



**Universidad  
Europea** MADRID

PROYECTO INTEGRADOR

Urban Fixer



Grupo Bit-Bang

Pablo Molina Hernández

Alberto Martín Naranjo

Chen YIJIN

## Índice

Resumen.....	2
1. Introducción .....	3
2. Objetivos .....	4
3. Tecnologías utilizadas.....	7
4. Desarrollo e implementación.....	9
5. Metodología .....	18
6. Resultados y conclusiones.....	22
7. Trabajos futuros .....	22
Anexos .....	24
Anexo I – Listado de requisitos de la aplicación .....	24
Anexo II – Guía de uso de la aplicación .....	34
Anexo III.....	39

## Resumen

Urban Fixer es una aplicación para facilitar la gestión de denuncias callejeras por código postal, cuyo fin no es otro que el de mejorar la comunicación entre los ciudadanos y las autoridades locales. La aplicación permite al usuario reportar incidencias, proporcionando descripciones y categorizando el tipo de denuncia para que el administrador la supervise y las pueda aceptar. Este proyecto ha sido elaborado con todo el conjunto de conocimientos adquirido en 1º de DAM de la Universidad Europea de Madrid, priorizando lenguajes de programación como Java y SQL, el uso de herramientas como Git o GitHub para permitir a los usuarios que puedan trabajar en conjunto, a través del control de versiones. Por otro lado, muy importante conocer por donde comenzamos un proyecto, la metodología Scrum, la cual nos agilizará el proceso de organización y los cuatro Sprints, en los que se ha sostenido nuestro proyecto. Después, hablar del desarrollo y diseño de una aplicación desde cero, con el uso de Diagramas de uso y luego de clases, que nos ayudarán a comprender mejor como actúa la base de datos dentro de nuestro programa y por último, como hemos ido fluyendo hasta el resultado final de la aplicación con toda su funcionalidad.

**Palabras clave:** denuncias, usuario, bienestar, ciudad, ciudadano.

## 1. Introducción

Somos el grupo Bit-Bang y queremos presentar nuestra aplicación, Urban Fixer. Esta herramienta está diseñada específicamente para facilitar la gestión de denuncias callejeras según el código postal en donde ocurran las incidencias, es decir, creada con el propósito de que se faciliten la gestión de denuncias callejeras, categorizándolas y poniéndoles localización sin entrar al detalle que la geolocalización o el GPS, que sería un proyecto muchísimo más ambicioso de lo que es nuestro nivel como alumnos de 1º de DAM. Con Urban Fixer, buscamos mejorar la eficiencia y efectividad de la resolución de problemas urbanos, promoviendo una comunicación más directa entre los ciudadanos y las autoridades locales promoviendo y siendo nuestro primordial objetivo que más tarde desarrollaremos en el siguiente apartado.

Uno de nuestros principales aliados en cuanto a la función de la aplicación es los ayuntamientos, los cuales son ellos los que tienen el poder de solucionar las incidencias en las calles. Nuestra inspiración surgió durante una visita al departamento de ciberseguridad del Ayuntamiento de Madrid. Allí observamos el crecimiento y la modernización de las ciudades, y cómo la gestión eficiente de incidencias en la vía pública es crucial para el bienestar urbano. Nos dimos cuenta de que esta idea podría tener un gran impacto, ya que son los ciudadanos quienes primero detectan los problemas en sus entornos. Nuestra aplicación podría ser el vínculo perfecto para mejorar la relación entre el Ayuntamiento y la ciudadanía.

Urban Fixer está enfocada en solucionar problemas cotidianos que afectan significativamente la calidad de vida en la comunidad si no se atienden a tiempo. Estamos hablando de baches, luminarias dañadas, vertidos ilegales de basura y otras irregularidades. Sin embargo, notamos que muchos ciudadanos desconocen los canales adecuados para reportar estas incidencias. Las vías tradicionales, como correos electrónicos o llamadas telefónicas, carecen de la inmediatez necesaria para una gestión eficaz. A continuación, nos dedicaremos a buscar sentido a esta idea proponiendo los objetivos que nos van a ayudar a desarrollar Urban Fixer.

La mala comunicación entre las autoridades y los ciudadanos puede llevar a una falta de atención a problemas que afectan diariamente a la comunidad. Urban Fixer pretende cambiar eso. Con nuestra aplicación, los usuarios pueden registrar denuncias detalladas, incluyendo descripciones, ubicaciones y fotos del incidente. Esta información es crucial para que las autoridades puedan priorizar y resolver las denuncias de manera más eficaz.

La mala comunicación entre las autoridades y los ciudadanos puede llevar a una falta de atención a problemas que afectan diariamente a la comunidad. Urban Fixer pretende cambiar eso. Con nuestra aplicación, los usuarios pueden registrar denuncias detalladas, incluyendo descripciones, ubicaciones y fotos del incidente. Esta información es crucial para que las autoridades puedan priorizar y resolver las denuncias de manera más eficaz. Los ciudadanos deben poder comunicar los problemas que observan en las calles de manera rápida y sencilla. Por eso, hemos identificado la necesidad de una aplicación que facilite esta comunicación inmediata. Urban Fixer está diseñada para cubrir esta necesidad, proporcionando una plataforma accesible y funcional para la gestión de denuncias callejeras.

## **2. Objetivos**

Nuestro principal objetivo del que partiremos será el de la imagen de ciudad ideal y digitalizada que se nos dio en el departamento de ciberseguridad de ayuntamiento de Madrid, una ciudad moderna sobre todo tiene que ser capaz de comunicar incidencias de manera instantánea para ayudar a la ciudadanía lo máximo posible. Otro punto muy a destacar sería la instantaneidad, convirtiéndose casi en nuestro eslogan ya que esa será la principal función de Urban Fixer, es decir, instantaneidad y rapidez a la hora de comunicar problemas cotidianos.

Además, Urban Fixer no solo mejora la comunicación entre ciudadanos y autoridades, sino que también promueve un entorno urbano más seguro y bien mantenido. La aplicación permite a los usuarios consultar el estado de sus denuncias, recibir actualizaciones sobre el progreso de su resolución y acceder a un historial detallado de denuncias previas. Para las

autoridades, hemos implementado herramientas robustas para filtrar, priorizar y asignar tareas relacionadas con las denuncias recibidas, así como la generación de reportes.

Los objetivos de nuestra aplicación son una extensión natural de la visión y misión que hemos discutido en el párrafo anterior. Nuestro principal objetivo es implantar una instantaneidad que permita una comunicación perfecta entre ciudadanos y autoridades. Esta frase encapsula la esencia de lo que queremos lograr y se ramifica en una serie de subobjetivos específicos. Para asegurar una implementación efectiva y organizada, hemos estructurado estos objetivos por temática, creando así una jerarquía clara y coherente dividida en diferentes módulos en los que nos vamos a centrar en una sección dentro de nuestra aplicación, los cuales los hemos separados en diferentes módulos.

### **Módulo de Interfaz de Usuario (UI)**

- **Diseño de una Interfaz Intuitiva y Amigable**, crearemos una interfaz de usuario que sea fácil de usar para todo tipo de edades y de habilidades con la tecnología. Aseguraremos de que los usuarios puedan navegar y utilizar la aplicación sin dificultades, proponiendo una estética sobria y deductiva que no recargue mucho al usuario, para que no canse el utilizarla.
- **Implementación de funcionalidades de accesibilidad**, incluíd características que hagan que la aplicación sea más accesible para personas con discapacidades, como las opciones de texto grande, compatibilidad con lectores de pantalla y botones de fácil acceso. Garantizar que todas las personas independientemente de sus capacidades físicas puedan utilizar Urban Fixer.

### **Módulo de localización y Código Postal**

- **Asignación de denuncias según el código postal**, desarrollo de un sistema que clasifique y que asigne un código postal que ponga el usuario. En este punto

también podemos mirar de cara al administrador el cual, será el que confirme las incidencias que les llegue, para localizarlas de manera más eficiente.

- **Del código postal a la geolocalización**, como hemos nombrado antes y seguiremos haciendo hincapié en el apartado de trabajos futuros, nos gustaría implementar la capacidad de que los ciudadanos puedan poner localización más precisa a las incidencias.

### **Módulo de Gestión de Incidencias**

- **Creación de una base de datos robusta**, para almacenar y gestionar la información de las incidencias reportadas. Para asegurarnos que todas las denuncias sean registradas y accesible para las autoridades, permitiendo una buena gestión organizada y sistemática de los problemas de la ciudad.
- **Desarrollo de panel de administración**, para que permita al usuario, visualizar, gestionar y actualizar el estado de las denuncias y posteriormente poder facilitar la labor de las autoridades locales a gestionar las incidencias de la manera más rápida y precisa.
- **Marcar denuncia como favorita**, para un acceso más rápido a las denuncias más relevantes, tanto para el usuario como para el administrador.

### **Módulo de seguridad y privacidad**

- **Garantizar la seguridad de los datos del usuario**, protegiendo toda su información personal y el anonimato a la hora de poner incidencias, por eso en nuestra aplicación no pedimos correos electrónicos, es decir, el usuario hace login mediante un nickname y la contraseña.
- **Cumplimiento con las Normativas de privacidad**, hay que asegurar que nuestra aplicación cumpla con las normativas de privacidad de datos.

### 3. Tecnologías utilizadas

En este apartado, englobaremos todas las tecnologías necesarias que nos han ayudado a crear nuestra aplicación, las cuales consideramos cruciales en cuanto al desarrollo de nuestra aplicación. Al igual que en los objetivos también hemos decidido presentarlas en diferentes módulos:

#### Lenguajes de programación

- **Java**, es el principal lenguaje de programación en el que nos hemos querido centrar con el JDK 21, el más moderno y adaptado a nuestros tiempos. Por otro lado, dentro de java, nos centramos en Swing para crear nuestra interfaz gráfica que es la que ayuda al usuario a navegar por nuestra aplicación.
- **SQL**, es el lenguaje utilizado para crear nuestra base de datos, creando tablas para almacenar los login, registros y denuncias de los usuarios.

#### Frameworks y Bibliotecas

- **Swing**, que contiene una base de datos gráficos y widgets diseñados para la elaboración de las ventanas de la aplicación, proporcionándonos botones que ayudan al usuario a usar Urban Fixer.
- **JDBC (Java Database Connectivity)**, nos permiten crear bases de datos relacionales, para llevar a cabo consultas SQL y administrar transacciones.

#### Entorno de Desarrollo

- **Eclipse**, como entorno de desarrollo principal y apoyándonos en el lenguaje Java, ha sido el IDE principal empleado para redactar, depurar y probar el código de la aplicación.



- **Oracle Express Edition**, para crear nuestra base de datos local y luego relacionarlo directamente con eclipse.
- **WireframeSketcher**, para el diseño primario de las ventanas de la aplicación antes de meternos de lleno con java.

### Control de Versiones y Colaboración

- **Git**, el sistema de control de versiones por excelencia, que nos posibilita a todos los desarrolladores implicados en este proyecto a trabajar de forma simultánea. Hemos utilizado Git para administrar el código fuente del proyecto, lo cual posibilita el trabajo en equipo a través de las ramas, commits y merges, para trabajar cada uno en su propia rama.
- **GitHub**, es la plataforma de almacenamiento de código fuente que aprovecha la aplicación de Git para controlar las versiones de nuestro proyecto. Hemos empleado el sitio web GitHub para albergar el repositorio del proyecto, lo cual posibilita la colaboración y el seguimiento de incidencias y solicitudes.
- **Trello**, es una buena solución en cuanto a la administración de proyectos que se estructuran en tableros, listas y tarjetas. Lo hemos usado como medio para organizar y asignar tareas, supervisar el progreso y administrar las etapas de desarrollo.
- **Google Drive**, para almacenar y sincronizar documentos, como por ejemplo este documento Word y poder colaborar todos, también lo hemos usado para compartir documentos, almacenar recursos del proyecto y colaborar en la elaboración de documentos de manera simultánea.

## Comunicación

- **WhatsApp**, el cual permite a los integrantes de este grupo poder conversar para decidir cuando quedar para desarrollar la aplicación, además de ser un gran vehículo por el cual nos hemos pasado mucho código.
- **Discord**, para llamadas de grupo y tener una comunicación instantánea con todos los miembros, su función más destacable es la de compartir pantalla la cual nos ayuda a mostrar nuestras ideas de código entre integrantes, a parte también utilizamos el chat.

## Herramientas de Pruebas y Validación

- **JUnit**, nos proporciona el llevar a cabo pruebas unitarias dentro del propio lenguaje de java, esto se ha utilizado para asegurar la calidad del código y los caminos que puede llegar a tomar y ver la eficacia del software.

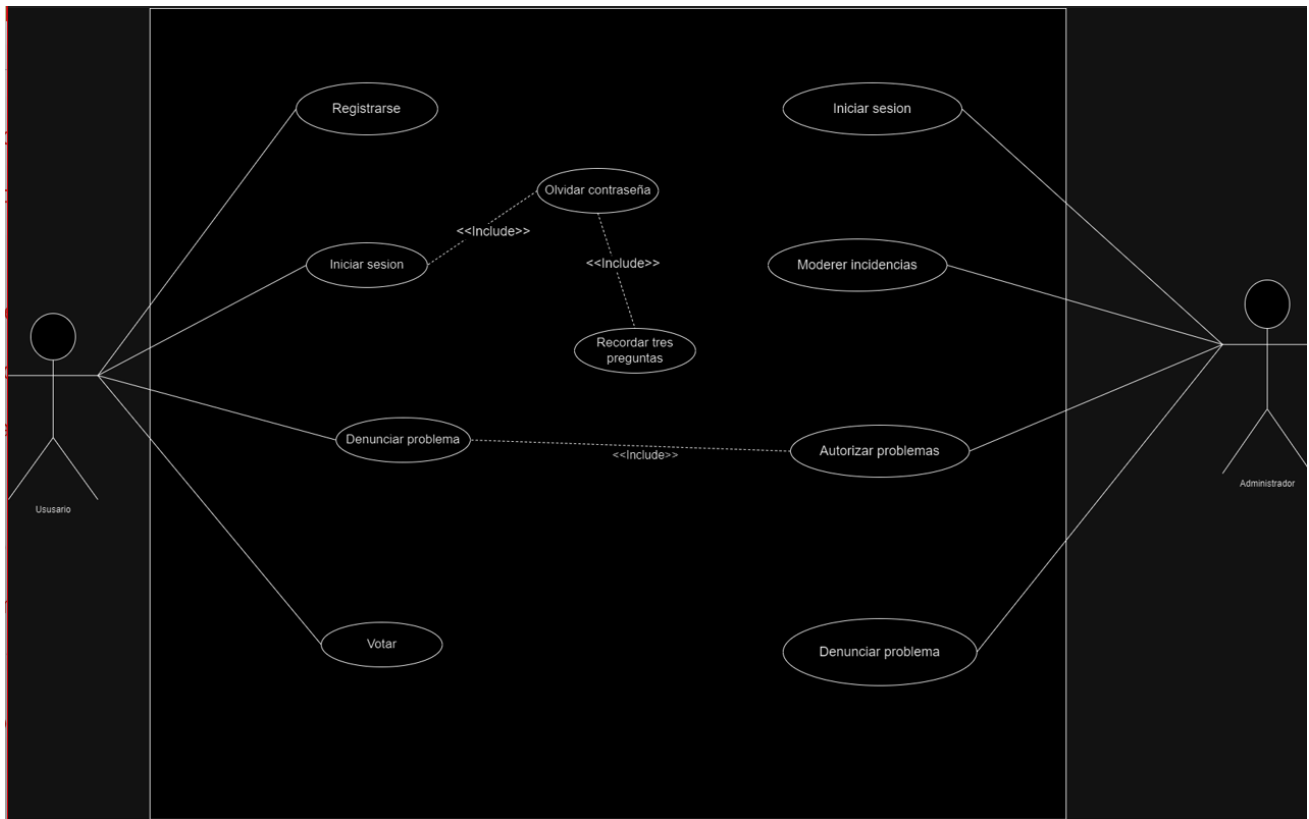
## Inteligencia Artificial

- **ChatGPT**, como apoyo y que nos ayude a dar ideas para nuestra aplicación.

Para finalizar, acabar diciendo que la implementación de todas estas tecnologías son las que nos han ayudado en nuestro día a día en el desarrollo de nuestro proyecto.

## 4. Desarrollo e implementación

En cuanto al desarrollo y la implementación, siendo este el apartado troncal de este documento. En primer lugar, que antes de realizar el código, hemos realizado un diagrama de caso de uso, para definir las diferentes interacciones que los usuarios pueden tener y definir también quienes son los actores principales y que van a hacer. En este apartado también nos vamos a dedicar a nombrar y explicar todas las etapas que ha tenido el proyecto y como las hemos ido desarrollando a lo largo del trimestre.



## Diagrama de Casos de Uso

El diagrama de caso de uso es fundamental para entender cómo interactúan los diferentes actores con la aplicación Urban Fixer. Este diagrama ayuda a visualizar los diferentes escenarios en los que la aplicación será utilizada y qué funcionalidades ofrece.

### Actores:

- Usuario: Representa a los usuarios del sistema que tienen acceso a ciertas funcionalidades.
- Administrador: Representa a los administradores del sistema que tienen privilegios adicionales para gestionar las incidencias reportadas por los usuarios.

### Casos de Uso del Usuario:

- Registrarse: permite al usuario crear una cuenta en el sistema.
- Iniciar sesión: permite al usuario autenticarse en el sistema utilizando sus credenciales.

- Olvidar contraseña: caso de uso para recuperar la contraseña olvidada. Este caso de uso incluye:
- Recordar tres preguntas: parte del proceso de recuperación de la contraseña donde el usuario responde a preguntas de seguridad.
- Denunciar problema: permite al usuario reportar un problema o incidencia en el sistema. Este caso de uso incluye:
- Autorizar problemas: autorización del problema reportado.
- Votar

### **Casos de Uso del Administrador:**

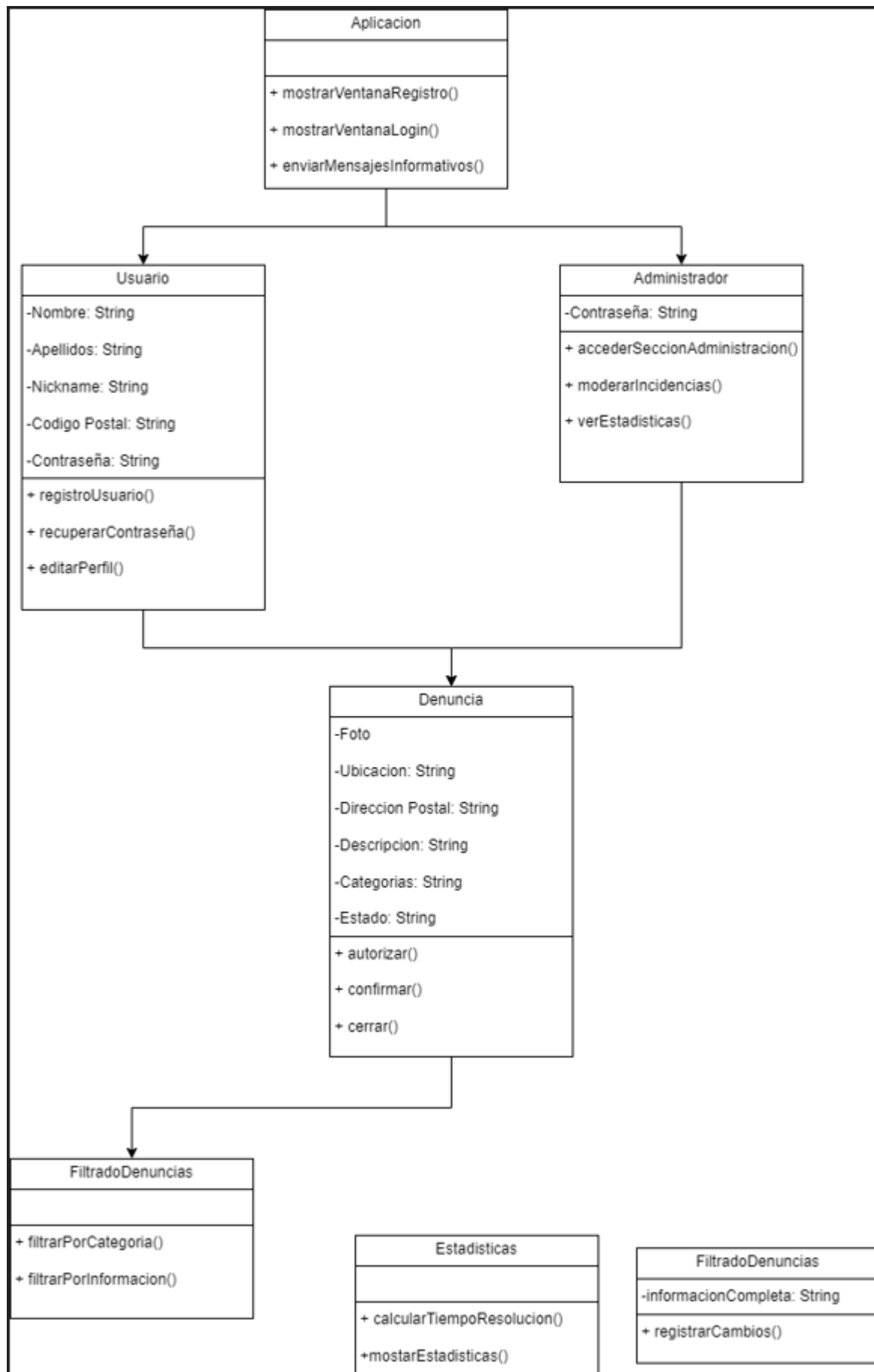
- Iniciar sesión: permite al administrador autenticarse en el sistema.
- Moderar incidencias: permite al administrador revisar y gestionar las incidencias reportadas por los usuarios.
- Autorizar problemas: permite al administrador autorizar o validar los problemas reportados por los usuarios.

### **Relaciones:**

- <<include>>: Se utiliza para indicar que un caso de uso incluye la funcionalidad de otro. Por ejemplo:
  - "Olvidar contraseña" incluye "Recordar tres preguntas".
  - "Denunciar problema" incluye "Autorizar problemas".

### **Diagrama de Clases**

El diagrama de clases nos proporciona una vista detallada de la estructura de la aplicación a nivel de código. Describe las clases principales, sus atributos, métodos y las relaciones entre ellas.



**Clases Principales:**▪ **Aplicacion**➤ **Métodos:**

- mostrarVentanaRegistro(): Muestra la ventana de registro para nuevos usuarios.
- mostrarVentanaLogin(): Muestra la ventana de inicio de sesión.
- enviarMensajesInformativos(): Envía mensajes informativos a los usuarios.

▪ **Usuario**➤ **Atributos:**

- Nombre: String: Nombre del usuario.
- Apellidos: String: Apellidos del usuario.
- Nickname: String: Apodo o nombre de usuario.
- Codigo Postal: String: Código postal del usuario.
- Contraseña: String: Contraseña del usuario.

➤ **Métodos:**

- registroUsuario(): Permite al usuario registrarse.
- recuperarContraseña(): Permite al usuario recuperar su contraseña.
- editarPerfil(): Permite al usuario editar su perfil.

▪ **Administrador**➤ **Atributos:**

- Contraseña: String: Contraseña del administrador.

➤ **Métodos:**

- accederSeccionAdministracion(): Permite al administrador acceder a la sección de administración.
- moderarIncidencias(): Permite al administrador moderar las incidencias reportadas.
- verEstadisticas(): Permite al administrador ver estadísticas del sistema.

▪ **Denuncia**➤ **Atributos:**

- Foto: Foto asociada a la denuncia.

- Ubicacion: String: Ubicación de la denuncia.
- Direccion Postal: String: Dirección postal de la denuncia.
- Descripcion: String: Descripción de la denuncia.
- Categorias: String: Categorías a las que pertenece la denuncia.
- Estado: String: Estado de la denuncia.

➤ **Métodos:**

- autorizar(): Permite autorizar una denuncia.
- confirmar(): Permite confirmar una denuncia.
- cerrar(): Permite cerrar una denuncia.

**Clases Auxiliares:**

- FiltradoDenuncias

➤ **Métodos:**

- filtrarPorCategoria(): Filtra las denuncias por categoría.
- filtrarPorInformacion(): Filtra las denuncias por información específica.
- Estadisticas

➤ **Métodos:**

- calcularTiempoResolucion(): Calcula el tiempo de resolución de las denuncias.
- mostrarEstadisticas(): Muestra las estadísticas calculadas.
- FiltradoDenuncias (otra clase con el mismo nombre, pero diferente propósito)

➤ **Atributos:**

- informacionCompleta: String: Información completa sobre las denuncias filtradas.

➤ **Métodos:**

- registrarCambios(): Registra los cambios realizados en las denuncias.

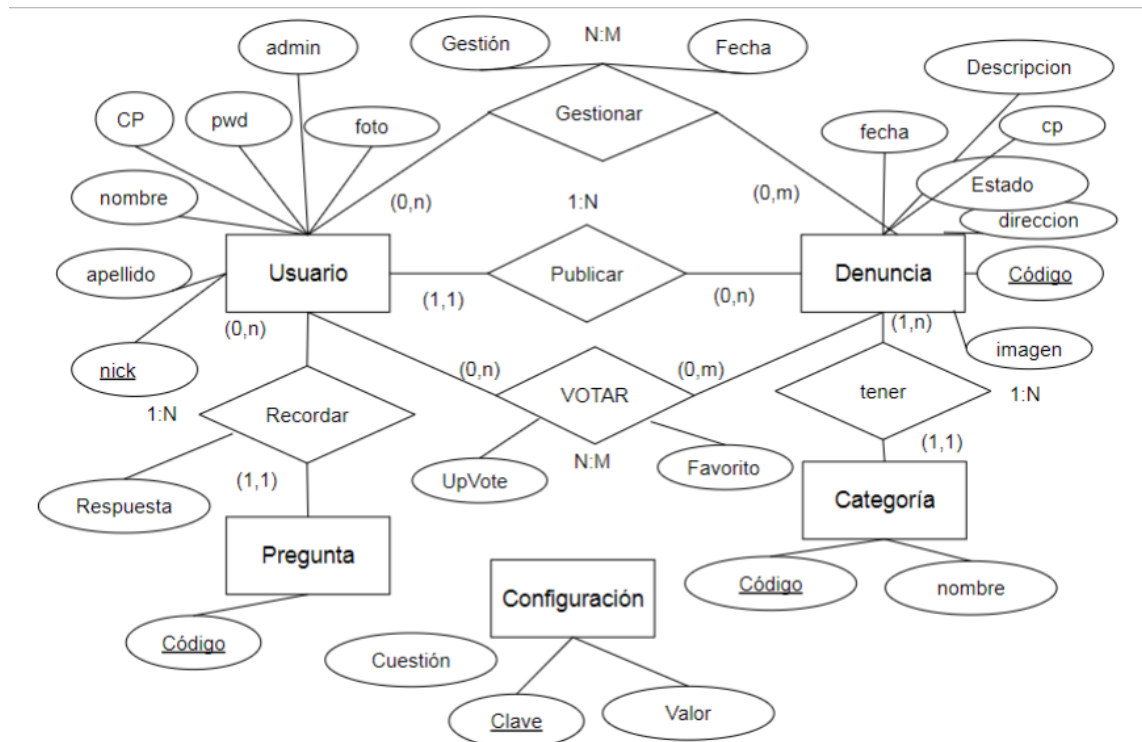
**Relaciones:** generalización (Herencia): La clase Usuario y Administrador heredan de la clase Aplicacion, indicando que ambos tipos de usuarios pueden interactuar con la aplicación.

**Asociaciones:** Usuario y Administrador están asociados con Denuncia, indicando que ambos roles pueden interactuar con las denuncias.

Denuncia está asociada con FiltradoDenuncias y Estadísticas, indicando que las denuncias pueden ser filtradas y analizadas para generar estadísticas.

## Bases de Datos

Por otro lado, realizar el Modelo Entidad-Relación para que se nos defina la estructura de la base de datos, siendo las principales tablas: usuarios, denuncia y administradores, es decir, las entidades principales con sus respectivos atributos que en los anexos colocaremos. En cuanto a las relaciones que hemos puesto, es que un usuario puede reportar múltiples incidencias y un administrador puede gestionar múltiples incidencias.





## Modelo Entidad Relación

Nos ayuda a estructurar y relacionar los datos de la aplicación, es esencial para entender cómo se relaciona la información que pasa por nuestra aplicación.

### Entidades y Atributos

- **Usuario:**
  - **Atributos:** CP (Código Postal), pwd (contraseña), foto, nombre, apellido, nick (nickname), admin. Esta entidad representa a los usuarios que pueden ser ciudadanos o administradores que interactúan con la aplicación.
- **Denuncia:** atributos: Descripción, fecha, cp (Código Postal), estado, dirección, imagen, código. Representa las incidencias o problemas reportados por los usuarios.
- **Pregunta:** Atributos: Código, Cuestión, Respuesta. Utilizada para gestionar preguntas de seguridad, probablemente relacionadas con la recuperación de contraseñas.
- **Categoría:** nivel de la denuncia.
  - **Atributos:** Código, nombre
- **Configuración:** ajustes del sistema según la necesidad de los usuarios.
  - Atributos: Clave, Valor

### Relaciones

- **Publicar:**
  - **Usuario a Denuncia:** Un usuario puede publicar muchas denuncias, pero cada denuncia es publicada por un único usuario.
- **Gestionar**

- Usuario a Denuncia: implica que los usuarios (probablemente administradores) pueden gestionar múltiples denuncias, y cada denuncia puede ser gestionada por múltiples usuarios.
- Recordar
  - Usuario a Pregunta: cada usuario tiene asociada una o más preguntas de seguridad para recordar su contraseña.
- Votar
  - Usuario a Denuncia: pueden votar múltiples denuncias, y cada denuncia puede recibir votos de múltiples usuarios.
- Favorito
  - Denuncia a Categoría: denuncias pueden ser clasificadas en una o más categorías, y cada categoría puede tener múltiples denuncias asociadas.
- UpVote
  - Usuario a Denuncia: la relación de votar, pero específica para la acción de dar un voto positivo.
- Fecha

Gestión a Denuncia: las denuncias tienen múltiples fechas de gestión asociadas, lo que puede indicar diferentes etapas o estados en su procesamiento.

### **Atributos Clave y Entidades Clave**

- Código (Categoría y Pregunta): Utilizado como identificador único para las categorías y preguntas.
- Clave (Configuración): Identifica las diferentes configuraciones del sistema.

### **Diseño del Logo**

El diseño de nuestro logotipo fue aportado por ChatGPT versión 4, al cual le pedimos un diseño minimalista y que fuese a la perfección con la aplicación.

### **Pruebas Realizadas**

Utilizamos el JUnit para realizar las pruebas para asegurar la calidad y el funcionamiento correcto de la aplicación. Estas pruebas irán adjuntas en el archivo de GitHub, con el código.

### **Proceso de diseño**

Realizamos el diseño de como iba a ser nuestra aplicación y de como iban a ir los botones situados a través, de la aplicación WireframeSketcher, la cual nos permitió hacer un concepto de aplicación minimalista para ayudar al usuario a moverse por ella sin sobrecargarse.

## **5. Metodología**

Para realizar este proyecto hemos aplicado una metodología ágil, en especial la Scrum, se adapta a la perfección con nuestro proyecto, permitiendo una organización efectiva y una gestión dinámica de las tareas. A continuación, vamos a detallar cómo se ha implementado Scrum en nuestro trabajo, incluyendo el flujo de trabajo con Sprint Backlog y Burndown Chart, y la coordinación del equipo para finalizar el proyecto.

### **Metodología Scrum**

Scrum es una metodología ágil que se centra en la entrega de incrementos de nuestro proyecto, en ciclos cortos llamados Sprints, los cuales vamos a ir detallando a lo largo de este epígrafe, este método nos ha ayudado también a mejorar continuamente nuestro proyecto a partir de la retroalimentación de todos los integrantes. En cuanto a los roles, no hemos especificado, porque todos los integrantes del grupo hemos ido haciendo diferentes funciones como, por ejemplo, el product owner, que es el que se encarga de definir las características del producto; el scrum master es el que gestiona la metodología

que le da su nombre y el equipo de desarrolladores y diseñadores. Alguno de estos roles los hemos practicado en clase.

Cada Sprint que hemos empleado ha tenido una duración justa de dos semanas e iba cambiando cada dos lunes. También hemos empleado Sprint Planning, para planificar cada el trabajo que se realiza en el Sprint. Daily Standup, todos los días el equipo se reúne 15 minutos para hablar de cómo va el proyecto. Sprint Review, para hacer feedback a cada Sprint al final de él y, por último Sprint Retrospective, para analiza que funcionó y para saber que mejorar de cara al siguiente. Estas metodologías no las hemos seguido rigurosamente, pero las hemos ensayado en el aula con la docente.

## Herramientas de Comunicación y Gestión

- **Trello**, como hemos nombrado en el anterior apartado utilizamos Trello para gestionar el Sprint Backlog y las tareas diarias. Cada tarjeta en Trello representaba una tarea, y el equipo podía mover las tarjetas a través de las columnas (To Do, In Progress, Done) para reflejar el estado actual del trabajo. Aquí dejamos adjunto el enlace:  
<https://trello.com/invite/b/iXQAGwXg/ATTI1a5d9edf8fcead496f940039323d51750899B5FB/proyecto-integrador>
- **GitHub**, utilizamos GitHub para el control de versiones y la colaboración en el código. Cada miembro del equipo tenía acceso al repositorio, donde podían hacer commits y pull requests para integrar sus cambios.
- **Enlace a GitHub Pedro:** <https://github.com/DAM-UEM-2324/uf52-tarea-final-pi-bit-bang.git>
- **Enlace a GitHub Irene:** <https://github.com/tuxiito77/WikiUrbanFixer.git>
  - ❖ En este repositorio se encuentran la Wiki, pruebas de JUnit y JavaDoc.

## Reuniones y Planificación

- **Reuniones de Planificación**, cada Sprint comenzó con una reunión de planificación en la que se definieron los objetivos y las tareas del Sprint. Estas reuniones aseguraron que todos los miembros del equipo entendieran sus responsabilidades y las prioridades del Sprint.

**Daily Standups**, las reuniones diarias fueron esenciales para mantener al equipo sincronizado. En estas reuniones, cada miembro del equipo respondía a tres preguntas clave:

*¿Qué hice ayer?*

*¿Qué haré hoy?*

*¿Hay algún obstáculo en mi camino?*

- **Revisiones de Sprint y Retrospectivas**, al final de cada Sprint, realizamos una revisión para presentar el trabajo completado y una retrospectiva para discutir mejoras. Estas reuniones fomentaron la transparencia y la mejora continua.

### Sprint 1

En este primer Sprint, tuvimos una toma de contacto con esta metodología y se nos mandó hacer un Trello para organizar lo que íbamos a hacer en cada Sprint y los días que iban a oscilar

### Sprint 2

En este apartado se nos asignó la realización de un diagrama de casos de uso y la realización de un loco para nuestra aplicación, que adjuntaremos más tarde en los anexos del trabajo.

### Sprint 3

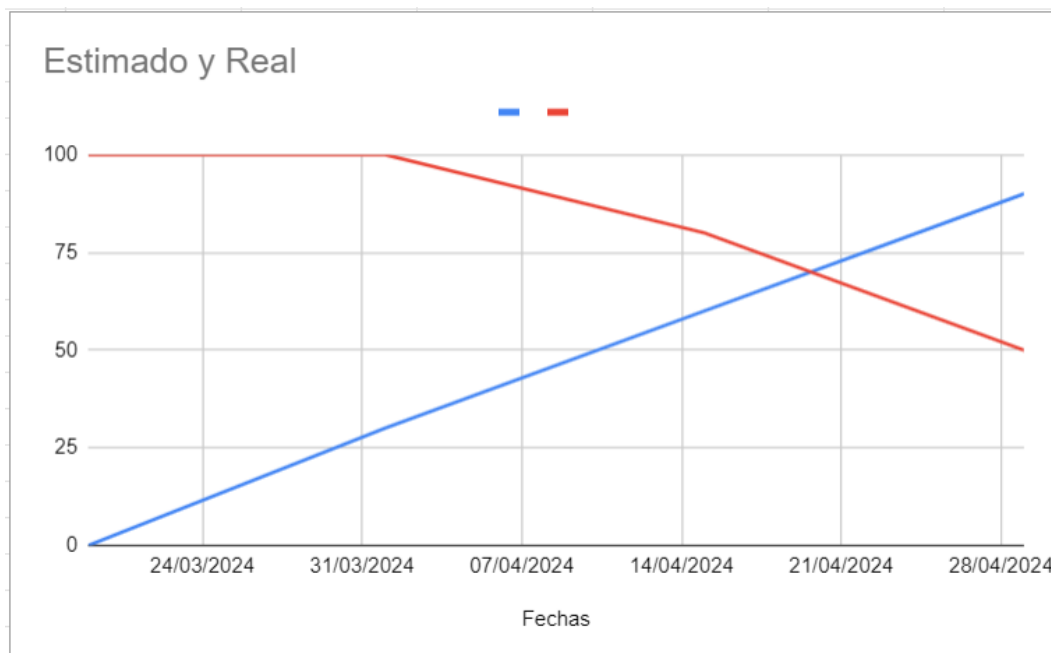
En este tercer Sprint, realizamos un diagrama de clases a partir del anterior que hicimos, que más tarde adjuntaremos en anexos.

### Sprint 4

En este último y más completo de los cuatro, realizamos pruebas con JUnit, para conocer las limitaciones de nuestro código. Después, realizamos un documento de JavaDoc para aplicarlo y realizamos también el manual de usuario en la wiki de GitHub.

### Burndown Chart

Es una herramienta que nos permite un visual que representa el progreso del Sprint y la cantidad de trabajo restante y el tiempo que disponemos para realizarlo. También lo añadiremos al anexo de imágenes.



## 6. Resultados y conclusiones

La frase con la que resumiríamos el final el proyecto integrado sería: *Urban Fixer es el punto de unión de la ciudadanía con las autoridades* y como hemos nombrado a lo largo del proyecto, queremos conseguir una gestión eficiente de las denuncias, por eso nosotros somos el nexo para el bienestar de la ciudadanía. Este proyecto también tratara de dar un paso hacia delante en la gestión de comunidades urbanas y causando un impacto positivo en el futuro, para poder fortalecer las relaciones entre ayuntamientos y ciudadanía y al fin conseguir mejorar la calidad de vida de los habitantes.

Haciendo repaso en general de la realización del proyecto integrador, nos ha gustado mucho la primera toma de contacto y los juegos de roles de cada uno de los integrantes, a lo único que podemos poner pegas es al poco tiempo que se nos ha concedido para realizar esta actividad, todo ha sido demasiado precipitado y no nos ha dado tiempo a empatizar más con el personaje, es decir, con el papel de desarrollado. Nos gustaría que el próximo curso y ya conociendo como va y teniendo más experiencia, que se haga una experiencia más placentera para nosotros los alumnos de DAM. La metodología de Sprints nos ha parecido bastante acertada porque dinamiza y ahorra tiempo, en vez de hacer como antaño reuniones largas y monótonas, que se realizan en empresas que usan metodologías más anticuadas.

Por otro lado, y mirando desde el punto positivo, este proyecto nos ha ayudado a saber como combatir los contratiempos que han surgido a través del trimestre, a veces las cosas no van como nosotros queremos, pero son las que luego nos harán más fuertes de cara a la vida laboral dentro de las empresas.

## 7. Trabajos futuros

A medida que íbamos realizando la aplicación, a nuestro grupo se le vino a la cabeza ideas muy ambiciosas pero que creemos que todavía escapan a nuestro conocimiento, por

eso lo mejor es que queden plasmadas en este escrito para poder mirar atrás algún día y poder rescatarlas.

### **Geolocalización Precisa**

Además, de ser de una aplicación para crear denuncias, podría convertirse en un futuro en una aplicación de socorro, que pueda a llegar a ayudar a personas en el mismo instante, es decir, a si alguien sufre algún robo o altercado mayor y que quede constancia de donde sucedió y pueda haber una rápida intervención policial.

### **Mapa Interactivo**

Un mapa interactivo que nos vaya señalando los lugares donde se frecuentes más incidencias dentro de un código postal, y poder marcar “zonas calientes”, donde las autoridades deben de estar más pendiente.

### **Inteligencia Artificial**

Sobre todo, para adaptarnos más aun a las necesidades de los ciudadanos y con la inclusión de un chat bot que ayude al ciudadano con las denuncias o hasta que le enseñe a manejar la propia aplicación o incluso a redactar las denuncias automáticamente.

### **Accesibilidad a las personas discapacitadas**

Como hemos nombrado en uno de nuestros objetivos, pensamos siempre en la accesibilidad y en abrirse a todos los públicos, uno de los fines también de Urban Fixer es el que todo el mundo pueda manejarla sin importar su condición social.

### **Sistema de Recompensa**

En este caso no se recompensaría con nada especial, sino como en Waze con algún icono de perfil, con estrellas y puntos.



## Anexos

### Anexo I – Listado de requisitos de la aplicación

#### Registro de usuario

RQ1: Durante el registro no intervendrán terceros, será mediante un autorregistro y una prueba anti-robots (un captcha).

RQ2.Consentimiento de notificaciones y términos al terminar el registro (incluyendo que eres mayor de 14 años)

RQ3: Después del registro, se mostrará una ventana de login.

RQ4.-Recuperar la contraseña a través de preguntas

RQ5-Contraseña segura=> la contraseña se guarda hasheada.

RQ6-Dos opciones a elegir si eres administrador o no y el administrador debe de tener que introducir una contraseña específica.

RQ7- Campos: Nombre y apellidos, nickname (opción de generación automática y debe ser único), CP, contraseña, y repetir contraseña para asegurarse de que está bien escrita. Todos son obligatorios.

## **Login**

RQ8 -Ventana única para administrador en el cual al administrador se le envíe a una sección reservada para los administradores los cuales podrán moderar las incidencias, y por el otro lado a los usuarios se le enviaría a la sección para poner las incidencias.

RQ9 - Para evitar fuerza bruta, cerrar el programa al tercer intento fallido.

RQ10 - Tener que introducir el nickname y la contraseña. (Ambos son obligatorios).

RQ11: Al iniciar la aplicación se mostrará la pantalla de login y desde esta podrás ir al registro si todavía no lo has realizado, y en desde aquí a la pestaña del olvido de contraseña (contestando a las preguntas seleccionadas en el registro).

## **Denuncias**

RQ12: Para denunciar un problema, se deberá proporcionar la siguiente información: - Foto - Ubicación (se generará automáticamente en la versión 2) - Dirección postal - Descripción del problema - Categorías.

RQ13: El administrador, autorizará los problemas, definirá cuando los casos estén cerrados confirmando la solución.

RQ14: Habrá un apartado donde se vean las denuncias marcadas como favoritas. Para que se pueda hacer un especial seguimiento de ellas.

RQ15: Un usuario puede avisar de que el problema ya ha sido arreglado.

RQ16: El administrador podrá ver la información completa de las denuncias, es decir, información de auditoría.

RQ17: Los usuarios podrán confirmar una denuncia ya existente.

RQ18: Sistema de filtrados: Para usuarios y administrador, los koalas las denuncias tengan categorías o información que haga más fácil buscar publicaciones que estás buscando (Usuario debe ponerlas).

RQ19: Si el usuario introduce una denuncia ya escrita (que coincida con el código postal y mismo tipo de denuncia), la aplicación le preguntará si se refiere a una de las publicadas, si es así podrá reconfirmarla.

RQ20: El panel de administrador mostrará estadísticas sobre los tipos de problemas, las zonas y el tiempo de resolución de forma sencilla.

RQ21: Las denuncias pueden pasar por los siguientes estados: Nueva (solo visible para el creador y el administrador), Publicada/Rechazada, el administrador puede publicar y rechazar las denuncias (si es rechazada explicar el porqué), En proceso y Finalizada. Cada nuevo estado se registrará la fecha.

## Base de Datos para el funcionamiento de la aplicación

```
-- Creación de tablas
CREATE TABLE Categoria (
    Codigo INTEGER NOT NULL ,
    nombre VARCHAR2(50) NOT NULL ,
    Denuncia_codigo INTEGER NOT NULL ,
    Denuncia_Usuario_Nick VARCHAR2(20) NOT NULL
);

CREATE TABLE Configuracion (
    Clave INTEGER NOT NULL ,
    Valor VARCHAR2(50) NOT NULL
);

CREATE TABLE Denuncia (
    codigo INTEGER NOT NULL ,
    fecha DATE NOT NULL ,
    Estado NUMBER(1,0) NOT NULL ,
    direccion VARCHAR2(250) NOT NULL ,
    imagen BLOB NOT NULL ,
    Usuario_Nick VARCHAR2(20) NOT NULL ,
    Fecha1 DATE ,
    Gestion DATE
);

CREATE TABLE Gestionar (
    Usuario_Nick VARCHAR2(20) NOT NULL ,
    Denuncia_codigo INTEGER NOT NULL ,
    Denuncia_Nick VARCHAR2(20) NOT NULL ,
    Gestion DATE
);

CREATE TABLE Pregunta (
    Codigo INTEGER NOT NULL ,
    Cuestion VARCHAR2(200) NOT NULL ,
    Usuario_Nick VARCHAR2(20) NOT NULL
);

CREATE TABLE Usuario (
    Nick VARCHAR2(20) NOT NULL ,
    apellido VARCHAR2(20) NOT NULL ,
    nombre VARCHAR2(20) NOT NULL ,
    CP INTEGER NOT NULL ,
    Pwd VARCHAR2(25) ,
    foto BLOB NOT NULL
);

CREATE TABLE Votar (
    Usuario_Nick VARCHAR2(20) NOT NULL ,
    Denuncia_codigo INTEGER NOT NULL ,
    Denuncia_Usuario_Nick VARCHAR2(20) NOT NULL ,
    UpVote INTEGER ,
    Favorito NUMBER(1,0)
);
```

```

ALTER TABLE Categoria ADD CONSTRAINT Categoria_PK PRIMARY KEY (Codigo);
ALTER TABLE Configuracion ADD CONSTRAINT Configuracion_PK PRIMARY KEY
(Clave);
ALTER TABLE Denuncia ADD CONSTRAINT Denuncia_PK PRIMARY KEY (codigo,
Usuario_Nick);
ALTER TABLE Gestionar ADD CONSTRAINT Gestionar_PK PRIMARY KEY
(Usuario_Nick, Denuncia_codigo, Denuncia_Nick);
ALTER TABLE Pregunta ADD CONSTRAINT Pregunta_PK PRIMARY KEY (Codigo);
ALTER TABLE Usuario ADD CONSTRAINT Usuario_PK PRIMARY KEY (Nick);
ALTER TABLE Votar ADD CONSTRAINT Votar_PK PRIMARY KEY (Usuario_Nick,
Denuncia_codigo, Denuncia_Usuario_Nick);

-- Inserción de datos en la tabla Configuracion
INSERT INTO Configuracion (Clave, Valor) VALUES (1, 'Config1');
INSERT INTO Configuracion (Clave, Valor) VALUES (2, 'Config2');
INSERT INTO Configuracion (Clave, Valor) VALUES (3, 'Config3');
INSERT INTO Configuracion (Clave, Valor) VALUES (4, 'Config4');
INSERT INTO Configuracion (Clave, Valor) VALUES (5, 'Config5');

-- Inserción de datos en la tabla Denuncia
INSERT INTO Denuncia (codigo, fecha, Estado, direccion, imagen,
Usuario_Nick, Fecha1, Gestion) VALUES
(101, TO_DATE('2024-04-01', 'YYYY-MM-DD'), 1, 'Direccion1', EMPTY_BLOB(),
'Usuario1', TO_DATE('2024-04-02', 'YYYY-MM-DD'), TO_DATE('2024-04-03',
'YYYY-MM-DD'));
INSERT INTO Denuncia (codigo, fecha, Estado, direccion, imagen,
Usuario_Nick, Fecha1, Gestion) VALUES
(102, TO_DATE('2024-04-02', 'YYYY-MM-DD'), 0, 'Direccion2', EMPTY_BLOB(),
'Usuario2', TO_DATE('2024-04-03', 'YYYY-MM-DD'), TO_DATE('2024-04-04',
'YYYY-MM-DD'));
INSERT INTO Denuncia (codigo, fecha, Estado, direccion, imagen,
Usuario_Nick, Fecha1, Gestion) VALUES
(103, TO_DATE('2024-04-03', 'YYYY-MM-DD'), 1, 'Direccion3', EMPTY_BLOB(),
'Usuario3', TO_DATE('2024-04-04', 'YYYY-MM-DD'), TO_DATE('2024-04-05',
'YYYY-MM-DD'));
INSERT INTO Denuncia (codigo, fecha, Estado, direccion, imagen,
Usuario_Nick, Fecha1, Gestion) VALUES
(104, TO_DATE('2024-04-04', 'YYYY-MM-DD'), 0, 'Direccion4', EMPTY_BLOB(),
'Usuario4', TO_DATE('2024-04-05', 'YYYY-MM-DD'), TO_DATE('2024-04-06',
'YYYY-MM-DD'));
INSERT INTO Denuncia (codigo, fecha, Estado, direccion, imagen,
Usuario_Nick, Fecha1, Gestion) VALUES
(105, TO_DATE('2024-04-05', 'YYYY-MM-DD'), 1, 'Direccion5', EMPTY_BLOB(),
'Usuario5', TO_DATE('2024-04-06', 'YYYY-MM-DD'), TO_DATE('2024-04-07',
'YYYY-MM-DD'));

-- Inserción de datos en la tabla Categoria
INSERT INTO Categoria (Codigo, nombre, Denuncia_codigo,
Denuncia_Usuario_Nick) VALUES (1, 'Categoria A', 101, 'Usuario1');
INSERT INTO Categoria (Codigo, nombre, Denuncia_codigo,
Denuncia_Usuario_Nick) VALUES (2, 'Categoria B', 102, 'Usuario2');
INSERT INTO Categoria (Codigo, nombre, Denuncia_codigo,
Denuncia_Usuario_Nick) VALUES (3, 'Categoria C', 103, 'Usuario3');
INSERT INTO Categoria (Codigo, nombre, Denuncia_codigo,
Denuncia_Usuario_Nick) VALUES (4, 'Categoria D', 104, 'Usuario4');
INSERT INTO Categoria (Codigo, nombre, Denuncia_codigo,
Denuncia_Usuario_Nick) VALUES (5, 'Categoria E', 105, 'Usuario5');

```

```
-- Inserción de datos en la tabla Gestionar
INSERT INTO Gestionar (Usuario_Nick, Denuncia_codigo, Denuncia_Nick,
Gestion) VALUES ('Usuario1', 101, 'Usuario2', TO_DATE('2024-04-03', 'YYYY-
MM-DD'));
INSERT INTO Gestionar (Usuario_Nick, Denuncia_codigo, Denuncia_Nick,
Gestion) VALUES ('Usuario2', 102, 'Usuario3', TO_DATE('2024-04-04', 'YYYY-
MM-DD'));
INSERT INTO Gestionar (Usuario_Nick, Denuncia_codigo, Denuncia_Nick,
Gestion) VALUES ('Usuario3', 103, 'Usuario4', TO_DATE('2024-04-05', 'YYYY-
MM-DD'));
INSERT INTO Gestionar (Usuario_Nick, Denuncia_codigo, Denuncia_Nick,
Gestion) VALUES ('Usuario4', 104, 'Usuario5', TO_DATE('2024-04-06', 'YYYY-
MM-DD'));
INSERT INTO Gestionar (Usuario_Nick, Denuncia_codigo, Denuncia_Nick,
Gestion) VALUES ('Usuario5', 105, 'Usuario1', TO_DATE('2024-04-07', 'YYYY-
MM-DD'));

-- Inserción de datos en la tabla Pregunta
INSERT INTO Pregunta (Codigo, Cuestion, Usuario_Nick) VALUES (1, '¿Cómo
puedo mejorar mi aplicación?', 'Usuario1');
INSERT INTO Pregunta (Codigo, Cuestion, Usuario_Nick) VALUES (2, '¿Cuál es
la mejor manera de optimizar consultas SQL?', 'Usuario2');
INSERT INTO Pregunta (Codigo, Cuestion, Usuario_Nick) VALUES (3, '¿Qué
lenguaje de programación debería aprender a continuación?', 'Usuario3');
INSERT INTO Pregunta (Codigo, Cuestion, Usuario_Nick) VALUES (4, '¿Cuál es
la mejor práctica para asegurar una red de computadoras?', 'Usuario4');
INSERT INTO Pregunta (Codigo, Cuestion, Usuario_Nick) VALUES (5, '¿Cómo
puedo mejorar la seguridad de mi sitio web?', 'Usuario5');

-- Inserción de datos en la tabla Usuario
INSERT INTO Usuario (Nick, apellido, nombre, CP, Pwd, foto) VALUES
('Usuario1', 'Johnson', 'John', 12345, 'password1', EMPTY_BLOB());
INSERT INTO Usuario (Nick, apellido, nombre, CP, Pwd, foto) VALUES
('Usuario2', 'Smith', 'Robert', 23456, 'password2', EMPTY_BLOB());
INSERT INTO Usuario (Nick, apellido, nombre, CP, Pwd, foto) VALUES
('Usuario3', 'Jones', 'Richard', 34567, 'password3', EMPTY_BLOB());
INSERT INTO Usuario (Nick, apellido, nombre, CP, Pwd, foto) VALUES
('Usuario4', 'Davis', 'David', 45678, 'password4', EMPTY_BLOB());
INSERT INTO Usuario (Nick, apellido, nombre, CP, Pwd, foto) VALUES
('Usuario5', 'Taylor', 'James', 56789, 'password5', EMPTY_BLOB());

-- Inserción de datos en la tabla Votar
INSERT INTO Votar (Usuario_Nick, Denuncia_codigo, Denuncia_Usuario_Nick,
UpVote, Favorito) VALUES ('Usuario1', 101, 'Usuario2', 1, 1);
INSERT INTO Votar (Usuario_Nick, Denuncia_codigo, Denuncia_Usuario_Nick,
UpVote, Favorito) VALUES ('Usuario2', 102, 'Usuario3', 0, 0);
INSERT INTO Votar (Usuario_Nick, Denuncia_codigo, Denuncia_Usuario_Nick,
UpVote, Favorito) VALUES ('Usuario3', 103, 'Usuario4', 1, 1);
INSERT INTO Votar (Usuario_Nick, Denuncia_codigo, Denuncia_Usuario_Nick,
UpVote, Favorito) VALUES ('Usuario4', 104, 'Usuario5', 0, 0);
INSERT INTO Votar (Usuario_Nick, Denuncia_codigo, Denuncia_Usuario_Nick,
UpVote, Favorito) VALUES ('Usuario5', 105, 'Usuario1', 1, 1);
```

```
--Estos fueron los datos ingresados en cada una de las tablas

-- Inserción de datos en la tabla Configuración
INSERT INTO Configuración (Clave, Valor) VALUES (6, 'Config6');
INSERT INTO Configuración (Clave, Valor) VALUES (7, 'Config7');
INSERT INTO Configuración (Clave, Valor) VALUES (8, 'Config8');
INSERT INTO Configuración (Clave, Valor) VALUES (9, 'Config9');
INSERT INTO Configuración (Clave, Valor) VALUES (10, 'Config10');

-- Inserción de datos en la tabla Denuncia
INSERT INTO Denuncia (codigo, fecha, Estado, direccion, imagen,
Usuario_Nick, Fecha1, Gestion) VALUES
(106, TO_DATE('2024-04-06', 'YYYY-MM-DD'), 0, 'Direccion6', EMPTY_BLOB(),
'Usuario1', TO_DATE('2024-04-07', 'YYYY-MM-DD'), TO_DATE('2024-04-08',
'YYYY-MM-DD'));
INSERT INTO Denuncia (codigo, fecha, Estado, direccion, imagen,
Usuario_Nick, Fecha1, Gestion) VALUES
(107, TO_DATE('2024-04-07', 'YYYY-MM-DD'), 1, 'Direccion7', EMPTY_BLOB(),
'Usuario2', TO_DATE('2024-04-08', 'YYYY-MM-DD'), TO_DATE('2024-04-09',
'YYYY-MM-DD'));
INSERT INTO Denuncia (codigo, fecha, Estado, direccion, imagen,
Usuario_Nick, Fecha1, Gestion) VALUES
(108, TO_DATE('2024-04-08', 'YYYY-MM-DD'), 0, 'Direccion8', EMPTY_BLOB(),
'Usuario3', TO_DATE('2024-04-09', 'YYYY-MM-DD'), TO_DATE('2024-04-10',
'YYYY-MM-DD'));
INSERT INTO Denuncia (codigo, fecha, Estado, direccion, imagen,
Usuario_Nick, Fecha1, Gestion) VALUES
(109, TO_DATE('2024-04-09', 'YYYY-MM-DD'), 1, 'Direccion9', EMPTY_BLOB(),
'Usuario4', TO_DATE('2024-04-10', 'YYYY-MM-DD'), TO_DATE('2024-04-11',
'YYYY-MM-DD'));
INSERT INTO Denuncia (codigo, fecha, Estado, direccion, imagen,
Usuario_Nick, Fecha1, Gestion) VALUES
(110, TO_DATE('2024-04-10', 'YYYY-MM-DD'), 0, 'Direccion10', EMPTY_BLOB(),
'Usuario5', TO_DATE('2024-04-11', 'YYYY-MM-DD'), TO_DATE('2024-04-12',
'YYYY-MM-DD'));

-- Inserción de datos en la tabla Categoría

INSERT INTO Categoría (Codigo, nombre, Denuncia_codigo,
Denuncia_Usuario_Nick) VALUES (6, 'Categoría F', 106, 'Usuario6');
INSERT INTO Categoría (Codigo, nombre, Denuncia_codigo,
Denuncia_Usuario_Nick) VALUES (7, 'Categoría G', 107, 'Usuario7');
INSERT INTO Categoría (Codigo, nombre, Denuncia_codigo,
Denuncia_Usuario_Nick) VALUES (8, 'Categoría H', 108, 'Usuario8');
INSERT INTO Categoría (Codigo, nombre, Denuncia_codigo,
Denuncia_Usuario_Nick) VALUES (9, 'Categoría I', 109, 'Usuario9');
INSERT INTO Categoría (Codigo, nombre, Denuncia_codigo,
Denuncia_Usuario_Nick) VALUES (10, 'Categoría J', 110, 'Usuario10');

-- Inserción de datos en la tabla Gestionar

INSERT INTO Gestionar (Usuario_Nick, Denuncia_codigo, Denuncia_Nick,
Gestion) VALUES ('Usuario6', 106, 'Usuario7', TO_DATE('2024-04-08', 'YYYY-
MM-DD'));
```

```
INSERT INTO Gestionar (Usuario_Nick, Denuncia_codigo, Denuncia_Nick,
Gestion) VALUES ('Usuario7', 107, 'Usuario8', TO_DATE('2024-04-09', 'YYYY-
MM-DD'));
INSERT INTO Gestionar (Usuario_Nick, Denuncia_codigo, Denuncia_Nick,
Gestion) VALUES ('Usuario8', 108, 'Usuario9', TO_DATE('2024-04-10', 'YYYY-
MM-DD'));
INSERT INTO Gestionar (Usuario_Nick, Denuncia_codigo, Denuncia_Nick,
Gestion) VALUES ('Usuario10', 109, 'Usuario6', TO_DATE('2024-04-11', 'YYYY-
MM-DD'));
INSERT INTO Gestionar (Usuario_Nick, Denuncia_codigo, Denuncia_Nick,
Gestion) VALUES ('Usuario9', 110, 'Usuario10', TO_DATE('2024-04-12', 'YYYY-
MM-DD'));
```

-- Inserción de datos en la tabla Pregunta

```
INSERT INTO Pregunta (Codigo, Cuestion, Usuario_Nick) VALUES (6, '¿Cómo
puedo mejorar mi base de datos?', 'Usuario6');
INSERT INTO Pregunta (Codigo, Cuestion, Usuario_Nick) VALUES (7, '¿Cuál es
la mejor manera de optimizar consultas de bases de datos?', 'Usuario7');
INSERT INTO Pregunta (Codigo, Cuestion, Usuario_Nick) VALUES (8, '¿Qué
lenguaje de programación debería aprender a continuación para el desarrollo
web?', 'Usuario8');
INSERT INTO Pregunta (Codigo, Cuestion, Usuario_Nick) VALUES (9, '¿Cuál es
la mejor práctica para asegurar una red de servidores?', 'Usuario9');
INSERT INTO Pregunta (Codigo, Cuestion, Usuario_Nick) VALUES (10, '¿Cómo
puedo mejorar la seguridad de mi aplicación web?', 'Usuario10');
```

-- Inserción de datos en la tabla Usuario

```
INSERT INTO Usuario (Nick, apellido, nombre, CP, Pwd, foto) VALUES
('Usuario6', 'Williams', 'William', 67890, 'password6', EMPTY_BLOB());
INSERT INTO Usuario (Nick, apellido, nombre, CP, Pwd, foto) VALUES
('Usuario7', 'Brown', 'Michael', 78901, 'password7', EMPTY_BLOB());
INSERT INTO Usuario (Nick, apellido, nombre, CP, Pwd, foto) VALUES
('Usuario8', 'Miller', 'Charles', 89012, 'password8', EMPTY_BLOB());
INSERT INTO Usuario (Nick, apellido, nombre, CP, Pwd, foto) VALUES
('Usuario9', 'Wilson', 'Joseph', 90123, 'password9', EMPTY_BLOB());
INSERT INTO Usuario (Nick, apellido, nombre, CP, Pwd, foto) VALUES
('Usuario10', 'Moore', 'Thomas', 01234, 'password10', EMPTY_BLOB());
```

-- Inserción de datos en la tabla Votar

```
INSERT INTO Votar (Usuario_Nick, Denuncia_codigo, Denuncia_Usuario_Nick,
UpVote, Favorito) VALUES ('Usuario6', 106, 'Usuario7', 0, 0);
INSERT INTO Votar (Usuario_Nick, Denuncia_codigo, Denuncia_Usuario_Nick,
UpVote, Favorito) VALUES ('Usuario7', 107, 'Usuario8', 1, 1);
INSERT INTO Votar (Usuario_Nick, Denuncia_codigo, Denuncia_Usuario_Nick,
UpVote, Favorito) VALUES ('Usuario8', 108, 'Usuario9', 0, 0);
INSERT INTO Votar (Usuario_Nick, Denuncia_codigo, Denuncia_Usuario_Nick,
UpVote, Favorito) VALUES ('Usuario9', 109, 'Usuario10', 1, 1);
INSERT INTO Votar (Usuario_Nick, Denuncia_codigo, Denuncia_Usuario_Nick,
UpVote, Favorito) VALUES ('Usuario10', 110, 'Usuario6', 0, 0);
```



```
--denuncias publicadas de Usuario
SELECT
    d.Estado,
    d.fecha AS Fecha,
    d.direccion AS Dirección,
    u.CP,
    u.nick AS Usuario
FROM
    Denuncia d
    JOIN Usuario u ON d.Usuario_Nick = u.Nick
ORDER BY
    d.Estado, d.fecha;
--Denuncias mas Votadas
SELECT
    COUNT(*) AS "Nº de votos",
    d.fecha AS Fecha,

    d.direccion AS "Dirección",
    u.CP,
    v.Usuario_Nick AS Usuario
FROM
    Votar v
    JOIN Denuncia d ON v.Denuncia_codigo = d.codigo
    JOIN Usuario u ON d.Usuario_Nick = u.Nick
GROUP BY
    d.codigo, d.fecha, d.direccion, u.CP, v.Usuario_Nick
ORDER BY
    COUNT(*) DESC;

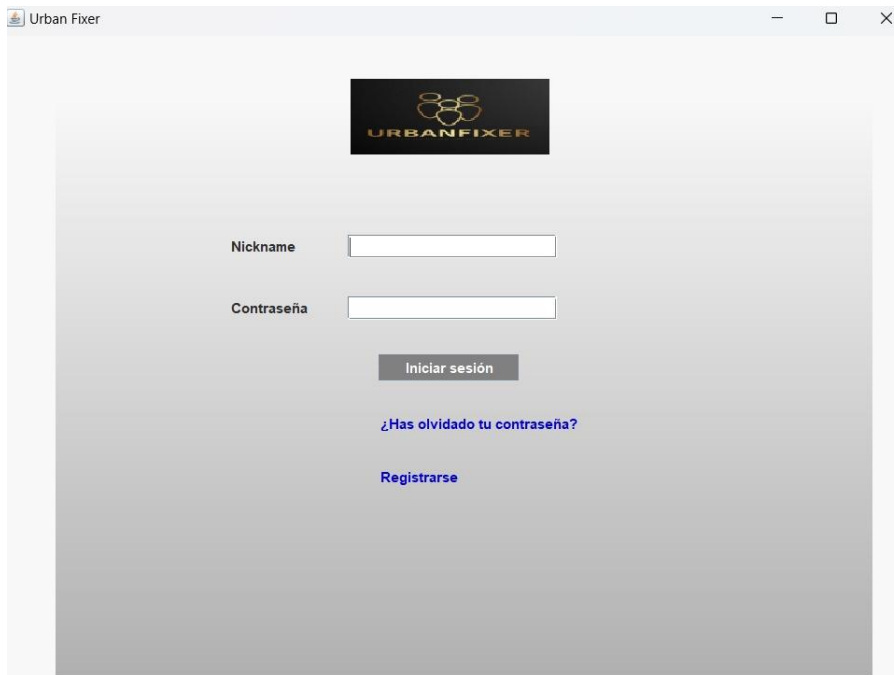
--Mis denuncias
SELECT
    d.fecha AS Fecha,
    d.direccion AS Dirección,
    u.CP,
    d.Estado
FROM
    Denuncia d
    JOIN Usuario u ON d.Usuario_Nick = u.Nick
WHERE
    u.Nick = :Usuario_Nick -- Sustituir :Usuario_Nick por el nick del
usuario logueado
ORDER BY
    d.fecha;
```

```
--1.Gestiones Realizadas
SELECT
    g.Gestion AS Tipo_Gestion,
    d.fecha,
    u.Nick AS Administrador,
    d.codigo AS Cod_Incidencia,
    c.nombre AS Descripcion
FROM Gestionar g
INNER JOIN Usuario u ON g.Usuario_Nick = u.Nick
INNER JOIN Denuncia d ON g.Denuncia_codigo = d.codigo
INNER JOIN Categoria c ON d.codigo = c.Denuncia_codigo
ORDER BY d.fecha;

-- 2.Listado de Usuarios
SELECT
    u.Nick,
    u.nombre AS Nombre,
    u.apellido AS Apellidos,
    u.CP,
    CASE
        WHEN EXISTS (SELECT 1 FROM Gestionar g WHERE g.Usuario_Nick =
u.Nick) THEN 'Si'
        ELSE 'No'
    END AS Admin
FROM
    Usuario u
ORDER BY
    u.Nick;

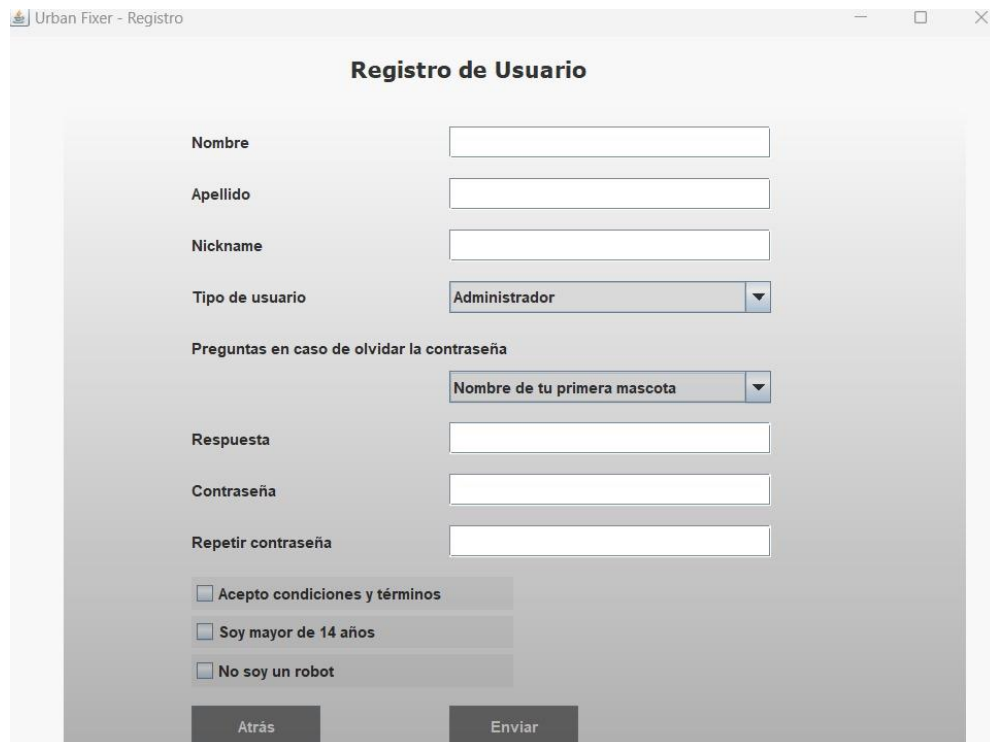
-- 3. Categorías
SELECT Codigo, nombre
FROM Categoria;
```

## Anexo II – Guía de uso de la aplicación



The screenshot shows the login interface of the Urban Fixer application. At the top center is the Urban Fixer logo, which consists of three interlocking rings above the text "URBANFIXER". Below the logo are two input fields: "Nickname" and "Contraseña" (Password). Under the password field is a button labeled "Iniciar sesión" (Log in). Below the button are two links: "¿Has olvidado tu contraseña?" (Forgot your password?) and "Registrarse" (Sign up).

Pantalla de login de Urban Fixer. En la cual el usuario tiene la opción de iniciar sesión o de crear una nueva cuenta pinchando registrarse.



The screenshot shows the registration interface of the Urban Fixer application, titled "Registro de Usuario" (User Registration). It contains several input fields and checkboxes. The fields are: "Nombre" (Name), "Apellido" (Surname), "Nickname", "Tipo de usuario" (User type) with a dropdown menu showing "Administrador", "Preguntas en caso de olvidar la contraseña" (Questions in case of forgetting the password) with a dropdown menu showing "Nombre de tu primera mascota" (Name of your first pet), "Respuesta" (Answer), "Contraseña" (Password), and "Repetir contraseña" (Repeat password). Below these fields are three checkboxes: "Acepto condiciones y términos" (I accept conditions and terms), "Soy mayor de 14 años" (I am over 14 years old), and "No soy un robot" (I am not a robot). At the bottom are two buttons: "Atrás" (Back) and "Enviar" (Send).

## Pantalla de Registro de Usuario

1. Nombre: Introduce tu nombre.
2. Apellidos: Introduce tus apellidos.
3. Nickname: Introduce un apodo único.
4. Tipo de Usuario: Selecciona el tipo de usuario (Administrador o Usuario).
5. Pregunta de Seguridad: Selecciona una pregunta de seguridad de la lista.
6. Respuesta: Introduce la respuesta a la pregunta de seguridad seleccionada.
7. Contraseña: Introduce una contraseña segura.
8. Repetir Contraseña: Vuelve a introducir la contraseña para confirmarla.



### Estado Denuncias



Codigo Pos...	Nombre	Categoria	Estado	Descripcion	Imagen	Favorita
28921	Asfalto roto	Grave	En proceso	El asfalto e...		No
28002	Problema 2	Normal	Solucionada	Descripción...		Sí
28003	Problema 3	Grave	En proceso	Descripción...		No
28004	Problema 4	Muy grave	Solucionada	Descripción...		Sí
28005	Problema 5	Leve	En proceso	Descripción...		No
28006	Problema 6	Normal	Solucionada	Descripción...		Sí
28007	Problema 7	Grave	En proceso	Descripción...		No
28008	Problema 8	Muy grave	Solucionada	Descripción...		Sí
28009	Problema 9	Leve	En proceso	Descripción...		No
28010	Problema 10	Normal	Solucionada	Descripción...		Sí
28011	Problema 11	Grave	En proceso	Descripción...		No
28012	Problema 12	Muy grave	Solucionada	Descripción...		Sí
28013	Problema 13	Leve	En proceso	Descripción...		No
28014	Problema 14	Normal	Solucionada	Descripción...		Sí
28015	Problema 15	Grave	En proceso	Descripción...		No
28016	Problema 16	Muy grave	Solucionada	Descripción...		Sí
28017	Problema 17	Leve	En proceso	Descripción...		No
28018	Problema 18	Normal	Solucionada	Descripción...		Sí
28019	Problema 19	Grave	En proceso	Descripción...		No
28020	Problema 20	Muy grave	Solucionada	Descripción...		Sí
28010	Problema 10	Normal	Solucionada	Descripción...		Sí
28011	Problema 11	Grave	En proceso	Descripción...		No
28012	Problema 12	Muy grave	Solucionada	Descripción...		Sí
28013	Problema 13	Leve	En proceso	Descripción...		No
28014	Problema 14	Normal	Solucionada	Descripción...		Sí
28015	Problema 15	Grave	En proceso	Descripción...		No

Denunciar Problema

Denuncias Favoritas

La **pantalla de Estado de Denuncia** permite a los usuarios y administradores ver el estado actual de las denuncias enviadas. Acceso a la Pantalla de Estado de Denuncia Para acceder a esta pantalla, selecciona la opción Estado de Denuncias en el menú principal.

Componentes de la Pantalla de Estado de Denuncia

1. Bienvenido: Mensaje de bienvenida a la pantalla de Usuarios.
2. Ajustes de Usuario: Botón que permite acceder a la configuración de la cuenta de usuario.
3. Estado de Denuncia: Título de la sección.
4. Tabla de Estado de Denuncias: o Código Postal: Código postal donde se encuentra el problema. o Categoría: Categoría del problema denunciado. o Estado: Estado actual de la denuncia (En proceso, Solucionada, etc.). o Imagen: Foto del problema. o Favorita: Indica si la denuncia es una de tus favoritas.
5. Denuncias Favoritas: Botón para acceder a la lista de denuncias favoritas.
6. Denunciar Problema: Botón para iniciar una nueva denuncia.

The screenshot shows the 'Denunciar Problema' form. On the left is a sidebar with the 'URBANFIXER' logo and two buttons: 'Estado Denuncias' and 'Denuncias Favoritas'. The main area has the title 'Denunciar Problema' and a user icon. The form fields are: 'Nombre del problema:' with a text input; 'Foto del problema:' with a text input; 'Código postal:' with a text input; 'Categoría:' with a dropdown menu showing 'Leve'; and 'Descripción del problema:' with a large text area. At the bottom right is a button labeled 'Enviar Denuncia'.

### Registro de una Denuncia

1. Nombre del Problema: Escribe "Bache en la calle principal".
2. Foto del Problema: Escribe una descripción como "Foto del bache".
3. Código Postal: Introduce "08001".
4. Categoría: Selecciona "Grave".

5. Descripción del Problema: Escribe "Hay un gran bache en la calle principal que causa problemas de tráfico". Una vez llenados todos los campos, haz clic en Enviar Denuncia. Si todos los campos están correctos, se te mostrará una confirmación de que la denuncia ha sido enviada correctamente.

The screenshot shows the 'URBANFIXER' administrator interface. At the top, it says 'Bienvenido a la pantalla de administradores'. On the left sidebar, there are buttons for 'Estado Denuncias', 'Denuncias Favoritas', and 'Denunciar Problema'. The main area is titled 'Denuncias pendientes de autorizar'. It features a dropdown menu set to 'Leves', a search input field, and a 'Buscar' button. Below this is a table with the following columns: 'Codigo Postal', 'Categoria', 'Descripción', 'imagen', and 'Favorita'. The table contains four rows with placeholder text 'Codigo Postal 1' through 'Codigo Postal 4'. At the bottom of the interface, there are two buttons: 'Denegar' and 'Autorizar'.

Codigo Postal	Categoria	Descripción	imagen	Favorita
Codigo Postal 1				
Codigo Postal 2				
Codigo Postal 3				
Codigo Postal 4				

### Componentes de la **Pantalla de Administración**

1. Bienvenida: Mensaje de bienvenida a la pantalla de administradores.
2. Ajustes Administrador: Botón que permite acceder a la configuración del administrador.
3. Denuncias pendientes de autorizar:
  - o Categoría: Selector de categoría de denuncias.
  - o Campo de búsqueda: Permite buscar denuncias por código postal.
  - o Botón de búsqueda: Ejecuta la búsqueda de denuncias.
4. Tabla de Denuncias Pendientes:
  - o Código Postal: Código postal donde se encuentra el problema denunciado.
  - o Categoría: Categoría del problema.
  - o Descripción: Descripción detallada del problema.
  - o Imagen: Representación visual del problema.
  - o Favorita: Indicador si la denuncia es marcada como favorita.

5. Botones de Acción: o Denegar: Rechaza la denuncia seleccionada. o Autorizar: Aprueba la denuncia seleccionada.
6. Otras Funcionalidades: o Denuncias Favoritas: Permite ver las denuncias marcadas como favoritas. o Denunciar Problema: Permite iniciar una nueva denuncia desde la perspectiva administrativa.

### Anexo III

Captura de pantalla de registro, posteriormente corregida, realizada con WireframeSketcher.

Captura de pantalla del prototipo de la pantalla de Usuarios, realizada con WireframeSketcher.



Fragmento de la clase `_00_Login`, que establece los intentos de inicio de sesión con dos condiciones para que tengas tres intentos para login.:

```
private void intentarLogin() {
    String NICK = txtNickname.getText();
    String PWD = new String(txtPassword.getPassword());

    if (modelo.verificarCredenciales(NICK, PWD)) {
        JOptionPane.showMessageDialog(this, "Login exitoso");
        controlador.cambiarVentana(0, 3);
        dispose();
    } else {
        intentos++;
        if (intentos < 3) {
            JOptionPane.showMessageDialog(this, "Usuario o contraseña
incorrectos. Intentos restantes: " + (3 - intentos));
        } else {
            JOptionPane.showMessageDialog(this, "Ha superado el número
de intentos. El programa se cerrará.");
            System.exit(0);
        }
    }
}
```

Fragmento de la clase `_01_Registro`, el cual obliga a que el usuario no deje los campos vacíos y por otro lado, la repetición del campo contraseña.

```
public void actionPerformed(ActionEvent e) {
    String nombre = txtNombre.getText();
    String apellidos = txtApellidos.getText();
    String nickname = txtNickname.getText();
    String tipoUsuario = (String)
comboTipoUsuario.getSelectedItem();
    String pregunta = (String)
comboPreguntas.getSelectedItem();
    String respuesta = txtRespuesta.getText();
    String password = new String(txtPassword.getPassword());
    String passwordRepeat = new
String(txtPasswordRepeat.getPassword());

    // Verificación de campos vacíos
    if (nombre.isEmpty() || apellidos.isEmpty() ||
nickname.isEmpty() ||
    tipoUsuario.isEmpty() || pregunta.isEmpty() ||
respuesta.isEmpty() ||
password.isEmpty() || passwordRepeat.isEmpty()) {
        JOptionPane.showMessageDialog(null, "Todos los campos
deben estar llenos.");
        return;
    }

    // Verificación de que las contraseñas coincidan
    if (!password.equals(passwordRepeat)) {
        JOptionPane.showMessageDialog(null, "Las contraseñas no
coinciden.");
        return;
    }
}
```

```

    }

    // Verificar si el usuario ya existe
    if (controlador.verificarUsuarioExistente(nickname)) {
        JOptionPane.showMessageDialog(null, "El nickname ya
está en uso.");
        return;
    }

    // Llamar al método del controlador para guardar en la base
de datos
    if (controlador.registrarUsuario(nickname, apellidos,
nombre, password, tipoUsuario, respuesta, passwordRepeat)) {
        JOptionPane.showMessageDialog(null, "Usuario registrado
exitosamente.");
    } else {
        JOptionPane.showMessageDialog(null, "Error al registrar
el usuario.");
    }
    controlador.cambiarVentana(1, 0);
}
});

```

Fragmento de la clase `_02_OlvidoPwd`, el cual muestra el selector para elegir la pregunta secreta para restablecer contraseña.

```

public void actionPerformed(ActionEvent e) {
    String selectedQuestion = (String)
comboBoxQuestions.getSelectedItem();
    String answer = textFieldAnswer.getText();
    controlador.cambiarVentana(2, 9);

    String expectedAnswer = "";
    switch (selectedQuestion) {
        case "Nombre de tu primera mascota":
            expectedAnswer = "expectedAnswer1";
            break;
        case "¿Qué mote tenías de pequeño?":
            expectedAnswer = "expectedAnswer2";
            break;
        case "¿Cuál es tu comida favorita?":
            expectedAnswer = "expectedAnswer3";
            break;
        case "País al que te gustaría viajar":
            expectedAnswer = "expectedAnswer4";
            break;
        case "Ciudad de nacimiento":
            expectedAnswer = "expectedAnswer5";
            break;
    }
}
});

```