las prácticas principales del proceso, algunas fueron dejadas de lado por ser consideradas innecesarias para este proyecto.

Una de las prácticas que fue omitida es la de definir pólizas explícitas para el nivel de servicio y fechas límites. Se optó por un enfoque más informal en el que las fechas de entrega y los resultados esperados de cada una fueron pactadas entre ambas partes. No se utilizaron documentos explícitos de validez legal para pactar cada una de las entregas del producto.

Otra práctica de KanBan que no se llevó a cabo por no ser necesaria en este proyecto es la de mejora contínua del proceso. La metodología propone como uno de sus puntos fuertes la evaluación, análisis y redefinición constantes de cada uno de los procesos de desarrollo del producto. En este proyecto el proceso fue definido al principio del mismo y se adoptó un enfoque simple, tanto para los desarrolladores como para clientes, por lo que no se vio necesario invertir tiempo en mejorar dicho proceso.

## 4.2. Aplicación de Kanban

### 4.2.1. Roles

En este proyecto el rol de *product owner* fue llevado a cabo por los clientes del sistema y creadores del proyecto ODILA. En particular, la persona más involucrada en la definición de requerimientos y seguimiento del flujo de trabajo fue Cristian Borgharello, director de Segu-Info.

El rol de equipo de desarrollo fue llevado a cabo por el alumno que redacta el informe, dado que este proyecto es de carácter individual.

### 4.2.2. Recopilación de requerimientos, análisis y diseño

Al comenzar el proyecto tuvimos varias reuniones con los fundadores del proyecto ODILA, donde los mismos explicaban cuáles eran los problemas que tenían con su sistema actual y cuáles eran las expectativas de desarrollar un sistema de información nuevo que extienda las funcionalidades del anterior. En los primeros encuentros fueron especificando a grandes rasgos las características con las que debía contar el sistema, lo que permitió realizar un bosquejo inicial de las dimensiones y el alcance del proyecto.

Luego de conocer la estructura y los componentes de la empresa, se ideó la arquitectura con la que contaría el sistema. A medida que fueron avanzando las reuniones se fueron detallando más los requerimientos, sin embargo, siempre se tuvo presente que estos podrían llegar a ser modificados. Una vez que se contara con el conocimiento suficiente sobre los mismos, se tradujeron en historias de usuarios, las cuales fueron priorizadas y estimadas para finalmente formar el *product backlog*.

La priorización de historias de usuarios estuvo a cargo del cliente, donde se calificaron con tres rangos de prioridad: baja, media y alta. Si bien hubo una priorización inicial, las necesidades del cliente fueron variando, por lo que en etapas posteriores se debieron realizar nuevas priorizaciones según se requirió.

La estimación del tiempo de las historias fue realizada en horas de ingeniería, basadas en experiencia previa trabajando en proyectos de desarrollo web. Se define una hora de ingenieria como una hora en la que no sufrimos interrupciones, no encontramos problemas inesperados, y no realizamos otras actividades más que dedicarnos a la actividad del desarrollo de software en sí.

### 4.2.3. Desarrollo

Una vez conformado el *product backlog,* se comenzó con el desarrollo del sistema. La metodología elegida consiste en la especificación de tareas en forma de tarjetas y ubicarlas en una de cuatro columnas. Para la gestión del proyecto de desarrollo de software se utilizó una herramienta web llamada Trello. Esta herramienta da soporte a varias metodologías de desarrollo de software, pero especialmente aquellas basadas en Kanban.

Al comienzo del proyecto todas las tareas se encontraban en la columna “Backlog”, ordenadas por prioridad de mayor a menor. Además de la prioridad, otro factor determinante en el ordenamiento de las mismas, es la interdependencia entre sí. Esto significa que algunas tareas dependen de que otra sea realizada previamente, aún si la tarea previa es de menor prioridad que la siguiente.

El primer paso para comenzar a trabajar en una tarea es comprender en su totalidad la descripción de la misma, y lo que se espera de ella una vez que sea realizada. Para esto, se disponía de canales de comunicación contínua con los clientes del sistema, ya sea vía e-mail, teléfono, Skype o WhatsApp. En caso de que una tarea no esté clara, se solicitaba una explicación o refinamiento de la misma.

El siguiente paso es pasar la tarjeta de la columna “Backlog” a la columna “En progreso”. Esto significa que la tarea en cuestión está siendo desarrollada por el equipo. Se entiende por “desarrollar” la actividad que lleva una tarea a su concreción. En la mayoría de los casos esto significaba escribir código fuente que consistía en funcionalidades para el

sistema, pero otras tareas implicaban configurar alguna herramienta, realizar una pequeña investigación, convertir datos, etc.

El tiempo que se dedicaba a cada una de las tareas debía ser similar al estimado, aunque este no era siempre el caso. En algunas ocasiones sucedía que una tarea llevaba más horas de trabajo que las esperadas, por ejemplo por encontrarse alguna complicación inesperada en su implementación. A su vez, algunas tareas eran realizadas más rápido que lo estimado, por lo que compensaban a las anteriores. Periódicamente, a través del intercambio de e-mails, se informaba a los clientes de un posible retraso o adelanto en las entregas de las nuevas funcionalidades.

Luego de que se realizaran las actividades necesarias para la concreción de una tarea en particular, la misma se pasaba a la siguiente columna “En revisión”. Esta estado implica que el cliente debe validar la completitud y correcta implementación o finalización de la misma, en el caso de que la tarea implique la participación de los mismos. El procedimiento para realizarlo es desplegar la nueva funcionalidad a un servidor de pruebas, en la que los clientes pueden interactuar con el sistema libremente, sin preocuparse por daños o errores, ya que la base de datos y los servicios externos conectados al sistema son descartables y no almacenan información relevante en la vida real (conocidos como Sandbox).

En el caso de que los clientes observaran una discrepancia entre el comportamiento observado y el esperado del sistema, los mismos realizaban un comentario en la tarjeta correspondiente, y la misma volvía al estado “En progreso”. Luego se procedía a implementar los cambios necesarios basados en la retroalimentación provista, y se volvía a pasar la tarjeta a la etapa “En revisión”. Este ciclo se repetía hasta que la tarea era aprobada y se pasaba a la columna “Lista”.

A lo largo del proceso de desarrollo, principalmente en la etapa de revisión, iban surgiendo nuevos requerimientos o cambios en los mismos, que a veces ameritaban la creación de una nueva tarea. En el caso de que la misma estuviese dentro del alcance del proyecto, se estimaba, se le asignaba una prioridad y se colocaba en la columna “Backlog”. De la misma manera, algunas tareas eran modificadas, y algunas eliminadas, dado que los

clientes se daban cuenta de que no eran necesarias. Estos sucesos son muy comunes en el desarrollo de cualquier proyecto de software, por lo que es importante poder reaccionar a los mismos, ajustar las estimaciones y mantener el foco en el alcance de cada entrega. De lo contrario se puede terminar en un estado en el que muchas tareas están inconclusas, que se deseche trabajo realizado, o que el proyecto no avance por un cambio constante en el alcance de cada etapa.

### 4.2.4. Seguimiento del progreso

En la planificación del proyecto se definieron las historias de usuario, se estimaron las mismas y se les asignó una prioridad. En base a esto, en trabajo conjunto con los clientes, se organizaron entregas parciales. Las mismas consisten en un conjunto de funcionalidades que agregan valor al sistema, y forman parte de un mismo objetivo que amerita que estas se desplieguen en conjunto.

Dichas entregas poseen un valor estimado de horas, correspondiente a la suma de las estimaciones de las historias que las componen. Este valor, asociado con la dedicación diaria de horas definidas previamente, nos permite calcular el tiempo necesario para cada entrega.

A continuación se presentan las tres entregas planificadas, con las historias de usuario que componen cada una, y las estimaciones en horas reales de las mismas. En la planificación original del proyecto se estimó una dedicación semanal de al menos 30 horas semanales, pero llegado el momento de la implementación, se contaba con una disponibilidad de alrededor de 20 horas semanales debido a una relación laboral de tiempo completo por parte del alumno.

Cabe mencionar además que en cada entrega, a partir de la segunda, se agregarían las correcciones y tareas pendientes que quedaron de las entregas anteriores, en caso de haber alguna, por lo que esta carga adicional no fue estimada inicialmente.

| Primer entrega |
| --- |

| Historia | Estimación (hs) |
|---|---|
| H1 - Formulario de denuncia | 16 |
| H2 - Glosario | 8 |
| H3 - Portada | 12 |
| H4 - Formulario de contacto | 12 |
| H5 - Descarga de reportes | 12 |
| H6 - Email de respuesta | 24 |
| H7 - Formulario por pasos | 16 |
| H8 - Datos de difusión | 8 |
| **Total** | 108 horas |
| | 5 a 6 semanas |

Tabla 4.2.4a. Planificación de primer entrega

| Segunda entrega | |
|---|---|
| Historia | Estimación (hs) |
| H9 - Login de administrador | 16 |
| H10 - URL de administración secreta | 8 |
| H11 - Denuncias recientes y por país | 12 |
| H12 - ABM y filtrado de centros de denuncia | 16 |
| H13 - ABM de delitos informáticos | 16 |
| H14 - ABM y filtrado de denuncias realizadas | 12 |
| H15 - ABM y filtrado de legislaciones | 16 |
| H16 - ABM de países | 8 |
| **Total** | 104 horas |

| | |
|---|---|
| | 5 a 6 semanas |

Tabla 4.2.4b. Planificación de segunda entrega

| Entrega final | |
|---|---|
| Historia | Estimación (hs) |
| H17 - Añadir nuevas preguntas | 12 |
| H18 - Asociar opciones simples | 12 |
| H19 - Asociar opciones múltiples | 16 |
| H20 - Respuestas en texto plano | 12 |
| H21 - Respuestas de tipo fecha | 12 |
| H22 - Email de contacto | 12 |
| H23 - OWASP Top 10 | 16 |
| H24 - Carga por lotes de legislaciones | 12 |
| H25 - Carga por lotes de centros de denuncia | 12 |
| **Total** | 116 horas |
| | 6 semanas |

Tabla 4.2.4c. Planificación de tercer entrega

### 4.2.5. Estrategias de test

A continuación se explican los distintos tipos de test y su aplicación en este sistema.

**Test unitario**

Los tests unitarios son una forma de probar el correcto funcionamiento de un módulo de código fuente. Esto sirve para asegurar que cada uno de los módulos funcione correctamente por separado. El objetivo es aislar cada parte del programa y mostrar que cada

una de las partes individuales realizan el trabajo correctamente. Para esto, se suministra a cada módulo una entrada predefinida, y un conjunto de salidas esperadas luego del procesamiento por parte del módulo. Los tests pasan cuando las salidas reales concuerdan con las esperadas.

En el desarrollo del sistema se escribieron tests unitarios para algunos componentes de la capa de modelos, ya que en ellos reside la principal lógica de negocio. Para esto se utilizó el framework de testing RSpec.

**Test de integración**

Los tests de integración son un tipo de tests de software en el que los módulos de software individuales se combinan y se prueban como un conjunto. Se producen después de los tests unitarios. Los tests de integración tienen como entrada los módulos que han sido sometidos a tests unitarios, aplica pruebas definidas en un plan de test de integración, y ofrece como salida el sistema integrado listo para la prueba del sistema.

Para este sistema se han escrito tests de integración también con la suite de tests RSpec. La principal diferencia con un test unitario, es que el procesamiento sometido a pruebas no involucra a un solo módulo (clase, método, función), sino a un conjunto de ellos. Para que la salida del subsistema concuerde con la salida esperada, todos los módulos deben funcionar correctamente y sus interacciones deben estar correctamente definidas.

**Test de sistemas**

Los tests de sistema se llevan a cabo en un sistema completo e integrado para evaluar que el sistema cumpla con los requisitos especificados. Los tests de sistema entran en el ámbito de las pruebas de caja negra y, como tal, no requieren ningún conocimiento del diseño interno del código o la lógica.

Por regla general, los tests de sistema se ejecutan sobre todos los componentes que constituyen el sistema. Para esto es necesario que los componentes estén correctamente integrados. A su vez, para garantizar esto, es fundamental que los tests de integración hayan sido ejecutados y no hayan detectado errores. Además, para la ejecución de los tests de sistema, es necesario que el entorno tenga una adecuada configuración de hardware y se encuentren operativa toda la infraestructura que el sistema requiere (por ejemplo, otros sistemas con los cuales el sistema interactúa, etc.). Con los tests de sistema se busca detectar defectos tanto dentro de los módulos como también dentro del sistema en su conjunto.

Los test de sistema se realizaron de forma manual en nuestro sistema. Intentamos simular un proceso completo de todas las actividades posibles del vendedor. En un principio iniciando un recorrido en la semana, cargando notas de pedidos, cobros y justificaciones para los clientes no visitados. Posteriormente corriendo los procesos de sincronización necesarios y asegurando que dichos cambios se muestren con el sistema Bejerman con el cual se interactúa. A su vez, se inyectaron cambios en lista de precios, productos y clientes para controlar la detección de dichos cambios y su correcta sincronización hasta la aplicación del vendedor, finalizando un círculo completo de intercambio de información.

Para la realización de estos tests, se utilizaron dos entornos de prueba distintos. El primero es configurado de forma local en la computadora del desarrollador, en el que se ejecuta el sistema y es accesible a través del navegador web en la dirección *localhost.* Para que esto sea posible, es necesario instalar todos los paquetes y servicios que necesita el sistema para ejecutarse (paquetes de sistema operativo, intérpretes, librerías, etc), y además configurar las variables de entorno requeridas por el mismo. La ventaja que tiene este método es que es muy rápido ya que todas las conexiones son de forma local y se tiene control sobre cada componente. El segundo entorno de prueba configurado fue una instancia de testing en Heroku, la cual se configuró para que sea idéntica a la instancia futura de producción. Este método si bien lleva más tiempo y recursos (dado que algunos servicios pueden ser pagos), nos asegura que si el sistema funciona en esta instancia, también funcionará en producción.

Es un ambiente ideal para realizar pruebas tanto por parte del desarrollador como de los clientes, antes de dar paso a nuevas funcionalidades a la versión productiva del sistema.

**Test de aceptación**

Los tests de aceptación consisten en comparar el programa con sus requerimientos iniciales y con las necesidades actuales del usuario final. Este es un tipo de test particular pues lo realiza generalmente el usuario final del programa y, por lo común, no se considera responsabilidad de la organización de desarrollo.

Para los test de aceptación, se realizaron demostraciones del sistema a los clientes, y posteriormente interactuaban con el mismo durante un periodo de tiempo significativo para obtener una buena retroalimentación. Se utilizó el ambiente de testing configurado en Heroku para este propósito, dado que provee una URL de acceso público y de esta manera todos se pueden conectar a ella a través de un navegador web.

# 5. Conclusión

Este apartado tiene por objetivo presentar los principales aportes y experiencias que obtuve como alumno durante la realización de este proyecto final de carrera.

## 5.1. Principales aportes

La idea del proyecto surge a partir de la necesidad de los clientes de tener una mejor plataforma para el proyecto ODILA. El sistema con el que contaban era muy básico y carecía de funcionalidades muy importantes para los usuarios administradores, así como de flexibilidad para su extensión. Además, estaba implementado utilizando tecnologías un poco pasadas de moda, por lo que se decidió elegir una pila de tecnologías moderna y preparada para el desarrollo de aplicaciones *cloud*.

Con el nuevo sistema, los clientes dependen mucho menos del equipo de desarrollo para la realización de consultas de datos, modificación o alta de nuevos registros, y

elaboración de estadísticas. Las funcionalidades desarrolladas le brindan a los clientes administradores muchas herramientas para que puedan gestionar su plataforma sin la intervención del equipo de desarrollo.

Del lado del usuario denunciante, se encuentra con muchas mejoras en cuanto a experiencia de usuario. El nuevo sistema presenta la información de forma más modular, y el formulario de denuncia que originalmente era muy tedioso de completar, ahora se hace más ameno a través del comportamiento interactivo del mismo. El usuario se encontrará con animaciones, una dinámica de preguntas y respuestas una a la vez, y componentes de interfaz que minimicen la interacción necesaria por parte del mismo.

## 5.2. Experiencia personal

En cuanto a lo personal, el proyecto me aportó la experiencia de cómo debe desarrollarse un sistema en su totalidad, desde la captura de requerimientos, atravesando por el análisis, diseño y desarrollo hasta el mantenimiento del mismo, teniendo en cuenta muchos aspectos en el medio.

Por un lado, experimenté el contacto con los clientes: cómo tratar con ellos, las presiones que pueden ejercer y lograr entender lo que realmente quieren. Esta fue una experiencia nueva para mí, dado que si bien ya había trabajado en proyectos de desarrollo, nunca había tenido la principal responsabilidad sobre el mismo. La metodología de desarrollo elegida permitió proveer de entregas funcionales tempranas, lo que permitió la posibilidad de una rápida retroalimentación por parte de los usuarios para poder ir mejorando y adaptarse rápidamente a las nuevas necesidades.

Otro aspecto a destacar es el aprendizaje y profundización de nuevas tecnologías. En mi caso particular ya había utilizado gran parte de las herramientas usadas para desarrollo web, pero este proyecto planteó desafíos que no había tenido previamente. En lo que respecta a diseño particularmente, el requerimiento de poder construir el formulario de denuncia dinámicamente a través de un panel de administración fue todo un reto, dado que implicó la

definición de estructuras de datos y tablas que permitan definir distintos tipos de preguntas, distintos tipos de opciones y respuestas, todo utilizando una base de datos relacional.

Respecto al trabajo en equipo, si bien realicé este proyecto de forma unipersonal, debo mencionar que algunos miembros del equipo de ODILA se prestaron para la participación activa en el desarrollo del proyecto. Si bien no escribían el código fuente, teníamos una comunicación casi a diario respecto de cómo implementar las funcionalidades, qué tecnologías utilizar, y qué diseño elegir para cada una.

Quisiera agregar además como satisfactorio el hecho de poder haber plasmado todo lo aprendido durante los años de estudios en un proyecto real, que creo será de utilidad para una causa noble y sin fines de lucro. Veo esto como una forma de contribuir a la sociedad y devolverle parte de lo que me dió al permitirme estudiar en la universidad pública.

### 5.3. Presente y futuro

El sistema se encuentra en estado operativo y sometido a pruebas constantes, para pulir los detalles que pudieran haber quedado pendientes. El plan es reemplazar pronto el sistema antiguo por el nuevo, haciendo una transición al cien por ciento, dado que los datos ya fueron migrados.

Luego de diversas charlas con los clientes surgieron nuevas oportunidades y funcionalidades que desean agregar a la plataforma, dado que están conformes con el resultado. Esto se planificará y probablemente sea parte de un proyecto futuro para extender el sistema desarrollado.

## 6. Referencias bibliográficas

- *Real-World Kanban: Do Less, Accomplish More with Lean Thinking* por Mattias Skarin ( ISBN 9781680500776)
- *Agile Web Development with Rails 5* por Sam Ruby (ISBN 9781680501711)

- *JavaScript and JQuery: Interactive Front-End Web Development* por Jon Duckett (ISBN 9781118531648)
- *HTML and CSS: Design and Build Websites* por Jon Duckett (ISBN 1118008189)

**Sitios web oficiales**

- Wikipedia: https://www.wikipedia.org
- Ruby: https://www.ruby-lang.org/es/
- Rails: https://rubyonrails.org/
- JQuery: https://jquery.com/
- Twitter Bootstrap: http://getbootstrap.com/
- Heroku: http://www.heroku.com
- StackOverflow: https://stackoverflow.com/
- CSS Tricks: https://css-tricks.com/
- Trello: https://trello.com/
- Github: https://github.com/
- The Open Web Application Security Project: https://www.owasp.org/
- AsegurarTe: https://www.asegurarte.com.ar/
- Segu-Info: https://www.segu-info.com.ar/