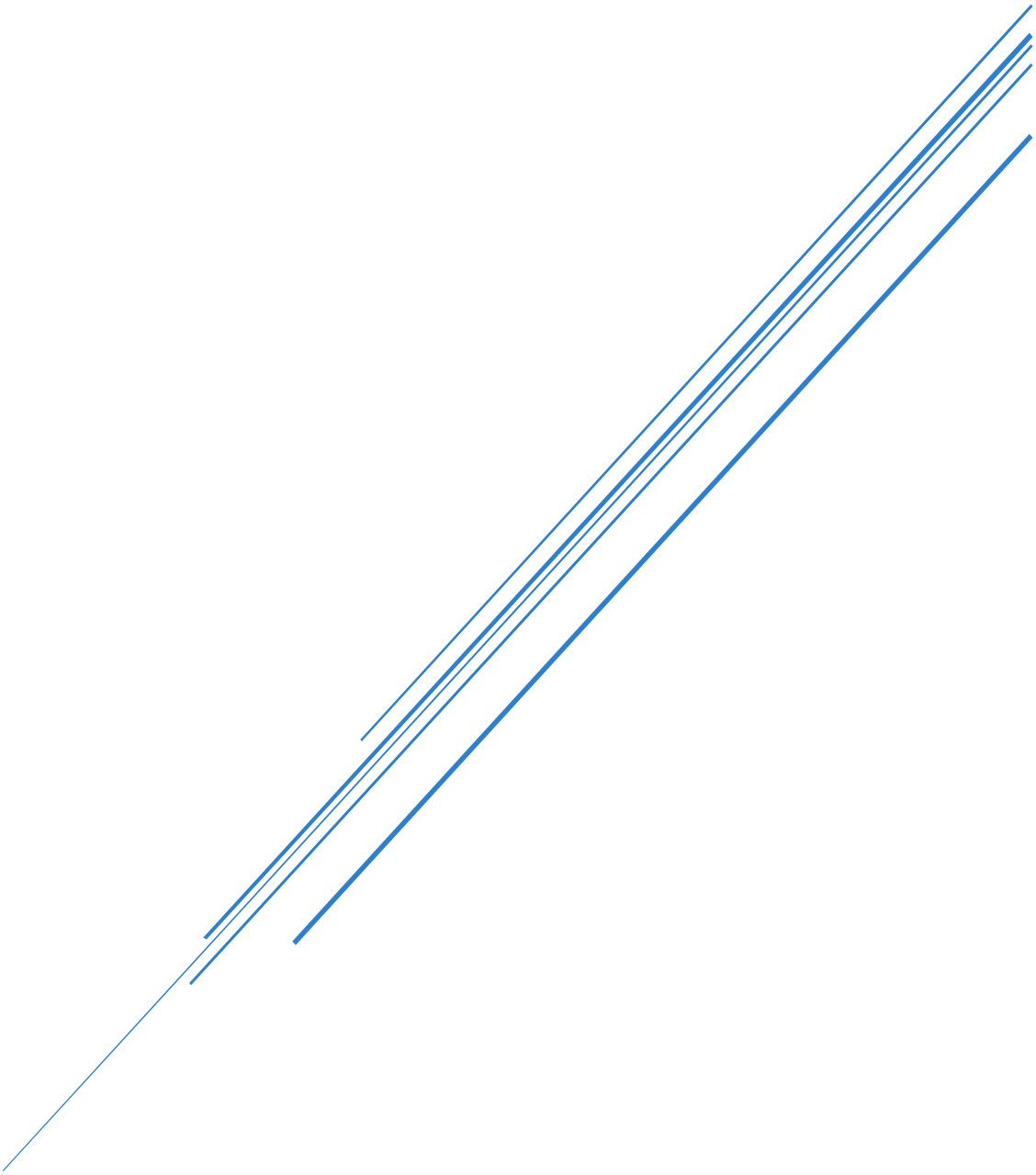
SHELTERLY

Gestor de refugios.



UNIR
Desarrollo de aplicaciones multiplataforma
Alberto Zaragosi Tenorio

Indice

1. Introducción del proyecto

Este proyecto de fin de ciclo de Desarrollo de Aplicaciones Multiplataforma es una aplicación de gestión y visualización de ciertas funcionalidades de un refugio de animales. La aplicación desarrollada, denominada "Shelterly", implementa funcionalidades específicas tanto para empleados como para administradores del refugio, facilitando la gestión eficiente de los datos y procesos relacionados con el cuidado y adopción de animales.

La elección del tema de este proyecto se ha basado en varias premisas. La primera es que el tema permitía una gran flexibilidad a la hora de escalar el proyecto, pudiendo añadir nuevas características en función del tiempo disponible para el desarrollo del mismo. La segunda premisa es que el desarrollo de aplicaciones para la gestión de refugios de animales tiene, en opinión del alumno, una alta aplicación laboral en el sector de la informática, ya que la gestión de datos y la organización son cruciales en este ámbito. La tercera y última premisa es que esta aplicación permitía demostrar los conocimientos que el alumno ha adquirido durante el ciclo formativo, tanto en el desarrollo del mismo con Java como en la gestión de bases de datos con MySQL y la implementación de funcionalidades lógicas y administrativas.

2. Objetivos del proyecto

En el proyecto desarrollado, se han implementado las siguientes funcionalidades, dirigidas tanto a empleados como a administradores del refugio de animales "Shelterly":

Funcionalidades para Empleados:

2.1 Gestión de Animales:

a. Visualización de Animales: Permite a los empleados ver una tabla con los animales disponibles, los que han sido dados de baja, o todos los animales en el refugio. Los usuarios pueden alternar entre estas vistas utilizando botones específicos.

b. Dar de Alta a un Animal: Facilita el registro de nuevos animales en el sistema mediante un formulario que recoge información relevante como nombre, especie, raza, edad, peso y fecha de ingreso.

c. Modificar Datos de un Animal: Permite actualizar la información de los animales existentes en el refugio a través de una ventana específica donde se pueden editar los datos previamente registrados.

d. Dar de Baja a un Animal: Permite registrar la salida de un animal del refugio, ya sea por adopción o fallecimiento. En caso de adopción, se solicitan los datos del adoptante.

2.2 Gestión de Vacunas:

a. Administración de Vacunas: Abre una ventana específica para gestionar las vacunas de los animales seleccionados. Los empleados pueden ver un historial de las vacunas administradas, la fecha de la última vacunación, la próxima fecha programada y la duración de cada inmunización.

2.3 Gestión de Adoptantes:

a. Visualización y Modificación de Adoptantes: Facilita la gestión de los datos de las personas que han adoptado animales, incluyendo el nombre, correo electrónico, número de teléfono y número de animales adoptados. Los empleados pueden modificar esta información según sea necesario.

b. Visualización de Animales Adoptados: Permite ver los detalles de los animales adoptados por cada persona, proporcionando un historial completo de adopciones.

2.4 Sesión de Usuario:

a. Login y Registro: Los empleados pueden iniciar sesión con sus credenciales o registrarse si no tienen una cuenta, lo que facilita la incorporación de nuevo personal.

b. Logout: Los empleados pueden cerrar sesión y regresar a la pantalla de inicio de sesión.

Funcionalidades Exclusivas para Administradores:

2.5 Gestión de Usuarios:

a. Visualización de Usuarios: Los administradores pueden ver una tabla con todos los usuarios registrados, incluyendo sus datos y roles.

b. Dar de Alta a un Empleado: Permite crear nuevas cuentas de usuario para empleados, con la opción de asignarles un rol de administrador.

c. Modificar Datos de Usuarios: Facilita la actualización de la información de los usuarios existentes.

d. Dar de Baja a un Empleado: Permite eliminar cuentas de usuario cuando un empleado deja de formar parte del refugio.

3. Módulos Formativos Aplicados en el FTG

Durante el desarrollo de este TFG, se han aplicado los módulos formativos que se detallan a continuación:

3.1 Programación

Los conocimientos en esta asignatura han sido, como no podía ser de otra forma, el eje central en el desarrollo del proyecto. Lenguajes de programación, programación estructurada y programación modular han sido fundamentales para poder abordar este TFG. La aplicación "Shelterly" se ha desarrollado utilizando Java, un lenguaje que permite la creación de aplicaciones robustas y escalables. Además, se ha implementado el paradigma de programación orientada a objetos, lo que ha facilitado la organización del código y la reutilización de componentes. La estructuración modular del código ha permitido mantener una separación clara entre las diferentes funcionalidades de la aplicación, mejorando así su mantenibilidad y escalabilidad.

3.2 Base de Datos

Para el diseño de la base de datos, se ha utilizado MySQL, un sistema de gestión de bases de datos relacional que permite manejar grandes volúmenes de datos de manera eficiente. Se han creado varias tablas para almacenar información crítica del refugio, como los datos de los animales, usuarios, vacunas y adoptantes. El diseño de la base de datos ha seguido principios de normalización para minimizar la redundancia y asegurar la integridad de los datos. Las relaciones entre las tablas han sido cuidadosamente planificadas para reflejar las interacciones reales entre los diferentes elementos del refugio, como la relación entre animales y adoptantes, y entre animales y vacunas.

3.3 Acceso a Datos

La interacción de la aplicación con los datos almacenados en la base de datos se ha realizado mediante JDBC (Java Database Connectivity). Se ha implementado un conjunto de clases y métodos para realizar operaciones CRUD (Crear, Leer, Actualizar y Borrar) sobre las diferentes tablas de la base de datos. El uso de prepared statements ha sido clave para prevenir inyecciones SQL y asegurar la seguridad de la aplicación.

3.4 DevOps

En el desarrollo de este proyecto, se han aplicado principios de DevOps para mejorar la eficiencia y la calidad del desarrollo del software. Se ha utilizado GitHub como sistema de control de versiones, lo que ha permitido llevar un seguimiento detallado de los cambios en el código, facilitando la colaboración y el manejo de versiones.

4. Herramientas y Lenguajes utilizados

4.1 Apache NetBeans

Apache NetBeans es un entorno de desarrollo integrado (IDE) de código abierto que facilita la programación en diversos lenguajes, especialmente Java. NetBeans ofrece un conjunto robusto de herramientas para el desarrollo de aplicaciones, incluyendo edición de código, depuración, pruebas y refactorización. En el desarrollo de "Shelterly", NetBeans se utilizó debido a su compatibilidad con Java y JavaFX, así como por su facilidad para integrar diversas bibliotecas necesarias para el proyecto.

4.2 MySQL

MySQL es un sistema de gestión de bases de datos relacional de código abierto. Es conocido por su fiabilidad, rendimiento y facilidad de uso. Para "Shelterly", MySQL ha sido empleado para almacenar y gestionar todos los datos relacionados con el refugio, como información sobre animales, empleados, adoptantes y registros de vacunación. La estructura de la base de datos ha sido diseñada para garantizar integridad y eficiencia en las operaciones.

4.3 JavaFX

JavaFX es una plataforma para el desarrollo de aplicaciones de escritorio y web con interfaces de usuario enriquecidas. Sustituye a Swing como la principal biblioteca gráfica en Java, proporcionando herramientas modernas para la creación de interfaces interactivas. En "Shelterly", JavaFX se ha utilizado para construir las diversas ventanas y componentes visuales, ofreciendo una experiencia de usuario intuitiva y atractiva.

@FXML es una anotación utilizada en el desarrollo de aplicaciones con JavaFX. Esta anotación marca métodos y campos en el controlador para que puedan ser referenciados en archivos FXML, que definen la interfaz de usuario de la aplicación.

```java
@FXML
private TableColumn<animal, Integer> idColumnPersona;

@FXML
private TableColumn<animal, String> nameColumnPersona;

@FXML
private Label nombre;
```

Ilustración 1: Ejemplo etiqueta @FXML

4.4 MySQL Workbench

MySQL Workbench es una herramienta visual para diseñadores, desarrolladores y administradores de bases de datos MySQL. Permite modelar datos, diseñar bases de datos y realizar consultas SQL. En el desarrollo de "Shelterly", Workbench ha sido fundamental para la creación y gestión de la estructura de la base de datos, así como para realizar pruebas y optimizaciones de consultas SQL.

4.5 Diagrams.net

Diagrams.net, anteriormente conocido como draw.io, es una herramienta en línea para la creación de diagramas de flujo, esquemas y modelos de datos. Esta herramienta ha sido utilizada para diseñar el diagrama del modelo de datos de "Shelterly", asegurando una representación clara y precisa de las relaciones entre las distintas entidades de la base de datos.

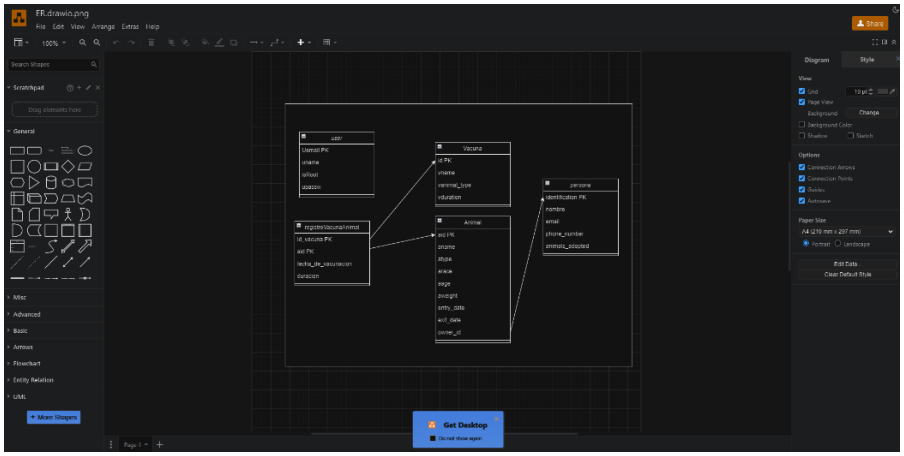


Ilustración 2: Uso de Diagrams.net

4.6 GitHub

GitHub es una plataforma de alojamiento para proyectos de desarrollo de software que utiliza el sistema de control de versiones Git. Proporciona herramientas para la colaboración y el seguimiento de cambios en el código fuente. Durante el desarrollo de "Shelterly", GitHub ha sido empleado para gestionar las versiones del código, facilitando la colaboración y asegurando la integridad y la historia del desarrollo del proyecto.

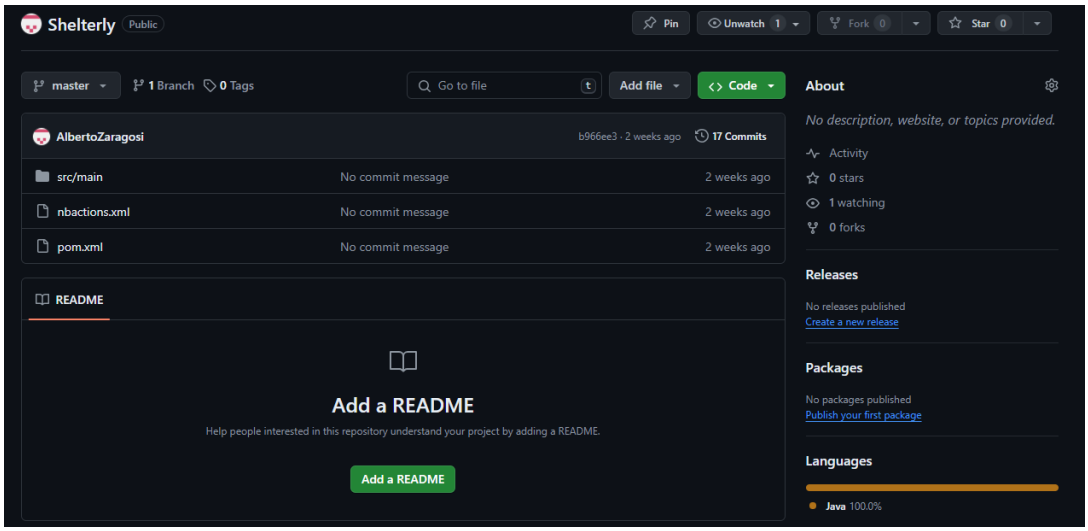


Ilustración 3: Proyecto Github

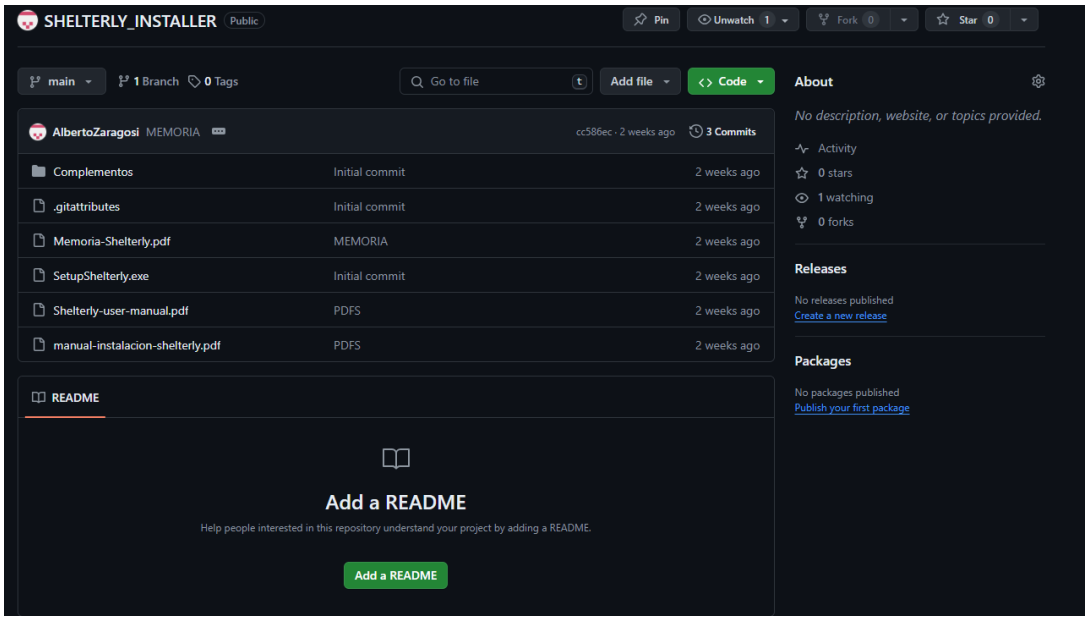


Ilustración 4: Github del TFG

4.7 WAMP

WAMP es un paquete de software que proporciona Apache, MySQL y PHP para sistemas Windows. En "Shelterly", WAMP se ha utilizado para configurar un entorno de desarrollo local, permitiendo la conexión entre la aplicación de escritorio y la base de datos MySQL alojada localmente. Esta configuración ha facilitado las pruebas y el desarrollo de la aplicación sin necesidad de un servidor remoto.

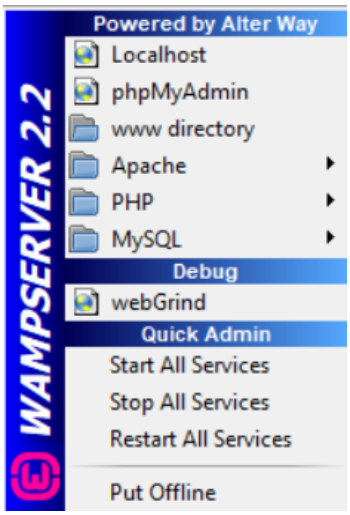


Ilustración 5: Iniciando Wampp

4.9 JDBC

Java Database Connectivity (JDBC) es una API que permite a los programas Java conectarse y ejecutar consultas en bases de datos. JDBC ha sido crucial en "Shelterly" para establecer la comunicación entre la aplicación de escritorio y la base de datos MySQL, permitiendo realizar operaciones CRUD (crear, leer, actualizar, eliminar) de manera eficiente.

4.10 Maven

Maven es una herramienta de gestión y comprensión de proyectos utilizada en la gestión de dependencias y la automatización del proceso de construcción. En "Shelterly", Maven ha sido empleado para gestionar las bibliotecas necesarias para el desarrollo, asegurando que todas las dependencias estén actualizadas y correctamente configuradas.

4.11 Scene Builder

Scene Builder es una herramienta de diseño visual que permite crear interfaces gráficas de usuario para aplicaciones JavaFX. Facilita el diseño de escenas mediante la manipulación directa de componentes, sin necesidad de escribir código manualmente. En "Shelterly", Scene Builder ha sido utilizado para diseñar las interfaces de usuario, mejorando la eficiencia del desarrollo y asegurando una experiencia de usuario cohesiva y atractiva.

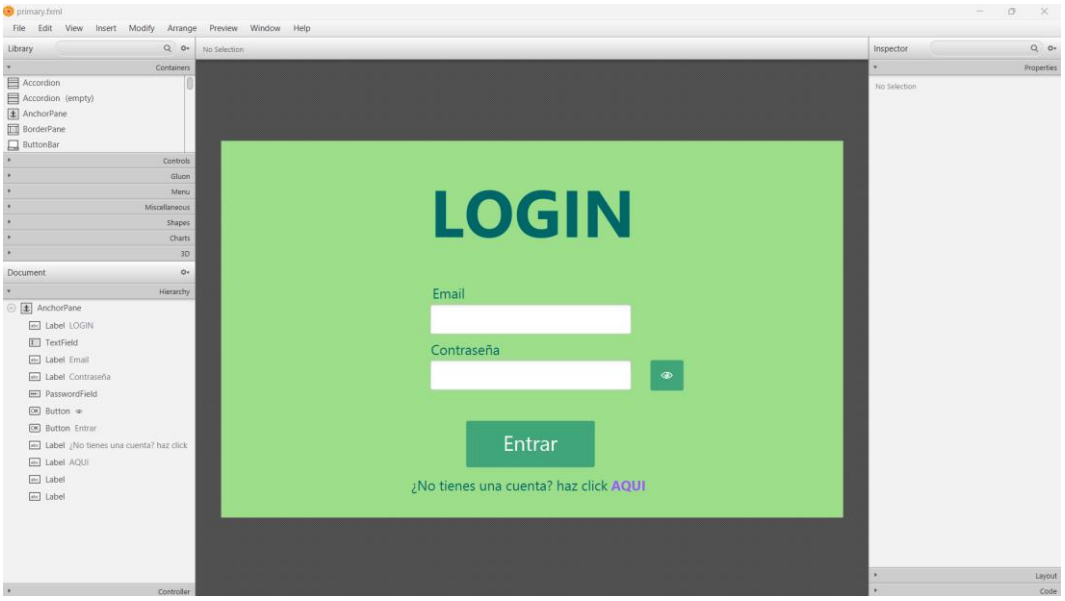


Ilustración 6: Menu Login usando Scene Builder

5. Fases del proyecto

5.1 Idea

Al momento de decidir, el estudiante consideró que quería desarrollar una aplicación que le permitiera adquirir conocimientos y habilidades valiosas para su futuro profesional. Tras evaluar varias opciones, decidió enfocarse en una aplicación de gestión para un refugio de animales, ya que este tipo de proyecto no solo le ofrecería una experiencia técnica significativa sino también una conexión emocional y social importante.

Además, la elección de tecnologías como Java, JavaFX, MySQL y el entorno WAMP, fue estratégica. Estas herramientas son ampliamente utilizadas en la industria del software, proporcionando una base sólida para el desarrollo de aplicaciones empresariales. La decisión de utilizar estas tecnologías no solo permitió al estudiante mejorar sus habilidades técnicas en programación y gestión de bases de datos, sino también asegurarse de que la aplicación Shelterly fuera escalable y mantenible en el futuro.

El diseño de la aplicación se pensó de manera modular, permitiendo agregar funcionalidades adicionales según el progreso del proyecto y el tiempo disponible. Inicialmente, se planteó desarrollar funcionalidades básicas como el registro y login de usuarios, la gestión de animales y la administración de adopciones. A medida que el proyecto avanzaba, se fueron añadiendo funcionalidades más complejas, como la gestión de vacunaciones y la administración de empleados por parte de los usuarios con roles de administrador.

Esta estructura modular no solo facilitó la planificación y ejecución del proyecto, sino que también permitió abordar problemas complejos de manera incremental. Cada módulo fue una oportunidad para aprender y aplicar nuevas técnicas de desarrollo y diseño de software, asegurando así un aprendizaje profundo y significativo a lo largo del desarrollo del TFC.

5.2 Primera aproximación

Para definir los primeros requisitos del proyecto, el estudiante se inspiró en su experiencia laboral en una cafetería, donde el sistema de gestión era intuitivo y basado en botones que abrían distintas ventanas. Esta interacción sencilla y directa le pareció ideal para la gestión de un refugio de animales, asegurando una experiencia de usuario eficiente y amigable.

El primer paso fue identificar los elementos clave que la aplicación debía gestionar. En un refugio de animales, los protagonistas principales son los animales y los empleados que los cuidan. Se decidió que la aplicación debería permitir el registro y gestión de ambos. Los animales debían tener un registro detallado con información sobre su tipo, raza, edad, peso, y fechas de entrada y salida del refugio.

También se consideró esencial incluir la capacidad de gestionar las vacunaciones de los animales. Se diseñó una funcionalidad específica para registrar y administrar las vacunaciones, permitiendo a los empleados ver y actualizar esta información fácilmente

Otro aspecto importante fue la gestión de las adopciones. El sistema debía registrar información sobre los adoptantes, incluyendo sus datos de contacto y el número de animales adoptados. Además, se pensó en la necesidad de gestionar la baja de animales, ya fuera por adopción o por fallecimiento.

Se consideraron dos tipos de usuarios: empleados y administradores. Los administradores tendrían acceso a funcionalidades adicionales, como la gestión de usuarios, permitiendo crear, modificar y eliminar cuentas de empleados.

Para garantizar que la aplicación fuera fácil de usar, se decidió implementar un diseño modular con botones que abrieran ventanas específicas para cada tarea. Esta aproximación facilitó el desarrollo y pruebas del software, ya que cada módulo podía ser desarrollado y revisado de forma independiente.

5.3 Modelo de datos

Con las premisas del punto anterior, se diseñó el siguiente modelo de datos. Este diagrama ("Diagrama_Shelterly.png").

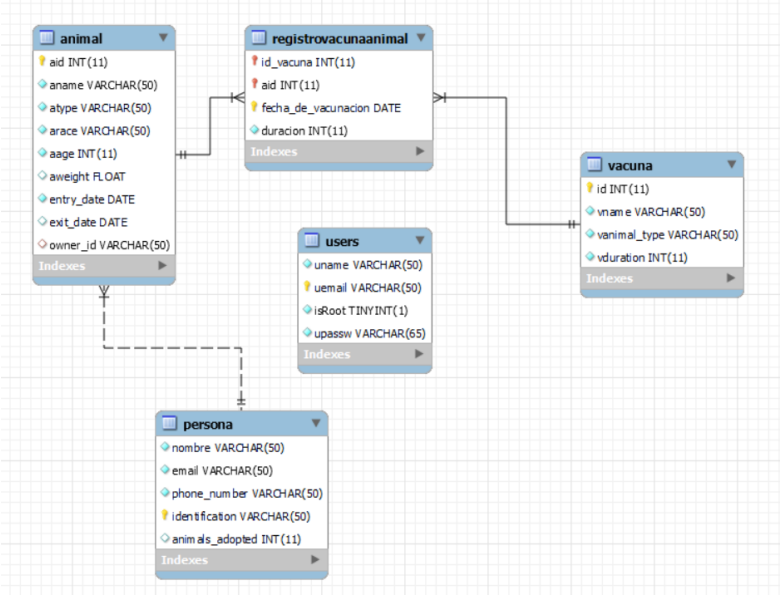


Ilustración 7: Diagrama de clases

5.4 Base de datos

Para la creación de la base de datos, se ha utilizado MySQL Workbench, una herramienta de diseño y administración de bases de datos MySQL. MySQL Workbench permite una gestión visual y una creación de esquemas más eficiente, lo cual ha facilitado la configuración de todas las tablas y relaciones necesarias para la aplicación “Shelterly”.

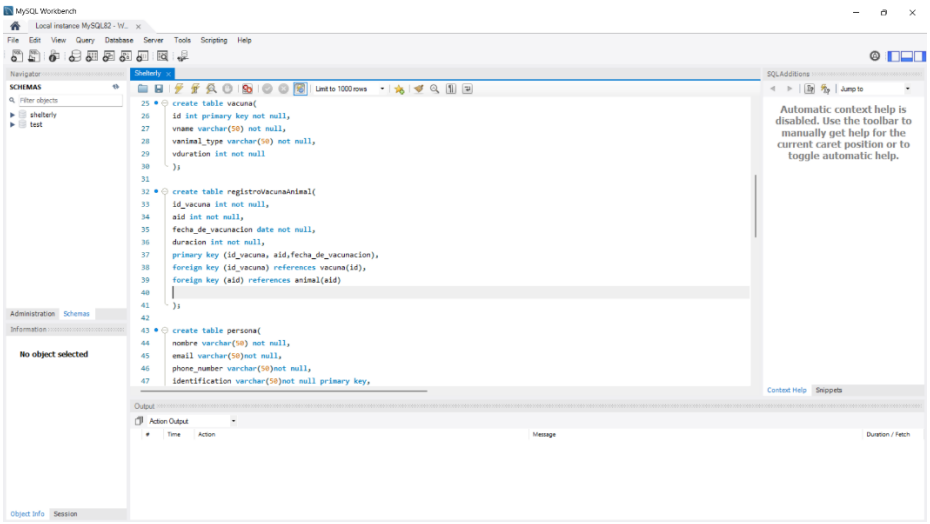


Ilustración 8: MYSQL Workbench editando Shelterly.sql

La base de datos del proyecto "Shelterly" se ha diseñado utilizando MySQL como sistema de gestión de bases de datos. A continuación se detallan las principales características y elementos utilizados en su estructura:

Tablas: Se han definido varias tablas para almacenar los datos esenciales del sistema, como usuarios, animales, vacunas, registro de vacunaciones y personas que han adoptado animales.

Claves primarias y foráneas: Cada tabla incluye claves primarias para identificar de manera única cada registro. Se han establecido claves foráneas para establecer relaciones entre las tablas y garantizar la integridad referencial de los datos.

Índices: Se han utilizado índices para optimizar la velocidad de las consultas y asegurar que no haya duplicidades en los datos clave, mejorando así la eficiencia y consistencia de las operaciones en la base de datos.

5.5 Modelo, Vista y Controlador (MVC)

En el desarrollo de la aplicación Shelterly, se ha adoptado el patrón Modelo-Vista-Controlador (MVC) para estructurar y organizar el código de manera eficiente y modular. A continuación se detalla cómo se implementa este patrón en el contexto particular del proyecto:

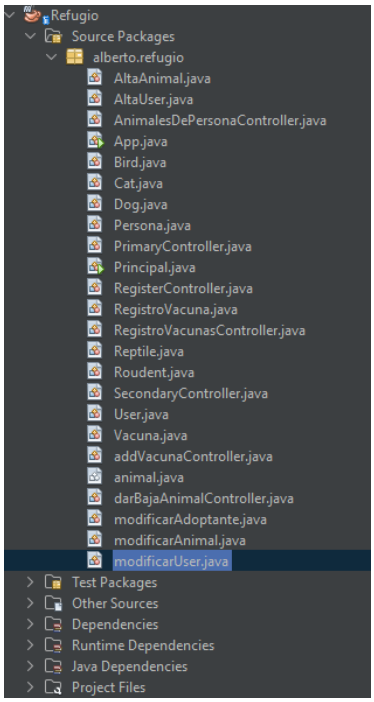


Ilustración 9: Clases .java

5.5.1 Modelo

Dentro del paquete alberto.refugio, el modelo comprende las clases Java que representan las entidades fundamentales del sistema. Estas clases encapsulan los datos y la lógica relacionada con las operaciones sobre esos datos, siguiendo un diseño orientado a objetos.

5.5.1.1 Clases de Modelo

Las clases de modelo en Shelterly han sido creadas manualmente para adaptarse específicamente a las necesidades del proyecto. Cada clase modela una entidad principal, como Persona, Animal, entre otras, y contiene los atributos correspondientes (nombre, email, número de teléfono, etc.) junto con sus respectivos métodos getters y setters.

5.5.2 Vista

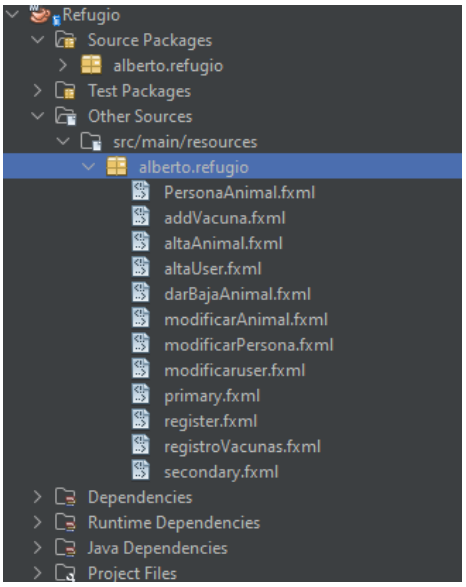


Ilustración 10: Archivos .fxml

La vista en Shelterly está implementada utilizando JavaFX, con archivos fxml que describen la estructura y el diseño de las interfaces de usuario. Estos se encuentran en otra carpeta en el directorio "src/main/resources"

5.5.2.1 Archivos FXML

Los archivos fxml definen la disposición y los elementos visuales de las pantallas de la aplicación. Cada archivo está asociado a un controlador que gestiona las interacciones del usuario y la comunicación con la lógica de negocio a través de los servicios y métodos disponibles en el modelo.

5.5.3 Controlador

Los controladores en Shelterly actúan como intermediarios entre la vista y el modelo, facilitando la interacción entre ambos componentes y asegurando que los datos ingresados por el usuario sean procesados de manera adecuada.

5.5.3.1 Implementación de Lógica

Cada controlador en el paquete alberto.refugio está diseñado para manejar eventos de usuario, validar entradas, y coordinar la ejecución de operaciones específicas del negocio. Estos controladores se encargan directamente de la manipulación de datos a través de consultas SQL, asegurando la persistencia de la información en la base de datos MySQL.

Por ejemplo:

```
public void modificar() {

    if (!nombre.getText().equalsIgnoreCase(anotherString: "") && !nombre.getText().equalsIgnoreCase(anotherString: " ")
            && !email.getText().equalsIgnoreCase(anotherString: "") && !email.getText().equalsIgnoreCase(anotherString: " ")) {
        try {
            // Carga el controlador JDBC
            Class.forName(className: "com.mysql.cj.jdbc.Driver");

            // Establece la conexión con la base de datos
            String url = "jdbc:mysql://localhost:3307/shelterly";
            String usuario = "root";
            String contraseña = "";
            Connection conexion = DriverManager.getConnection(url, user: usuario, password: contraseña);

            Statement stmt = conexion.createStatement();

            String query = "UPDATE users set uname ='" + nombre.getText() + "' , isRoot = " + root.isSelected() + " WHERE uemail = '" + email.getText() + "';";
            stmt.executeUpdate(sql: query);
            stmt.close();
            conexion.close();
            App.setRoot(fxml: "Secondary");
        } catch (ClassNotFoundException | SQLException ex) {
            ex.printStackTrace();
            // Manejo de excepciones, por ejemplo, mostrar un mensaje de error
        } catch (IOException ex) {
            Logger.getLogger(name: modificarUser.class.getName()).log(level: Level.SEVERE, msg: null, thrown: ex);
        }
    }else{
        errMess.setText(string: "Ningun campo debe estar vacio");
    }

}
```

Ilustración 11: Ejemplo uso de SQL en el proyecto

5.6 Desarrollo en Java

Para la implementación de Shelterly se ha empleado la versión 19 del Software Development Kit (SDK) de Java (JDK 19). Esta elección asegura compatibilidad con las últimas funcionalidades y mejoras del lenguaje, proporcionando un entorno de desarrollo moderno y eficiente.

El uso de Java en Shelterly ha permitido a los desarrolladores aplicar conocimientos adquiridos a lo largo del ciclo formativo, garantizando una base sólida y experiencia práctica en un entorno profesionalmente relevante.

5.7 Desarrollo con FXML y JavaFX

En el desarrollo de la interfaz gráfica de usuario para Shelterly, se optó por utilizar FXML junto con JavaFX. A continuación se destacan las razones principales de esta elección y las características clave:

Integración con JavaFX: FXML se integra perfectamente con JavaFX, proporcionando una forma estructurada y declarativa de definir la interfaz de usuario.

Separación de la Lógica y la Interfaz: FXML separa claramente la lógica de presentación (controladores) de la estructura de la interfaz, facilitando la mantenibilidad y la colaboración en el desarrollo.

Diseño Visual y Herramientas: Permite diseñar interfaces gráficas visualmente utilizando herramientas como Scene Builder, lo que agiliza el desarrollo y mejora la productividad.

Ventajas de FXML sobre Swing: FXML y JavaFX ofrecen ventajas modernas y mejoradas en comparación con Swing ya que JavaFX es la API gráfica más reciente de Java, con un soporte activo y una comunidad robusta que proporciona soluciones a problemas comunes, además al igual que Swing garantiza que la interfaz de usuario se vea y funcione de manera consistente en diferentes sistemas operativos.

5.8 Desarrollo de Interfaces con FXML

El diseño de interfaces en Shelterly se fundamenta en la organización modular y unificada utilizando FXML, lo cual ha sido crucial para gestionar y navegar eficazmente entre las diversas vistas de la aplicación. A continuación se detallan los aspectos clave que han guiado el desarrollo de estas interfaces:

5.8.1 Organización Modular y Gestión de Vistas

Uso Estratégico de FXML y Controladores: Cada funcionalidad principal de la aplicación está encapsulada en un archivo FXML dedicado, estrechamente vinculado a su controlador Java correspondiente. Esta separación clara entre la presentación (FXML) y la lógica de la aplicación (controlador) facilita un desarrollo estructurado y mantenible.

Vista Principal Dinámica y Navegación Contextual: La interfaz principal de Shelterly ofrece múltiples paneles intercambiables, como gestión de usuarios, animales y adoptantes. Estos paneles se muestran dinámicamente según las acciones del usuario, proporcionando una navegación fluida y contextual.

5.8.2 Interacción Intuitiva y Flujo de Trabajo

Navegación Guiada por Menús: Los botones ubicados estratégicamente en el menú principal permiten a los usuarios moverse sin esfuerzo entre diferentes secciones de la aplicación, como la gestión detallada de usuarios y animales. Cada sección está diseñada con operaciones específicas, como altas, bajas y modificaciones, adaptadas para optimizar la productividad del usuario.

Consistencia Visual y Usabilidad: Se ha puesto un énfasis significativo en mantener una interfaz coherente y accesible. Los elementos visuales, como botones y tablas, están diseñados para facilitar la interacción del usuario y la manipulación eficiente de datos, promoviendo así una experiencia de usuario intuitiva.

5.8.3 Flexibilidad y Escalabilidad

Adaptabilidad y Mantenimiento Simplificado: La arquitectura basada en FXML proporciona una plataforma flexible y escalable. Esta flexibilidad es crucial para añadir nuevas funcionalidades o ajustar las existentes de manera eficiente, sin comprometer la integridad del diseño ni la funcionalidad central.

Optimización del Ciclo de Desarrollo: Aprovechando las ventajas del diseño declarativo de FXML y las capacidades robustas de JavaFX, el equipo de desarrollo ha optimizado el ciclo de desarrollo. Esto permite una implementación ágil de mejoras y actualizaciones,

alineando continuamente la aplicación con las necesidades cambiantes del usuario y del mercado.

5.9 Modelo de Datos Utilizado:

Para este proyecto de gestión de refugios de animales, se optó por utilizar una base de datos relacional SQL en lugar de una base de datos NoSQL. Esta elección se fundamenta en las siguientes consideraciones:

5.9.1.1 Estructura de los Datos

Los datos del sistema, como usuarios, animales, personas adoptantes, vacunas y registros de vacunación, tienen relaciones claras y definidas entre sí. Un modelo relacional facilita la representación de estas relaciones mediante tablas, claves primarias y foráneas, lo que asegura la integridad referencial y la consistencia de los datos.

5.9.1.2 Transacciones Complejas

Las operaciones del sistema, como la gestión de usuarios, la adopción y seguimiento de animales, y el registro de vacunas, requieren transacciones complejas que se benefician de las características ACID (Atomicidad, Consistencia, Aislamiento y Durabilidad) proporcionadas por las bases de datos relacionales. Estas características garantizan la fiabilidad y la integridad de las operaciones críticas.

5.9.1.3 Consultas Flexibles

Las consultas en SQL permiten realizar operaciones complejas de manera eficiente, como búsquedas por criterios múltiples, agregaciones y operaciones de JOIN. Esto es especialmente útil para generar informes, estadísticas y análisis de datos dentro del sistema de gestión de refugios de animales.

5.9.1.4 Escalabilidad y Mantenimiento

Aunque las bases de datos NoSQL son altamente escalables horizontalmente y adecuadas para casos de uso con grandes volúmenes de datos no estructurados, en este contexto, la escalabilidad vertical típica de las bases de datos relacionales es suficiente. Además, las bases de datos relacionales son generalmente más maduras en términos de herramientas de administración y soporte, facilitando el mantenimiento a largo plazo del sistema.

5.9.2 Modelo de Datos Representado

A continuación se presenta el diagrama de clases que ilustra las tablas y sus relaciones en el diseño de la base de datos utilizada para el sistema de gestión de refugios de animales:

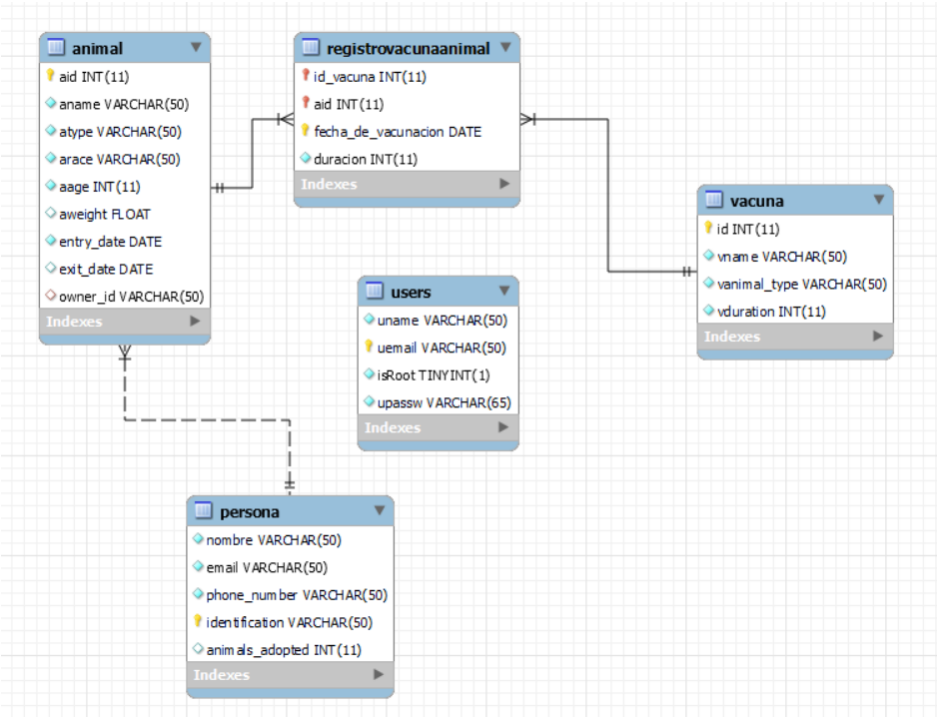


Ilustración 12: Diagrama de clases

Este diagrama muestra visualmente las entidades (users, persona, animal, vacuna, registroVacunaAnimal) junto con sus atributos principales y las relaciones establecidas mediante claves primarias y foráneas. Este diseño de base de datos asegura una gestión integral y eficiente de los datos relacionados con los animales, los adoptantes, las vacunas y los usuarios del sistema.

5.9.3 Diagrama de Casos de Uso

El siguiente diagrama de casos de uso detalla las interacciones principales de los empleados, quienes pueden tener permisos de administrador, dentro del sistema de gestión de refugios de animales. El diagrama incluye funcionalidades como la gestión de animales, vacunas, adoptantes y usuarios, reflejando las acciones específicas disponibles para los empleados y administradores.

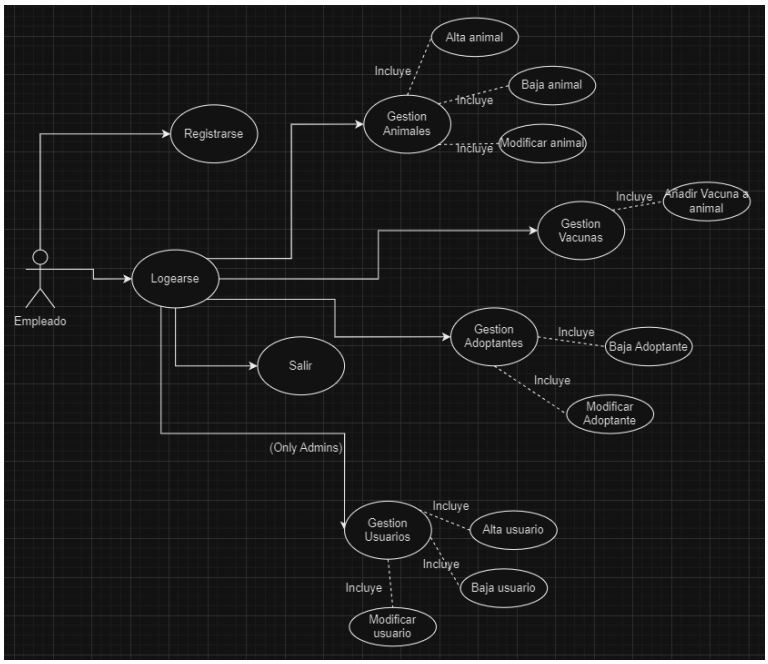


Ilustración 13: Diagrama de caso de uso

Este diagrama de casos de uso proporciona una visión clara de las operaciones permitidas y los actores involucrados en cada caso, facilitando la comprensión del flujo de trabajo y las responsabilidades dentro del sistema.

6. Conclusiones y Mejoras del Proyecto

A lo largo de la realización de este proyecto, se ha adquirido una valiosa combinación de conocimientos técnicos y funcionales, aplicables tanto en el ámbito personal como profesional. En particular, se ha profundizado en el diseño y gestión de bases de datos relacionales utilizando SQL, así como en el desarrollo de sistemas de gestión integral. Este proyecto ha sido una excelente oportunidad para consolidar habilidades en la administración de datos, la creación de interfaces de usuario eficientes y la implementación de funcionalidades críticas para un sistema de gestión de refugios de animales.

El desarrollo del sistema ha permitido al estudiante obtener un entendimiento sólido en el manejo de bases de datos relacionales, la implementación de relaciones entre entidades, y la utilización de claves primarias y foráneas para asegurar la integridad referencial de los datos. Además, se ha mejorado significativamente en la conceptualización y desarrollo de un sistema de gestión que abarca diversas áreas funcionales, como la gestión de animales, adoptantes, vacunas y usuarios. Este conocimiento será invaluable en futuros proyectos que requieran una gestión integral de datos y procesos. Aunque el foco principal no ha sido el diseño de interfaces gráficas, se ha trabajado en la creación de interfaces funcionales y eficaces que permiten una interacción intuitiva por parte de los usuarios del sistema.

Mejoras Propuestas

Aunque el proyecto cumple con los objetivos establecidos inicialmente, se identifican algunas mejoras que podrían implementarse para enriquecer aún más la funcionalidad del sistema:

Exportación de Datos a PDF: Una mejora significativa sería la inclusión de una funcionalidad que permita exportar los datos del sistema a formato PDF, lamentablemente, a falta de tiempo, no ha podido implementarse.

Mejora de la Interfaz Gráfica: Con más tiempo, se podría mejorar la estética y la usabilidad de la interfaz gráfica creando una experiencia de usuario incluso más atractiva e intuitiva.

En resumen, el proyecto ha sido un éxito en términos de aprendizaje y aplicación práctica de conocimientos. Se ha logrado desarrollar un sistema robusto y funcional que satisface las necesidades básicas de gestión de un refugio de animales.

7. Bibliografía

Para la elaboración del proyecto el alumno se ha basado en un 80% en lo aprendido durante su formación en el centro UNIR https://unirfp.unir.net/

Para el resto de información, ha realizado búsquedas a través de paginas como

https://stackoverflow.com/ , https://www.wikipedia.org/ , https://github.com/ , https://www.w3schools.com/ .

Así como documentación oficial como https://openjfx.io/ , https://docs.oracle.com/en/java/javase/19/ , https://dev.mysql.com/doc/ .

8. Anexos

En el mismo directorio donde se encuentra este documento se incluyen también los siguientes anexos:

- Guía de usuario: (Shelterly-user-manual.pdf)
- Manual de instalación: (manual-instalacion-shelterly.pdf)
- Caso de uso: (Use_case.png)
- Diagrama de clases: (Diagrama_Shelterly.png)

Código fuente del proyecto: https://github.com/AlbertoZaragosi/Shelterly