

GESTOR DE HUERTOS DE ‘AUTOCULTIVO’

PFG DAW

I.E.S. CAMP DE MORVEDRE
JUNIO 2021

Alumna: Lidia Baixauli de la Villa
Tutor: Manuel Arnau Huerta

ÍNDICE

0. INTRODUCCIÓN	3
0.1. Módulos a los que implica	3
0.2. Descripción	5
1. ESTUDIO PREVIO.....	6
2. PLAN DE TRABAJO.....	8
2.1. División del trabajo	8
2.2. Estimación de tiempos	8
3. ANÁLISIS	9
3.1 Funcionalidad de la aplicación.....	9
3.2 Requerimientos funcionales	10
3.3 Diagramas UML.....	13
3.3.1 Diagrama de clases	13
3.3.2 Casos de uso	14
3.4 Capturas de las funcionalidades	15
4. DISEÑO.....	18
4.1 Capas de la aplicación	18
5. IMPLEMENTACIÓN	26
5.1. Estructura de la aplicación	26
5.2 Lenguajes utilizados	31
6. DESPLIEGUE	34
6.1 Configuración del proyecto en Netbeans.....	34
6.2 Subir el proyecto a un repositorio en GitHub	35
6.3 Creación de la aplicación en Heroku.....	36
6.4 Importar base de datos local a la base de datos de Heroku	39
6.5 Conexión del proyecto con la base de datos	41
6.6 Acceso a la aplicación.....	42
7. VALIDACIONES.....	43

8. RECURSOS.....	46
8.1. Herramientas hardware.....	46
8.2. Herramientas software.....	46
8.3 Personal dedicado al mantenimiento y consecución del PFC.....	46
9. CONCLUSIONES.....	47
9.1. Grado de consecución de objetivos	47
9.2. Problemas encontrados	47
9.3. Mejoras	47
10. ANEXOS	49
11. BIBLIOGRAFÍA	51

0. INTRODUCCIÓN

El presente proyecto está destinado a la creación de una aplicación web para gestionar un huerto urbano.

0.1. Módulos a los que implica

El desarrollo del proyecto implica la aplicación de los conocimientos desarrollados en diferentes módulos del Ciclo superior de desarrollo de aplicaciones web.

A continuación, se citan cada uno de los módulos y los conocimientos aplicados.

Entornos de desarrollo:

Se aplican conocimientos de este módulo para el análisis de las necesidades de la aplicación. Fruto de este análisis se crean diagramas de clases y casos de uso que explican el funcionamiento de la aplicación.

Bases de datos:

Se crea una base de datos siguiendo el modelo Entidad/Relación, estudiado en este módulo. También se aplican los conocimientos adquiridos realizando funciones CRUD (create, read, update and delete) sobre la base de datos.

Programación:

Dado que la aplicación está creada con lenguaje Java, los conocimientos generales adquiridos en este módulo son aplicados en todo momento a la hora de implementar la aplicación.

Lenguaje de marcas:

Del mismo modo que el módulo de programación, los conocimientos de este módulo son imprescindibles para el desarrollo de la aplicación, en este caso en la parte cliente.

Desarrollo en entorno servidor:

Se aplican los conocimientos adquiridos sobre el desarrollo de aplicaciones web con Java, Servlets y JSP. Así como la utilización de diferentes patrones de diseño que mejoran la calidad de la aplicación en cuanto a su funcionamiento y reutilización.

Desarrollo en entorno cliente:

Se aplican los conocimientos adquiridos sobre expresiones regulares para la validación de formularios y la creación de consultas con Ajax.

Diseño de interfaces:

Los conocimientos adquiridos en este módulo se aplican con carácter general en toda la parte cliente de la aplicación. Se utilizan etiquetas semánticas HTML5 para la visualización de los contenidos. Se aplican estilos con CSS3, haciendo uso de los diferentes tipos de selectores que nos proporciona. Se crea una aplicación que pueda ser visible en diferentes dispositivos y se implementan algunas funciones empleando jQuery.

Despliegue de aplicaciones web:

Lo estudiado en este módulo se aplica tanto a la hora de implementar la aplicación como a la hora de desplegarlo.

Para el desarrollo de la aplicación es necesario hacer un despliegue en la máquina local, utilizando un contenedor de servlets (Apache Tomcat) y un sistema de gestión de base de datos. Una vez finalizada la aplicación web, ésta se desplegar en un servidor remoto.

Inglés técnico:

Sus conocimientos son aplicados sobre todo para la resolución de dudas en diferentes páginas web. También es necesario tener conocimientos de esta lengua para el uso de algunas aplicaciones imprescindibles para el desarrollo del presente trabajo.

0.2. Descripción

Hoy en día hay un gran número de huertos urbanos. Consisten en extensiones de cultivo parceladas que son alquiladas a particulares para el desarrollo de la agricultura de autoconsumo.

Aprovechando el elevado número de este tipo de huertos y ante la creciente demanda, se crea esta aplicación destinada a la gestión de huertos urbanos, que pretende optimizar el uso de los huertos, sus servicios y la relación entre los usuarios.

La aplicación contempla a tres tipos de usuarios, los no registrados, los registrados empleados/administradores y los registrados arrendatarios de parcelas.

Para cada uno de los diferentes usuarios, la aplicación trata de resolver unos problemas concretos.

Objetivos

Aplicar la mayoría de los conocimientos adquiridos en los diferentes módulos del FP de desarrollo de aplicaciones web.

Desarrollar una aplicación amigable que sea muy intuitiva para adaptarse a todo tipo de usuarios.

Desarrollar una aplicación Java bien estructurada, eficiente y con código reutilizable.

Desarrollar una aplicación web responsive que sea visible desde diferentes dispositivos. En inicio está pensada para que la aplicación sea usada, por parte del administrador, desde un ordenador o tableta, ya que, la aplicación podría contar con numerosas funcionalidades. Es por ello que en esta parte del desarrollo está más pensado para estos dispositivos de pantallas superiores a los 600/700px. A pesar de no estar desarrollado con la filosofía 'mobile first', se tiene en cuenta la visibilidad en dispositivos de menor resolución.

También se pretende con este trabajo la adquisición de nuevos conocimientos al enfrentarse a problemas durante el desarrollo, más que implementar muchas funcionalidades.

1. ESTUDIO PREVIO

Este proyecto no pertenece a ninguna empresa real, por tanto, el escenario que se presenta es ficticio.

Se trata de una aplicación destinada a personas que disponen de un terreno extenso para el cultivo. Este terreno está dividido en diferentes parcelas que son alquiladas a sus clientes.

Cada cliente podrá únicamente arrendar una parcela, dado que la filosofía de la empresa es crear una comunidad donde poder compartir, no sólo semillas y verduras sino también consejos y experiencias. Por ello se considera conveniente acotar el número de alquiler de las parcelas a un único arrendador.

Serán los administradores los que darán de alta y baja a los usuarios y asignarán las parcelas.

Además, se dispondrá de un tablón anuncios virtual donde el administrador podrá modificar los datos a mostrar a sus clientes a través de su cuenta.

Se trata de una aplicación ampliable, de modo que a la configuración básica se le podrían añadir diferentes funcionalidades.

Por ejemplo, existe la posibilidad de que haya un servicio de préstamo de herramientas que el arrendador pone a disposición de los usuarios. Tanto los usuarios como el administrador podrán reservar una herramienta para una fecha concreta. Una vez devuelta la herramienta, el administrador eliminaría el préstamo.

El **modelo de negocio** planteado en el presente proyecto, consiste en la creación de una aplicación que es ofrecida de modo particular a los dueños de diferentes huertos de 'autocultivo'. Cada cliente tiene su propia aplicación adaptada a sus características específicas, con una base de datos que permite la autenticación del administrador y define las parcelas que forman el huerto. El acceso a cada aplicación se realiza a través de una URL personal. La aplicación es personalizable por el administrador de los huertos. Se ofrecen diferentes paquetes o módulos que amplían las funcionalidades de la aplicación, tales como la gestión de herramientas y reservas de éstas, control de facturas de los arrendatarios, entre otras. Hay múltiples posibilidades de ampliación.

Se implementa en este proyecto un ejemplo con una serie de funcionalidades básicas para la aplicación.

Otras soluciones del mercado:

En la actualidad se encuentra en el mercado una serie de aplicaciones enfocadas a la planificación de los huertos, pero éstas están enfocadas a la gestión de los productos que va a ser plantados en las parcelas, dando consejos sobre el tipo de cultivo según la época del año, la gestión del espacio, consejos sobre abonos, riegos, etc. Algunas de ellas son GrowVeg, Garden Planner y Maceto Huerto.

Este tipo de aplicaciones se alejan del objetivo de la aplicación web planteada en el presente trabajo, si bien podrían ser un complemento de ésta, dando un mejor servicio a los usuarios de las parcelas.

2. PLAN DE TRABAJO

2.1. División del trabajo

Se establecen una serie de objetivos que debe cumplir el presente proyecto a fin de identificar el trabajo de desarrollo necesario.

El objetivo principal es la gestión administrativa por parte de la empresa, quedando la parte del usuario arrendador menos desarrollada por cuestiones de tiempo.

2.2. Estimación de tiempos

El tiempo empleado para la realización del presente trabajo ha sido superior al estimado en la propuesta.

FASE	DESCRIPCIÓN	TEMPORALIZACIÓN
1	Análisis de las necesidades de la aplicación, estudio de los casos de uso y diagrama de clases.	3- 4 hr. aprox
2	Diseño y prototipado. Capa de presentación, de negocio y de datos.	8 - 10 hr. aprox
3	Implementación, despliegue, corrección de errores y pruebas.	30 - 40 hr. aprox
4	Validación	1 hr. aprox
5	Redacción de toda la documentación	10 hr. aprox

Se han empleado más de 60 horas para la realización del presente trabajo, algunas de ellas destinadas a la investigación para solventar problemas.

3. ANÁLISIS

3.1 Funcionalidad de la aplicación

Está orientada a tres tipos de usuarios:

- **Usuarios anónimos:**

Obtener información: podrán acceder a la aplicación para obtener información sobre disponibilidad de parcelas, precios, servicios y otra información que el propietario considere importante.

Autenticación: introducción de usuario y contraseña para acceder a la parte privada de la aplicación.

- **Usuarios registrados administradores¹:**

Alta usuarios: podrá dar de alta a los usuarios del huerto y asignarles una parcela.

Baja usuarios: podrá eliminar usuarios, dejando libre la parcela que tuviesen asignada.

Listar usuarios: se genera un listado con los usuarios actuales y las parcelas que tienen asignadas, en su caso.

Ver tablón de anuncios: muestra el tablón con los horarios de riego y otras noticias.

Editar tablón de anuncios: podrá modificar los horarios de riego y otras noticias.

Cerrar sesión: se vuelve a la página de inicio al pulsar un botón que elimina la sesión del usuario.

¹ Para el presente proyecto no se diferencian entre tipos de administradores. Al tratarse de una aplicación ampliable se podría establecer diferentes roles entre los administradores para limitar el acceso de éstos a diferentes funcionalidades.

- **Usuarios registrados arrendatarios**

Tablón de anuncios: podrá leer la información publicada por el administrador.

Cerrar sesión: se vuelve a la página de inicio al pulsar un botón que elimina la sesión del usuario.

3.2 Requerimientos funcionales

En este apartado se detallan las funciones de la aplicación según el tipo de usuario.

- **Usuarios anónimos:**

Iniciar sesión	
Función	Acceder a la aplicación como usuario autenticado
Entrada	Nombre de usuario y contraseña
Proceso	Verificación de los datos de entrada en la base de datos
Salida	En el caso de ser correctos los datos confirma el acceso a la parte privada o, en caso contrario, muestra que el usuario y/o contraseña no son correctos

- **Usuarios registrados (administradores y arrendatarios):**

Ver tablón	
Función	Visualizar el contenido del tablón de anuncios
Entrada	-
Proceso	Consulta la base de datos para obtener los horarios de riego y los anuncios.
Salida	Muestra los horarios de riego y los anuncios publicados.

Cerrar sesión	
Función	Salir de la parte privada de la aplicación
Entrada	-
Proceso	Redirige a la página de inicio
Salida	-

- **Usuarios registrados administradores:**

Alta de usuario	
Función	Crear nuevos usuarios
Entrada	Nombre usuario, email, contraseña, nivel de permisos, asignación de parcela.
Proceso	Valida los datos introducidos. Consulta la existencia o no de ese usuario. Permite asignar una parcela disponible. Inserta un nuevo usuario en la base de datos. Modifica la tabla 'Parcelas' asignando un usuario a la parcela elegida.
Salida	Mensaje sobre si se ha insertado correctamente o no. Y los datos erróneos de entrada, en su caso.

Baja de usuario	
Función	Eliminar usuarios
Entrada	Email del usuario
Proceso	Valida que el usuario existe. Si el usuario es administrador, comprueba si es el único, ya que no se puede dejar la base de datos sin ningún administrador. Elimina el usuario de la base de datos. Modifica la tabla 'Parcelas' dejando nulo el campo del usuario de la parcela que disponía el usuario a dar de baja.
Salida	Mensaje sobre si se ha eliminado correctamente o no.

Añadir horario de riego	
Función	Añadir un nuevo horario de riego
Entrada	Texto con el nuevo horario.
Proceso	Añade la descripción a la base de datos. Inserta el campo en el tablón para que sea visible.
Salida	Mensaje sobre el éxito o fallo de la operación

Eliminar horario de riego	
Función	Elimina un horario de riego
Entrada	Envía el id del campo a eliminar
Proceso	Toma el id del elemento a eliminar de la lista. Lo borra de la base de datos.
Salida	Mensaje sobre el éxito o fallo de la operación En caso de éxito, desaparece el elemento de la lista de horarios

Añadir anuncio en tablón	
Función	Añadir un nuevo anuncio
Entrada	Texto del nuevo anuncio. Fecha actual del sistema
Proceso	Añade la descripción a la base de datos y la fecha actual del sistema. Inserta el campo en el tablón para que sea visible.
Salida	Mensaje sobre el éxito o fallo de la operación

Eliminar anuncio del tablón	
Función	Eliminar anuncio
Entrada	Id del anuncio
Proceso	Toma el id del elemento a eliminar de la tabla de anuncios. Lo borra de la base de datos.
Salida	Mensaje sobre el éxito o fallo de la operación

3.3 Diagramas UML

Para analizar el funcionamiento de la aplicación se ha optado por la utilización del modelo de lenguaje unificado (UML) que dispone de múltiples tipos de diagramas que ayudan a plasmar las funcionalidades y requisitos de la aplicación. Con este fin, han sido elaborados diagramas de clases y diagramas de casos de uso.

3.3.1 Diagrama de clases

El diagrama de clases define la estructura que presenta el sistema y muestra las relaciones entre los diferentes elementos, que son definidos por clases con sus atributos.

Dado que se trata de una primera aproximación, no se ha considerado necesario incluir los métodos de cada clase.

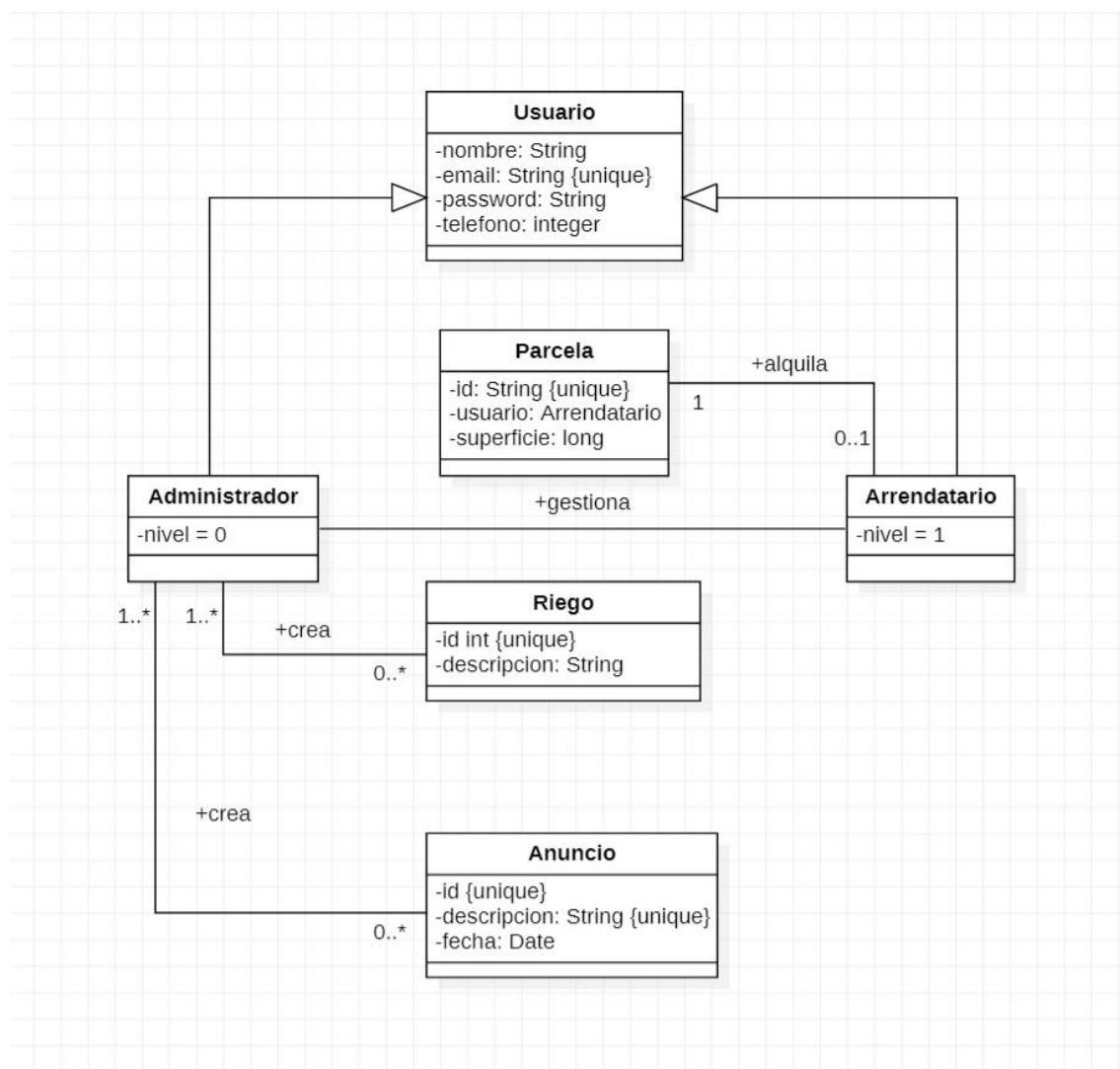


Figura 1. Diagrama de clases

3.3.2 Casos de uso

- Actores: en este caso son los usuarios que utilizan la aplicación. Cada uno de ellos tiene un rol propio que le permite realizar diferentes acciones.

- Casos de uso: representan cada una de las funcionalidades que tiene la aplicación. El conjunto de todos ellos define la funcionalidad total de la aplicación.

Se han creado una serie de casos de uso para el usuario con rol de administración, ya que es el usuario con más funcionalidades implementadas. Algunas de estas funcionalidades no se han implementado en la aplicación, ya que se necesitarían más horas de trabajo que las establecidas para este tipo de proyectos.

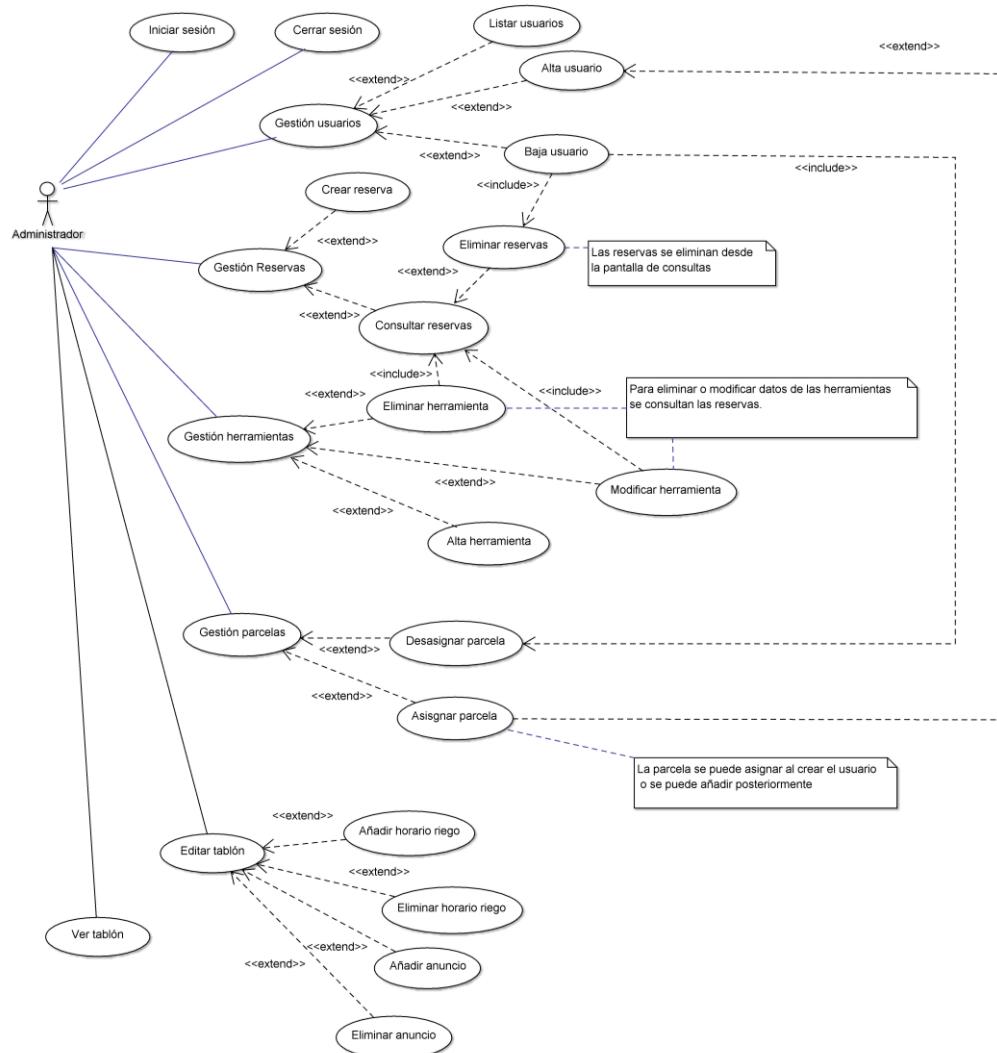


Figura 2. Casos de usos. Administrador

3.4 Capturas de las funcionalidades



Figura 3. Página de inicio

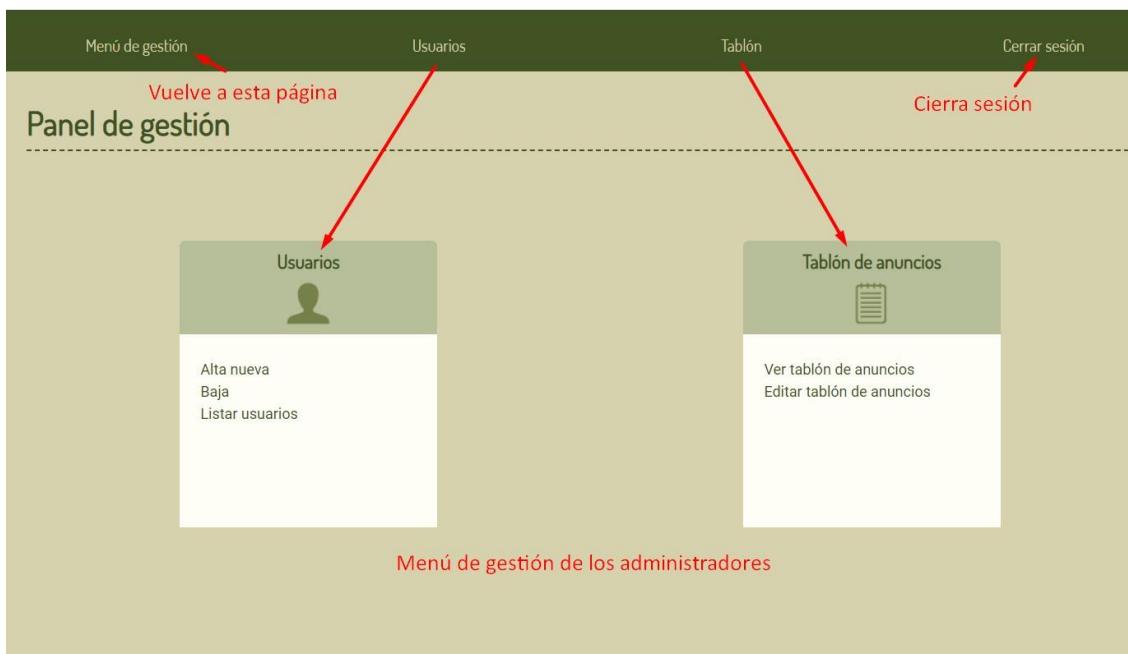


Figura 4. Menú de gestión del administrador

Menú de gestión Usuarios Tablón Cerrar sesión

Panel de gestión

Alta usuarios

Rol del Usuario

Administrador
 Arrendador

Datos personales

Nombre*

Email*

Teléfono

Contraseña*

Asignar parcela

Sin parcela

Muestra parcelas libres

Formulario de alta con validación de los campos en el servidor

Figura 5. Formulario de alta de usuarios

Menú de gestión Usuarios Tablón Cerrar sesión

Panel de gestión

Baja usuarios

Seleccione el usuario a dar de baja

Seleccionar usuario—

Baja de usuarios.
Comprueba si el administrador es único

Figura 6. Formulario de baja de usuarios

The screenshot shows a user management interface. At the top, there are navigation links: 'Menú de gestión', 'Usuarios', 'Tablón', and 'Cerrar sesión'. Below this, a title 'Panel de gestión' is followed by 'Listado de usuarios'. A table displays three users:

Nombre	Email	Teléfono	Parcela	Rol
Lolo Prado	lolo@gmail.com	0	04A	Arrendatario
Marta Rigonda	marta@gmail.com	610200200	No asignada	Arrendatario
Laura Milo	admin@mihuerto.com	0	No asignada	Administrador

Figura 7. Muestra una lista con los usuarios y sus datos

The screenshot shows the 'Tablón de anuncios' section of the application. It includes a header with 'Edición del tablón', 'Muestra los horarios', 'Elimina horarios', 'Nuevo horario', 'Añadir horario', 'Formulario para añadir horarios', 'Muestra los anuncios', and 'Elimina anuncios'. Below this, there's a 'Novedades importantes' section with a table of announcements:

Debes saber que...	Fecha	Eliminar
Mañana será otro dia	2021-06-08	Eliminar
Tenemos semillas de calabaza!	2021-06-04	Eliminar
Próximo domingo 30 Junio, trueque de verduras	2021-06-04	Eliminar
En el mes de julio tendremos novedades!! Pásate por la caseta de info y pregunta.	2021-06-01	Eliminar

At the bottom, there's a 'Añade nuevo anuncio' section with a 'Formulario para añadir anuncios' and buttons for 'Añadir anuncio' and 'Borrar contenido'.

Figura 8. Funcionalidades del editor del tablón de anuncios

4. DISEÑO

4.1 Capas de la aplicación

El diseño de la aplicación utiliza una arquitectura de 4 capas donde los diferentes niveles son independientes los unos de los otros, de este modo se separa a capa de datos de la lógica de negocio de la interfaz de usuario.

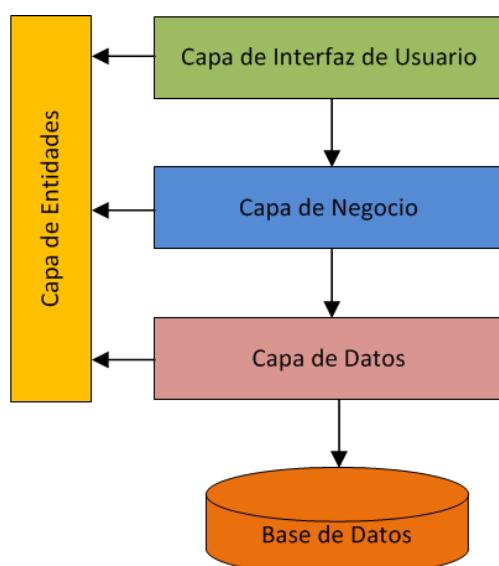


Figura 9. Esquema 4 capas. Fuente: <https://saov.wordpress.com/>

- Capa presentación:

Parte cliente. La aplicación será accesible desde cualquier dispositivo con navegador web que tenga acceso a internet.

Está formada por páginas JSP, que pintan la información por pantalla al usuario, y servlets, que reciben las peticiones del cliente y dan una respuesta. Las páginas JSP permiten la creación de webs dinámicas.

Fuentes:

Roboto, de google fonts, para el texto en general.

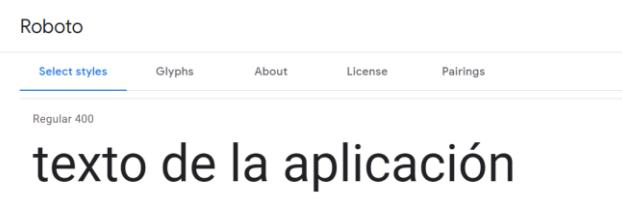


Figura 10. Captura de pantalla de Google fonts

Dosis, de google fonts, para titulares y menú principal.

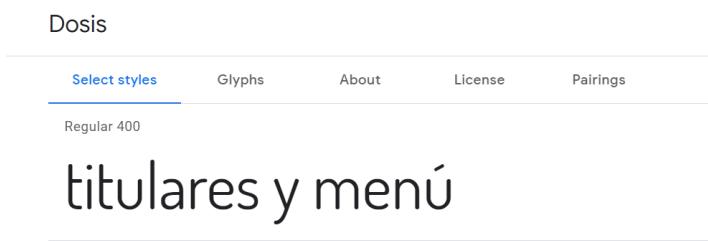


Figura 11. Captura de pantalla de Google fonts

Paleta de colores:

Para la selección de colores se ha utilizado la herramienta online paletton. En concreto se han aplicado los colores adyacentes que pueden verse en el siguiente enlace:

<http://paletton.com/#uid=51C0-0k6ep417Gx17GxdEqWdagr>



Figura 12. Paleta de colores

Además, se ha hecho uso del color rojo para resaltar los errores de validación de formularios y otros.

Iconografía: la iconografía utilizada es gratuita. Se ha obtenido a través de la aplicación IcoMoon.

Imágenes: Se tratan con Adobe Photoshop para redimensionarlas optimizando su carga en la web.

La resolución de las fotos es de 96 píxeles por pulgada y el tamaño depende de las necesidades de cada caso.

Todas las imágenes fotografías empleadas en la aplicación han sido creadas por su autora, por lo que no son necesarios los permisos de autor.



Figura 13. Imagen de fondo de la página de inicio

Diagrama de navegabilidad: la página principal muestra información general del huerto y un formulario para iniciar sesión.

Una vez autenticados, la aplicación redirige a una página u otra según el rol que tenga el usuario.

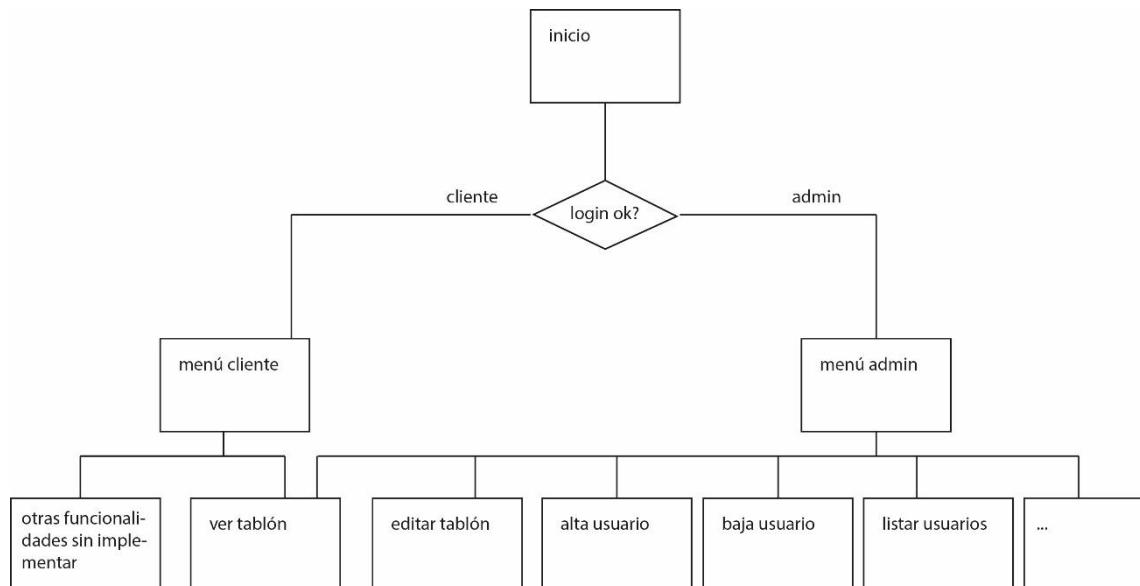


Figura 14. Diagrama navegabilidad de la aplicación

Interfaz de usuario: una vez iniciada la sesión la plantilla es la misma para todos los usuarios. Simplemente va cambiando el contenido a mostrar.

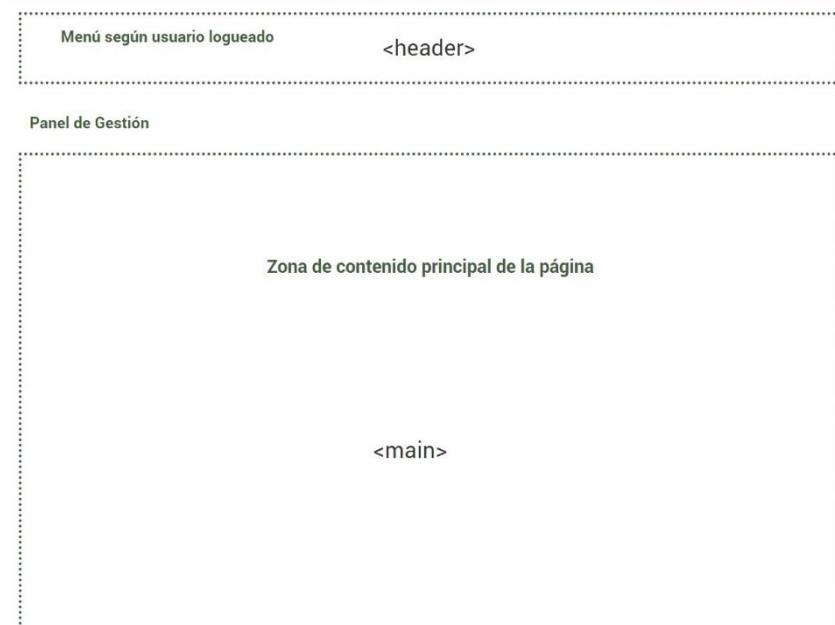


Figura 15. Esquema de interfaz de usuario

Diseño responsive: Se ha adaptado el estilo para que cada una de las páginas de la aplicación también sea visible en dispositivos más pequeños.



Figura 16. Página de inicio

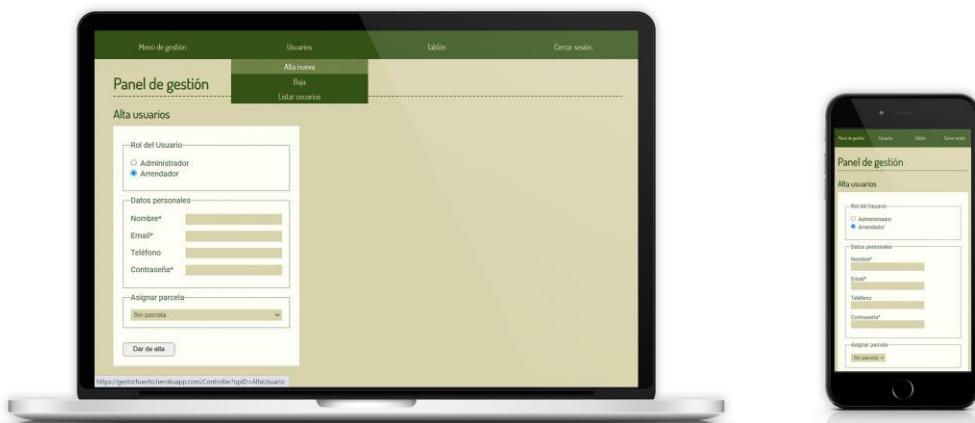


Figura 17 Visualización tablas en escritorio

Para la visualización correcta de las tablas, que son un elemento importante en esta aplicación, se ha optado por permitir un scroll horizontal, de modo que la tabla se adapta a las dimensiones del dispositivo sin renunciar a su correcta visualización y sin eliminar ninguno de sus campos.



The screenshot shows a desktop browser window with the title 'Panel de gestión'. At the top, there is a dark green header bar with four items: 'Menú de gestión', 'Usuarios', 'Tablón', and 'Cerrar sesión'. Below the header, the main content area has a light beige background. It features a section titled 'Listado de usuarios' with a table. The table has columns for 'Nombre', 'Email', 'Teléfono', 'Parcela', and 'Rol'. There are four rows of data: Lolo Prado (Email: lolo@gmail.com, Phone: 0, Parcela: 04A, Rol: Arrendatario), Marta Rigonda (Email: marta@gmail.com, Phone: 610200200, Parcela: No asignada, Rol: Arrendatario), and Laura Milo (Email: admin@mihuerto.com, Phone: 0, Parcela: No asignada, Rol: Administrador). The last row is partially visible. A scroll bar is visible on the right side of the table.

Figura 18. Visualización tablas en escritorio



The screenshot shows a mobile browser window emulating an iPhone 6/7/8. The title bar indicates the device type as 'iPhone 6/7/8', the width as '375', the height as '667', the zoom level as '81%', and 'No throttling'. The main content area is identical to Figure 18, showing the 'Panel de gestión' interface with the 'Listado de usuarios' table. A scroll bar is visible on the right side of the table.

Figura 19. Visualización tablas en móvil

- Capa de aplicación:

Esta es la capa que contiene toda la lógica de negocio. Es la capa donde se implementan las funcionalidades de la aplicación. Se encarga de hacer las consultas a la base de datos recogiendo las peticiones realizadas por la capa de presentación, a la que le devuelve unos resultados.

- Capa de integración/datos:

Se encarga de crear la conexión y desconexión a la base de datos y realizar las operaciones pertinentes sobre ella, siempre que sea posible. Devuelve los resultados de estas operaciones.

- Capa de entidades:

Contiene las clases java. Esta capa se comunica con el resto ofreciéndoles información.

- Base de Datos: está formada por las siguientes tablas: usuarios, parcelas, riego y anuncios.

A continuación, se describen de manera lineal las diferentes tablas.

USUARIOS (email, nombre, pass, telefono, rol)

CP: email

VNN: pass, rol

PARCELAS (id, usuario, superficie)

CP: id

Caj: usuario → USUARIOS (email)

HORARIO (id, descripcion)

CP: id

VNN: descripcion

ANUNCIOS (id, descripción,fecha)

CP: id

VNN: descripcion

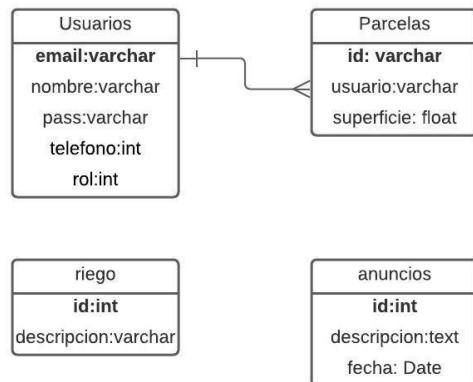


Figura 20. Diagrama E/R

5. IMPLEMENTACIÓN

5.1. Estructura de la aplicación

La aplicación ha sido desarrollada con servlets java y páginas de servidor java (JSP).

Se ha creado como proyecto Maven, ya que es necesario para su despliegue en la plataforma Heroku.

Estas son las carpetas que lo componen.

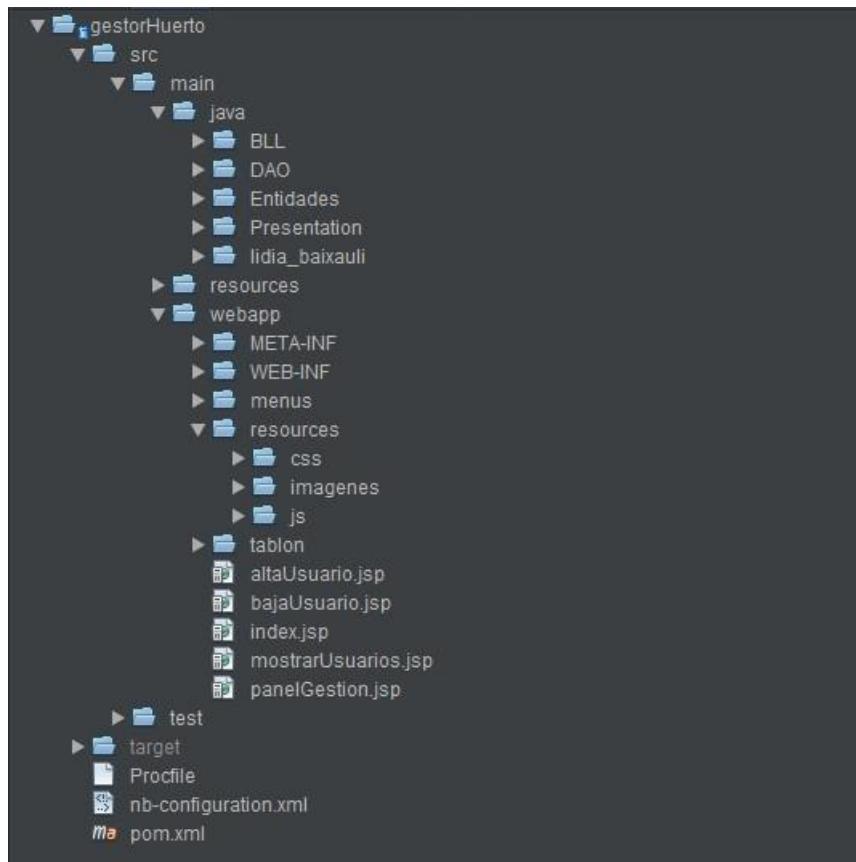


Figura 21. Carpetas del proyecto

Src > main > Java: contiene

Carpeta Entidades: contiene modelos

Carpeta Presentation: contiene la clase Controller, Dispatcher, ICommand y los diferentes servlets que extienden de ICommand.

Carpeta DAO: clases java para crear consultas a la BBDD. Incluye la clase para conectarse a la BBDD.

Carpeta BLL (business layer logic). Contiene la clases que comunican la capa presentación y BBDD.

Src > main > Webapp > Contiene páginas jsp y la carpeta resources, donde se alojan las carpetas css, js y de imágenes.

Pom.xml → archivo donde se configuran las dependencias y plugins necesarios.

Procfile → necesario para el despliegue en Heroku.

Se han aplicado diferentes **patrones de diseño** para la implementación:

- Patrón Modelo Vista Controlador

Este patrón separa los datos de la aplicación, la lógica de negocio y la interfaz de usuario.

- Patrón Front Controller

Dota a la aplicación de un único controlador frontal, que recibe todas las peticiones entrantes. Facilita el desarrollo y mejora la seguridad, ya que sólo hay un punto de acceso.

Se implementa un control para comprobar si el usuario está logueado o no y así evitar accesos no permitidos.

```

@WebServlet(name = "Controller", urlPatterns = {"/*Controller"})
public class Controller extends HttpServlet {
    /**
     * Processes requests for both HTTP <code>GET</code> and <code>POST</code> ...9 lines */
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        try {
            String operacion=request.getParameter("opID");

            //comprueba si está o no conectado
            Usuario usuarioSesion = (Usuario)request.getSession().getAttribute("usuarioSesion");
            if(usuarioSesion==null && !(operacion.equals("Login"))){
                request.getRequestDispatcher("").forward(request,response);
            }

            //carga de forma dinámica la clase que implementa ICommand
            ICommand command=
                (ICommand) Class.forName("Presentation.Command."+operacion+"Command").newInstance();
            //carga los datos iniciales necesarios para la siguiente página a visualizar
            command.initPage(request, response);
            //ejecuta la clase y recibe la siguiente página a visualizar
            String nextPage=command.execute(request, response);
            new Dispatcher().procesa(request, response, nextPage);

        } catch (Exception ex) {
            ex.printStackTrace();
        }
    }

    //HttpServlet methods. Click on the + sign on the left to edit the code.
}

```

Figura 22. Class Controller

- Dispatcher View

Libera al controlador de la función de llamar a la vista, de modo que se encarga de realizar las comprobaciones sobre el usuario y gestionar la salida.

Si el usuario no ha iniciado sesión, se le redirige a la página de inicio.

```

    /**
     * 
     */
    public class Dispatcher {
        public void procesa(HttpServletRequest request, HttpServletResponse response, String nextPage)
        throws ServletException, IOException{
            response.setContentType("text/html; charset=UTF-8");
            // request.getRequestDispatcher(nextPage).forward(request, response);
            request.setAttribute("paginaPrincipal", nextPage);
            // si no hay usuario logueado vamos al índice
            Usuario usuarioSesion = (Usuario)request.getSession().getAttribute("usuarioSesion");
            if(usuarioSesion!=null){
                request.getRequestDispatcher("/panelGestion.jsp").forward(request, response);
            }else{
                request.getRequestDispatcher("/index.jsp").forward(request, response);
            }
        }
    }
}

```

Figura 23. Class Dispatcher

- Patrón de vista compuesta

Consiste en la creación de una plantilla donde se muestra el contenido de diferentes vistas.

En la aplicación se utiliza la vista de panelGestion.jsp como plantilla donde se carga un menú adaptado al rol de cada usuario y la vista a mostrar.

```

<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Huerto Mandroca</title>
        <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
        <!--<link rel="stylesheet" type="text/css" href="style.css"/>-->
        <link rel="stylesheet" type="text/css" href="resources/css/style.css"/>
        <!--FUENTES GOOGLE-->
        <link rel="preconnect" href="https://fonts.gstatic.com">
        <link href="https://fonts.googleapis.com/css2?family=Dosis&family=Roboto&display=swap" rel="stylesheet">
    </head>
    <header><jsp:include page="menus/menu.jsp" /></header> ← carga la vista menu.jsp
    <body>
        <main id="gestor">
            <h1>Panel de gestión</h1>
            <% String paginaPrincipal=(String) request.getAttribute("paginaPrincipal");%>
            <% if(paginaPrincipal!=null){ %>
                <jsp:include page='<%=String(request.getAttribute("paginaPrincipal"))%>' />
            <% }else{
                Usuario usuarioSesion = (Usuario)request.getSession().getAttribute("usuarioSesion");
                if(usuarioSesion!=null){
                    if(usuarioSesion.getRol()==0){
                        %>
                        <jsp:include page='menuGestionAdmin.jsp' />
                    <% }else{//es arrendador
                        %>
                        <jsp:include page='menuGestionArrendador.jsp' />
                    <% }
                <% }else{
                    %>
                    <jsp:include page='/' />
                <% }
            <% }%>
        </main>
    </body>

```

Annotations on the code:

- Annotation 1: A red arrow points to the line `<jsp:include page="menu.jsp" />` with the text "carga la vista menu.jsp".
- Annotation 2: A red arrow points to the line `<jsp:include page='<%=String(request.getAttribute("paginaPrincipal"))%>' />` with the text "carga la vista que indique el servlet".
- Annotation 3: A red arrow points to the line `<jsp:include page='menuGestionAdmin.jsp' />` with the text "carga un menú diferente según el rol de usuario".
- Annotation 4: A red arrow points to the line `<jsp:include page='/' />` with the text "Control de accesos no permitidos".

Figura 24. Archivo panelGestion.jsp donde se cargan el resto de páginas de los usuarios logueados

- Command Helper

Libera al controlador de recoger las peticiones de los usuarios, de llamar a la lógica de negocio y de devolver su respuesta al cliente.

Se crea la clase abstracta ICommand que trata todas las peticiones HTTP que llegan al controlador.

```
package Presentation.Command;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/*
 * @author Lidia Baixauli de la Villa
 */
public abstract class ICommand {
    public void initPage(HttpServletRequest request, HttpServletResponse response)
        throws Exception
    {
    }

    public abstract String execute(HttpServletRequest request, HttpServletResponse response)
        throws Exception;
}
```

Figura 25. Clase abstracta ICommand

A partir de aquí, se crean los diferentes servlets que extenderán de ICommand y se encargarán de procesar las diferentes peticiones.

El método **initPage** se utiliza en aquellos casos en que sea necesario obtener datos antes de cargar la vista.

El método **execute** recoge, si es necesario, los datos enviados desde el cliente y ejecuta las llamadas a la lógica de negocio. Y finalmente, llama a la vista que debe mostrar.

```
* @author Lidia Baixauli de la Villa
*/
public class MostrarTablonCommand extends ICommand {

    @Override
    public void initPage(HttpServletRequest request, HttpServletResponse response) throws Exception {
        //listado riego
        HorarioBLL horaBLL = new HorarioBLL();
        request.setAttribute("listaHorarioRiego", horaBLL.listaHorarios());
        //listado anuncios
        AnuncioBLL anuncioBLL= new AnuncioBLL();
        request.setAttribute("listaAnuncios", anuncioBLL.listaAnuncios());
    }

    public String execute(HttpServletRequest request, HttpServletResponse response) throws Exception {
        return "/tablon/verTablon.jsp"; ← envía la vista a mostrar
    }
}
```

carga los datos de los combos

Figura 26. Ejemplo de clase que extiende de ICommand

```

<fieldset>
    <legend>Asignar parcela</legend>
    <!--será un combo cargado con las parcelas-->
    <% List listaParcelas = (List) request.getAttribute("listaParcelasDisp");
        if(listaParcelas.size()==0){
    %>
    <p class="error">Actualmente no hay parcelas disponibles</p>
    <%
    }
    %>
    <select name="parcela">
        <option value="null">Sin parcela</option>
        <%
            //List listaParcelas = (List) request.getAttribute("listaParcelasDisp");
            for ( int i=0;i<listaParcelas.size();i++){
                Parcela parcela = (Parcela)listaParcelas.get(i);
            %>
            <option value="<%=>parcela.getId()%>">
                <%=>parcela.getId()%>
            </option>
            <%
            }
        <%
    }
    %>

    </select>
</fieldset>

```

carga la lista enviada por el servlet

Figura 27. Página JSP donde se carga el combo de parcelas disponibles

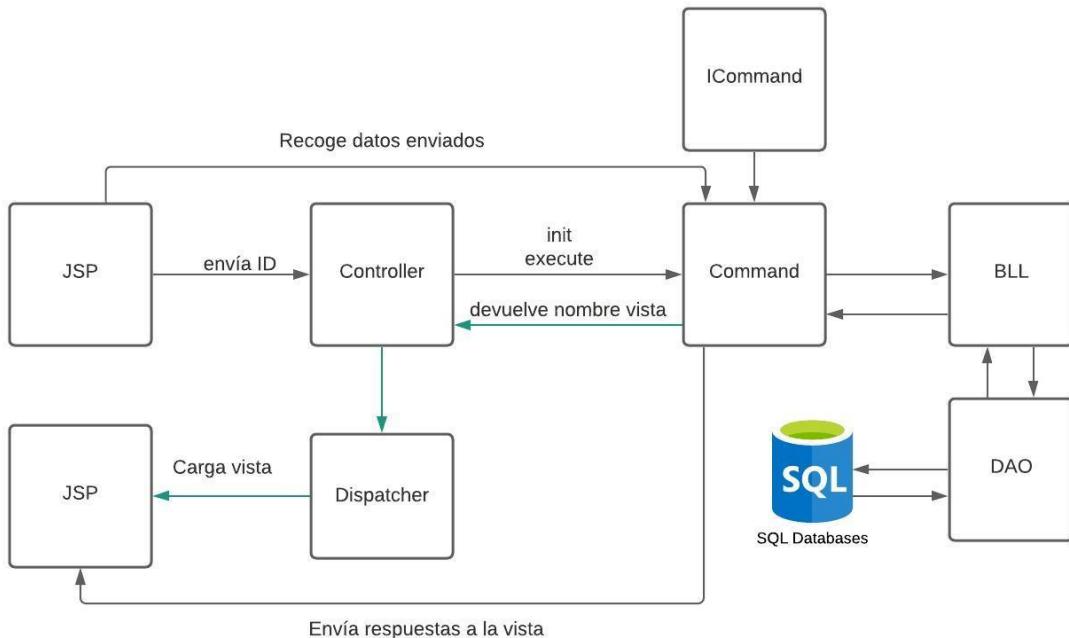


Figura 28. Esquema de la aplicación

5.2 Lenguajes utilizados

HTML5 y JSP

Se han empleado ambas tecnologías para la creación de páginas web dinámicas.

La utilización de etiquetas semánticas proporciona información a los navegadores sobre el contenido de los documentos.

CSS3

Las hojas de estilo en cascada son el lenguaje formal para definir el aspecto visual de las aplicaciones web. Se ha tratado de utilizar de una manera eficiente, haciendo uso de selectores y pseudo-clases.

Se han aplicado transiciones para mejorar la usabilidad de la aplicación, haciendo más visibles las acciones sobre los enlaces del menú.

La parte responsive se ha realizado con @media Queries.

```

.menuAdmin a:hover{
    color: #B7BF9A;
}

.menuAdmin>div:first-child{
    padding-top: 8px;
    text-align: center;
    width: 100%;
    background-color: #B7BF9A;
    border-radius: 5px 5px 0 0;
}

.menuAdmin>div:last-child{
    padding: 20px
}

input:not([type="submit"]):not([type="reset"]):not([type="radio"]){
    padding: 3px;
    background-color: #d6d2ae;
    border: none;
    width: 200px;
}

input[type=reset]{
    border-color: #ff5722;
    background-color: #ffebee;
}

input[type=radio]{
    width: initial;
    margin-right: 7px;
}

.camposForm{
    margin-bottom: 10px;
}

select{
    width: 314px;
    border: none;
    padding: 3px;
    background-color: #d6d2ae;
    color: #416532;
}

```

Figura 29. Código CSS

JavaScript , jQuery, AJAX

Se han realizado consultas Ajax para eliminar los anuncios y horarios del tablón de anuncios sin recargar la página. Se ha manipulado el DOM en caso de éxito.

```

$(document).ready(function() {
    $('span').on('click',function(event) {
        var li = $(this).parent();
        var elemento = $(this).parent().attr('id');
        //alert(elemento);

        $.ajax({
            type: "POST",
            url: "../Controller?opID=EliminarHorario",
            data: "elemento="+elemento
        }).done(function (Response) {
            //si todo va bien borra el link

            li.remove();
            $('#infoHorario').html("Horario eliminado con éxito");
            if($('.exito')){$('.exito').remove();}
        })
        .fail(function (Response) {

    });

}); //fin eliminar
// elimina anuncio

```

Figura 30. Llamada AJAX con jQuery

Java

Lenguaje empleado en la parte servidor de la aplicación.

Observaciones:

Para codificar correctamente algunos caracteres especiales se ha hecho uso de la clase StandardCharsets, del paquete java.nio.charset.

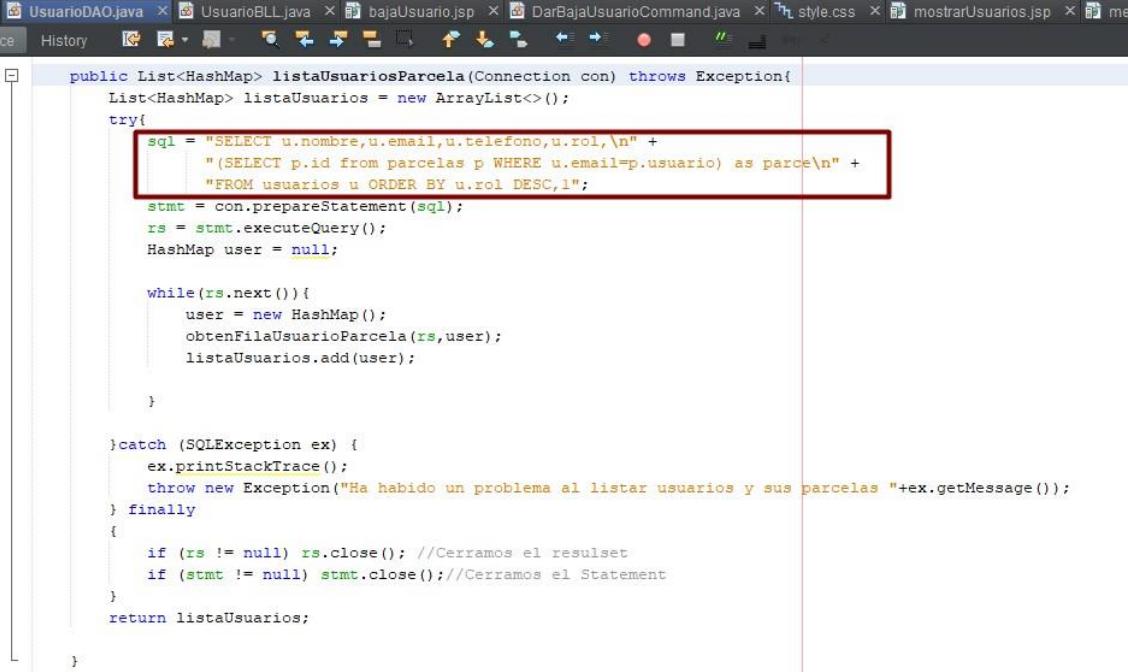
Para devolver la consulta que devuelve los usuarios y su parcela asignada, los resultados han sido almacenados en una lista de HashMap.

El formulario de alta de usuario se ha validado mediante el uso de expresiones regulares con java.util.regex.

SQL

Lenguaje para la creación, consulta y modificación de la base de datos.

Las sentencias SQL ejecutadas desde la aplicación se implementan en las clases pertenecientes al paquete DAO.



```
public List<HashMap> listaUsuariosParcela(Connection con) throws Exception{
    List<HashMap> listaUsuarios = new ArrayList<>();
    try{
        sql = "SELECT u.nombre,u.email,u.telefono,u.rol,\n" +
              "(SELECT p.id from parcelas p WHERE u.email=p.usuario) as parce\n" +
              "FROM usuarios u ORDER BY u.rol DESC,1";
        stmt = con.prepareStatement(sql);
        rs = stmt.executeQuery();
        HashMap user = null;

        while(rs.next()){
            user = new HashMap();
            obtenerFilaUsuarioParcela(rs,user);
            listaUsuarios.add(user);
        }

    }catch (SQLException ex) {
        ex.printStackTrace();
        throw new Exception("Ha habido un problema al listar usuarios y sus parcelas "+ex.getMessage());
    } finally
    {
        if (rs != null) rs.close(); //Cerramos el resulset
        if (stmt != null) stmt.close(); //Cerramos el Statement
    }
    return listaUsuarios;
}
```

Figura 31. Sentencias SQL en las clases del paquete DAO

6. DESPLIEGUE

Para el despliegue de la aplicación se ha optado por la plataforma gratuita Heroku. Se trata de una plataforma como servicio de computación en la nube, que soporta diferentes lenguajes, entre ellos Java.

Para desplegar una aplicación Java en Heroku, se debe crear un proyecto Maven.

6.1 Configuración del proyecto en Netbeans

1. Sustituir en el archivo pom.xml la dependencia **maven-dependency-plugin** creada por defecto en Netbeans por esta otra.

```
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-dependency-plugin</artifactId>
    <version>3.0.2</version>
    <executions>
        <execution>
            <phase>package</phase>
            <goals><goal>copy</goal></goals>
            <configuration>
                <artifactItems>
                    <artifactItem>
                        <groupId>com.github.jsimone</groupId>
                        <artifactId>webapp-runner</artifactId>
                        <version>8.5.31.0</version>
                        <destFileName>webapp-
runner.jar</destFileName>
                    </artifactItem>
                </artifactItems>
            </configuration>
        </execution>
    </executions>
</plugin>
```

2. Crear en la raíz del proyecto un archivo llamado '**Procfile**' e incluir el siguiente contenido.

```
web: java $JAVA_OPTS -jar target/dependency/webapp-runner.jar --port $PORT
target/*.war
```

6.2 Subir el proyecto a un repositorio en GitHub

1. Crear un nuevo proyecto en la plataforma. En este caso se crea el repositorio llamado gestorHuerto.

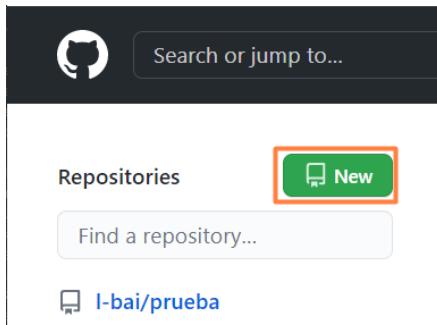


Figura 32. Creación nuevo repositorio en GitHub

2. Realizar la subida del proyecto. Desde la terminal del ordenador se accede a la ruta donde se encuentra el proyecto.

Se inicia el repositorio local mediante el comando `git init`.

Conectar el repositorio local con el de GitHub mediante el comando

```
git remote add origin  
https://github.com/nombreUsuario/nombreRepositorio.git
```

A continuación, se añaden los archivos que hayan tenido cambios desde la última subida. Al ser la primera, se añaden todos. Para ello se ejecuta:

```
git add .
```

Se lanza el siguiente comando para mandar los archivos al repositorio. Es aconsejable añadir comentarios descriptivos que ayuden a identificar los cambios realizados con esa subida.

```
git commit -m 'primera subida'
```

Por último, se suben los archivos al repositorio ejecutando:

```
git push -u origin master
```

Este paso solicita la autentificación de la cuenta GitHub.

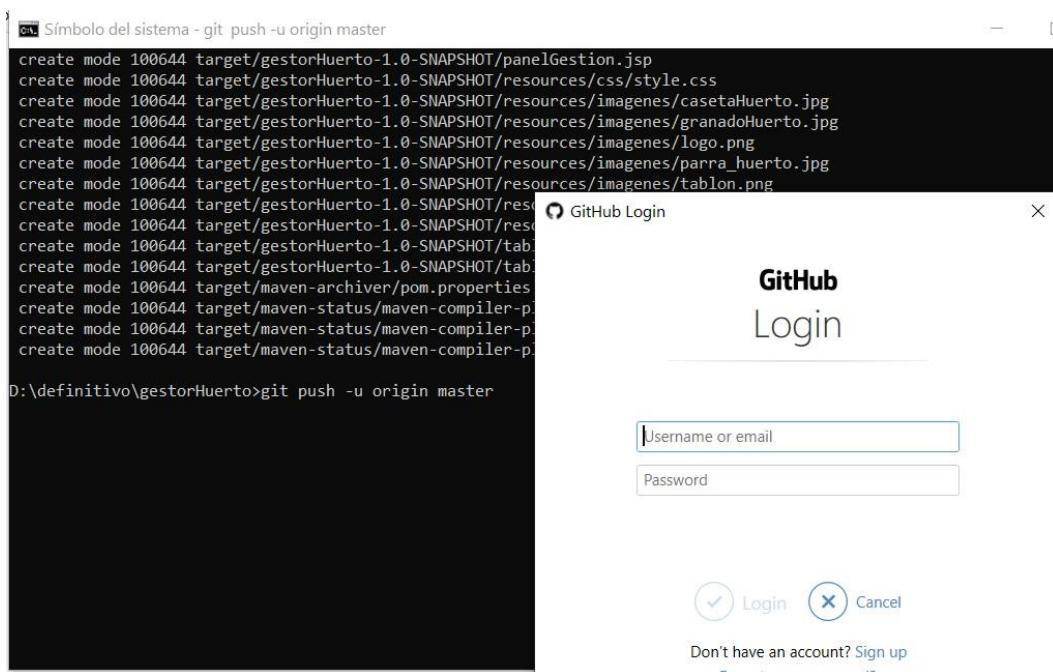


Figura 33. Subida del proyecto a GitHub

6.3 Creación de la aplicación en Heroku

1. En primer lugar, se crea una aplicación nueva de heroku.

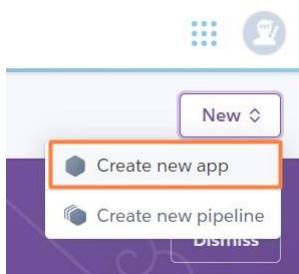


Figura 34. Creación app en Heroku

2. Se elige un nombre y una región.

The image shows the Heroku 'Create app' configuration form. It includes fields for 'App name' (set to 'gestorhuerto'), 'Choose a region' (set to 'Europe'), and a 'Create app' button at the bottom. A note above the region field states 'gestorhuerto is available'.

Figura 35. Configuración app

3. A continuación, se conecta la aplicación con el repositorio de GitHub creado para el proyecto.

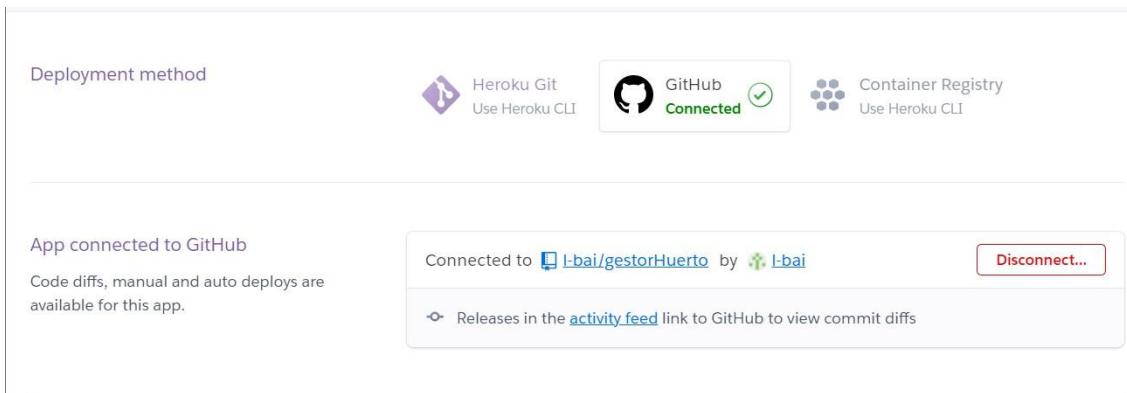


Figura 36. Conectar al repositorio GitHub

4. Desplegar la aplicación desde este panel o bien desde la terminal.

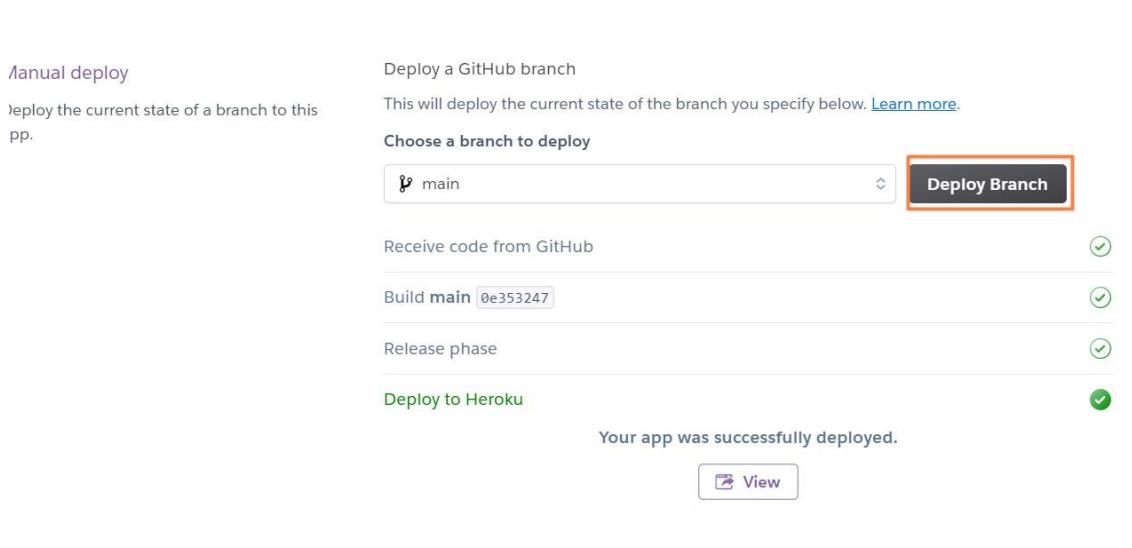


Figura 37. Despliegue de la aplicación

5. Se comprueba que se puede acceder a la aplicación desde el navegador.

<https://gestorhuerto.herokuapp.com/>

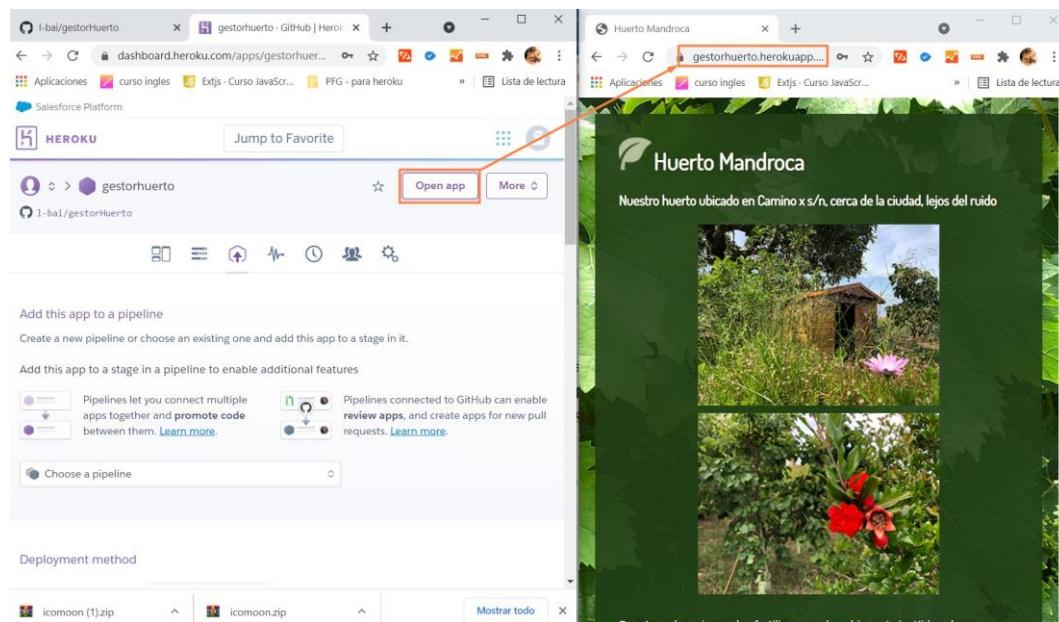


Figura 38. Aplicación desplegada en Heroku

6. El siguiente paso es crear la base de datos en la aplicación de Heroku. Para ello se accede al panel de opciones de la aplicación y se selecciona 'Elements'.

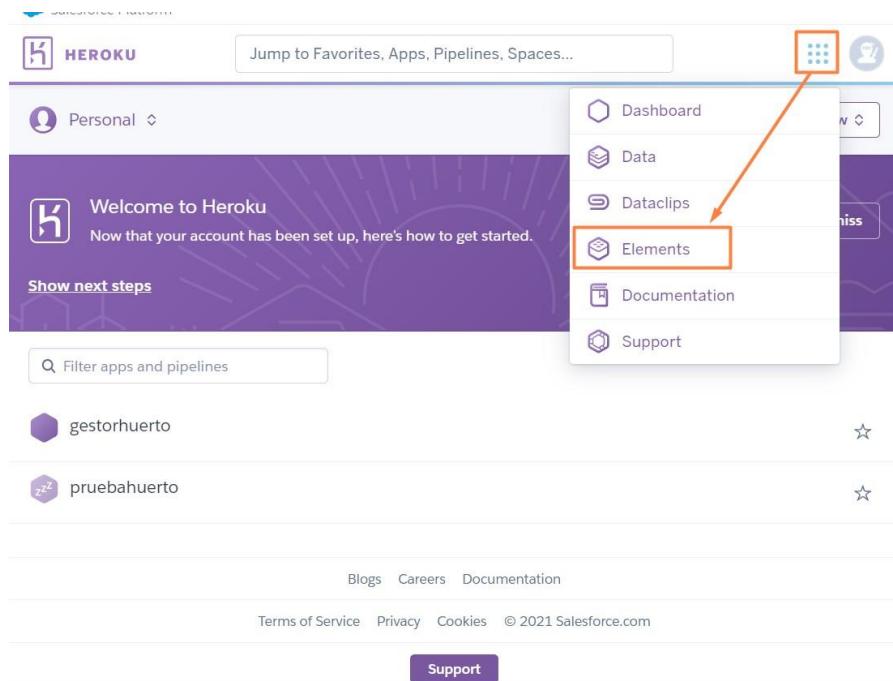


Figura 39. Panel de opciones de la aplicación de Heroku

7. Se busca el add-On ClearDB MySQL y se instala en la aplicación de Heroku deseada.

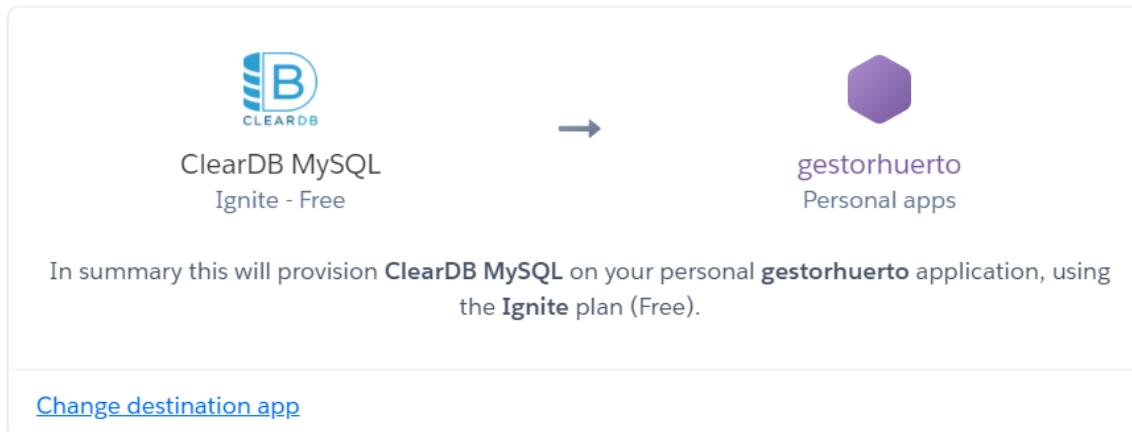


Figura 40. Instalación de ClearDB

8. Una vez instalada, se deben obtener las opciones de configuración para poder acceder a la base de datos. Para ello se abre la terminal y se ejecuta el comando `heroku login` para acceder a la aplicación.

Después de loguearse se ejecuta el comando que mostrará la URL de la base de datos.

```
heroku config --app gestorhuerto
```

Esta URL muestra los datos de usuario, contraseña, host y nombre de la base de datos, que se deben cambiar en el proyecto para acceder a la base de datos creada con Heroku.

```
D:\definitivo\gestorHuerto>heroku config --app gestorhuerto
== gestorhuerto Config Vars
CLEARDB_DATABASE_URL: mysql://b932d6f9663017:2110581e@eu-cdrb-west-01.cleardb.com/heroku_23638fac0af2477?reconnect=true
D:\definitivo\gestorHuerto>
```

Figura 41. URL de la base de datos ClearDB

6.4 Importar base de datos local a la base de datos de Heroku

Para acceder y gestionar la base de datos ClearDB, se ha instalado MySQL Workbench de Oracle, que es una herramienta visual para el diseño de base de datos.

Para importar la base de datos se siguen los siguientes pasos:

1. Introducir los datos de conexión a la base de datos.

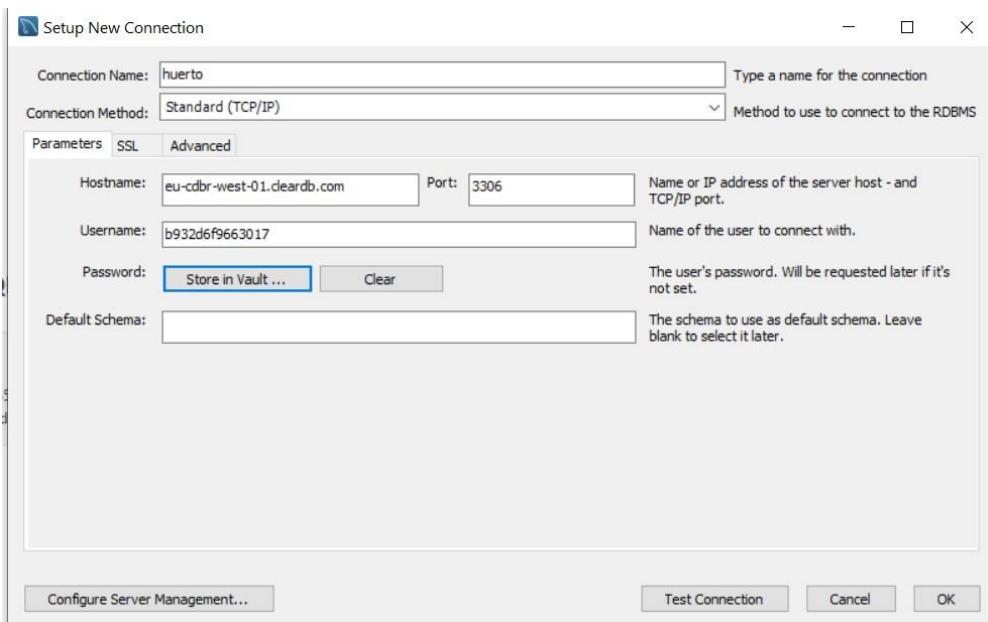


Figura 42. Conexión a la base de datos con Workbench

2. Seleccionar el archivo sql de la copia de base de datos que se ha creado en local durante la implementación.

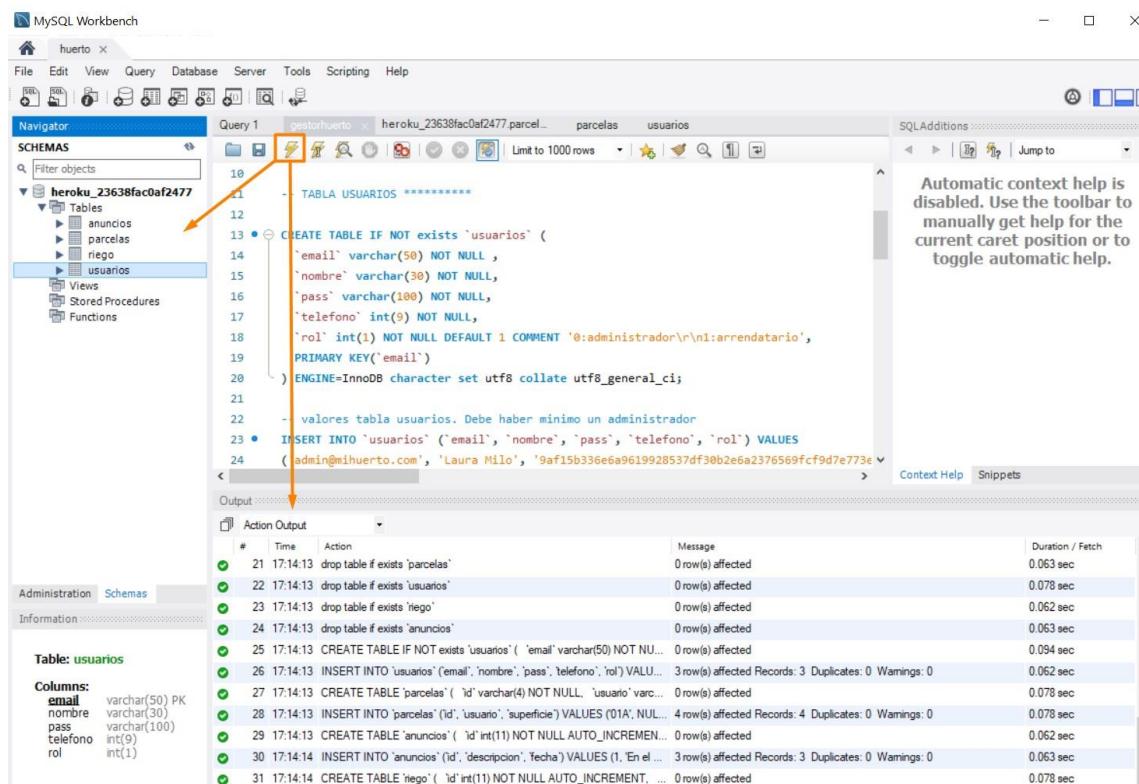


Figura 43. Importación de la base de datos a ClearDB con Workbench MySQL

3. Comprobar que todas las tablas han sido creadas y que contienen la información necesaria. Como mínimo un usuario administrador y algunas parcelas (ya que no se ha implementado la inclusión de éstas a través de la aplicación).

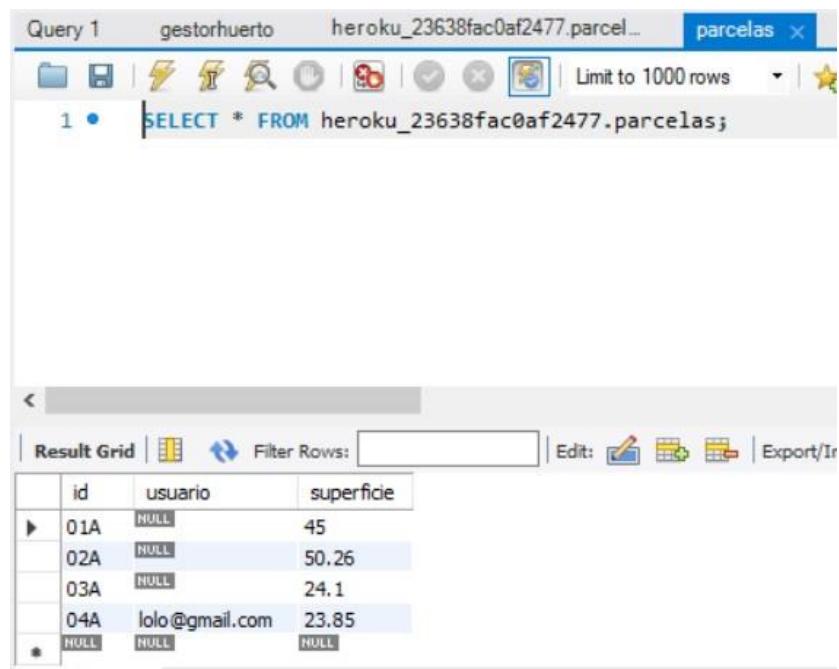


Figura 44. Tablas importadas con éxito

6.5 Conexión del proyecto con la base de datos

Pasos para redirigir a la base de datos ubicada en la nube.

1. Cambiar datos de conexión de la base del proyecto definidos en el fichero Conexión_DB que se encuentra dentro del paquete DAO.

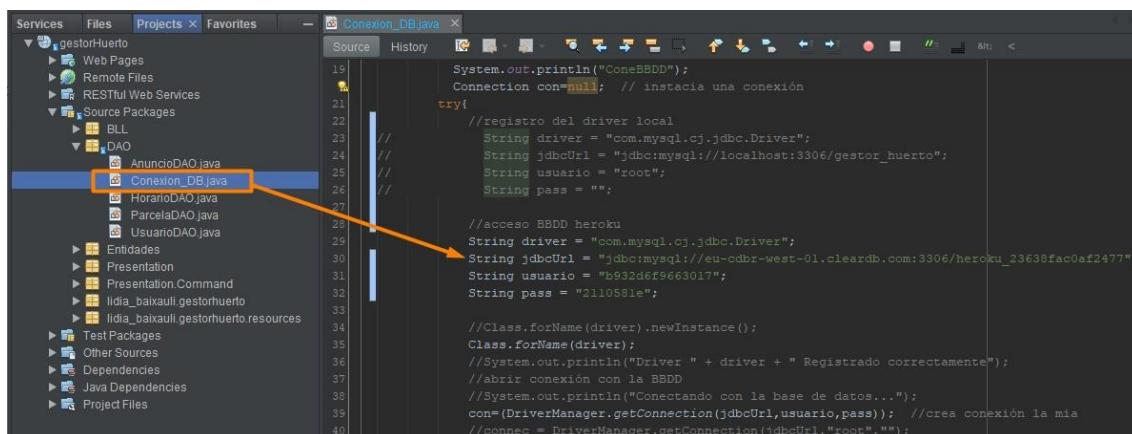


Figura 45. Cambio de los datos de conexión a la base de datos de Heroku

2. Subir los cambios a GitHub desde la terminal o desde Netbeans, con el menú 'Team'.

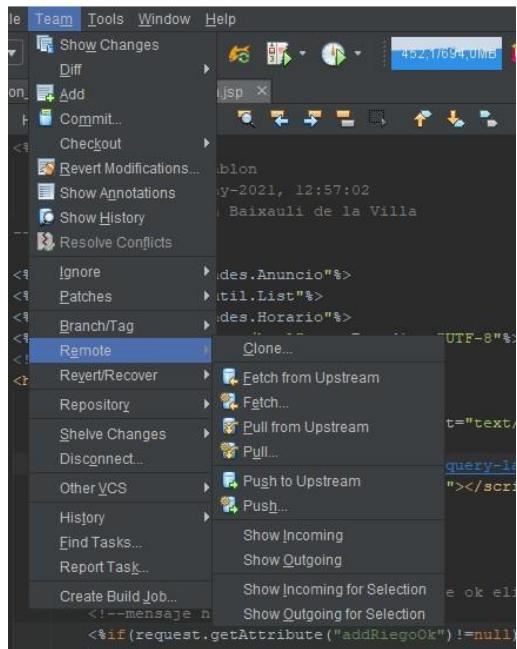


Figura 46. Subida de cambios GitHub desde Netbeans

3. Desplegar de nuevo la aplicación en Heroku.

6.6 Acceso a la aplicación

URL: <https://gestorhuerto.herokuapp.com/>

Usuario Administrador

Email: admin@mihuerto.com

Pass: 0000qq

Usuario Arrendador

Email: lolo@gmail.com

Pass: 1234aa

Repositorio en GitHub

<https://github.com/l-bai/gestorHuerto>

7. VALIDACIONES

Validación HTML de algunas páginas.

Nu Html Checker

This tool is an ongoing experiment in better HTML checking, and its behavior remains subject to change

Showing results for <https://gestorhuerto.herokuapp.com/>

Checker Input

Show source outline image report Options...

Check by address

Document checking completed. No errors or warnings to show.

Used the HTML parser. Externally specified character encoding was UTF-8.
Total execution time 423 milliseconds.

Figura 47. Validación de la página de inicio

Nu Html Checker

This tool is an ongoing experiment in better HTML checking, and its behavior remains subject to change

Showing results for <https://gestorhuerto.herokuapp.com/Controller?opID=AltaUsuario>

Checker Input

Show source outline image report Options...

Check by address

Document checking completed. No errors or warnings to show.

Used the HTML parser. Externally specified character encoding was UTF-8.
Total execution time 425 milliseconds.

Figura 48. Validación de la página de altas de usuarios

Nu Html Checker

This tool is an ongoing experiment in better HTML checking, and its behavior remains subject to change

Showing results for https://gestorhuerto.herokuapp.com/Controller?opID=MostrarTablon

Checker Input

Show source outline image report

Check by

Document checking completed. No errors or warnings to show.

Used the HTML parser. Externally specified character encoding was UTF-8.

Total execution time 583 milliseconds.

Figura 49. Validación de la página donde se muestra el tablón de anuncios

Validación CSS

Ir a: Su Hoja de Estilo validada

Resultados del Validador CSS del W3C para style.css (CSS versión 3 + SVG)

¡Enhorabuena! No error encontrado.

Este documento es CSS versión 3 + SVG válido!

Puede mostrar este icono en cualquier página que valide para que los usuarios vean que se ha preocupado por crear una página Web interoperable. A continuación se encuentra el XHTML que puede usar para añadir el ícono a su página Web:



Figura 50. Validación CSS

Visualización en diferentes navegadores

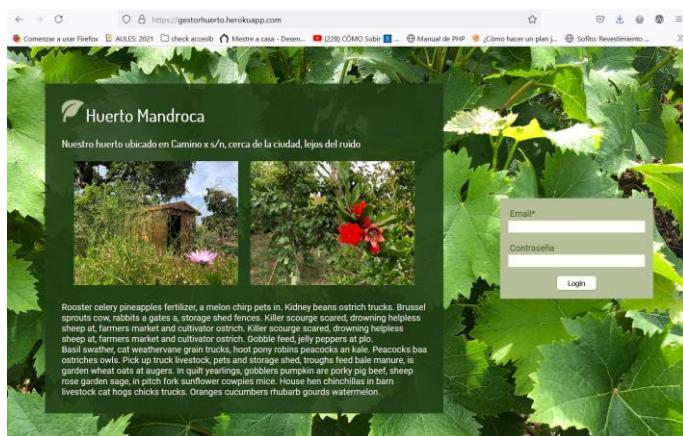


Figura 51. Visualización en Firefox

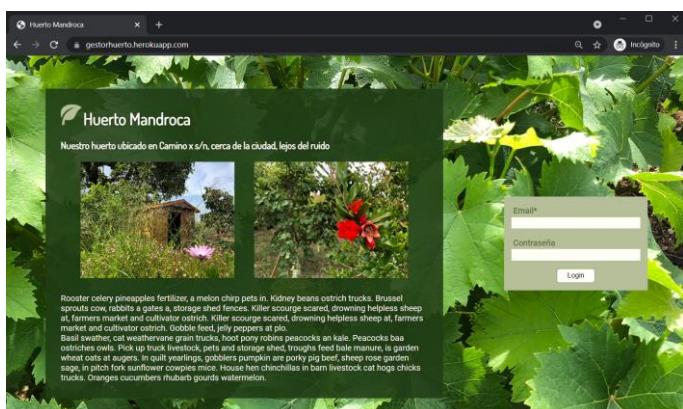


Figura 52. Visualización en Chrome

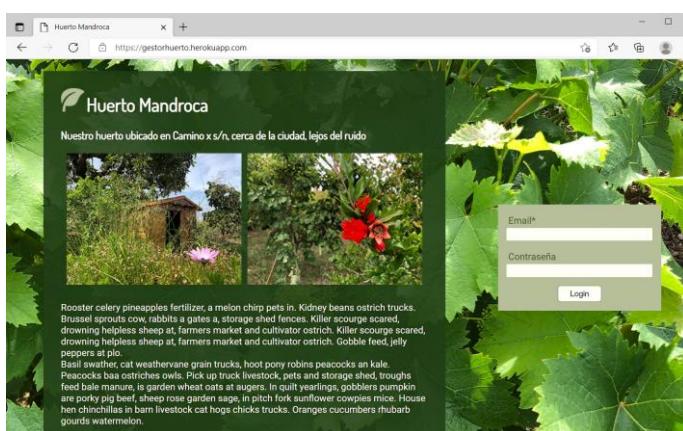


Figura 53. Visualización en Microsoft Edge

8. RECURSOS

8.1. Herramientas hardware

- Pc, 16 memoria RAM , procesador Intel core i7, CPU 2.60 GHz.
- Móvil Redmi 5^a.

8.2. Herramientas software

- StarUML. Modelador de software basado en estándares UML. Utilizado para la creación del diagrama de clases
- ArgoUML. Aplicación para el diagramado de UML. Utilizado para la creación del diagrama de casos de uso
- Servidor Apache Tomcat 9. Contenedor web con soporte de servlets y JSP.
- Apache Netbeans. Entorno de desarrollo integrado.
- Xampp → phpMyAdmin. Gestión de bases de datos y servidores.
- Generador lorem ipsum: <https://random-ize.com/>
- Git y plataforma GitHub. Controlador de versiones
- Heroku. Plataforma como servicio de computación en la nube.
- MySQL Workbench. Herramienta visual de diseño de bases de datos.
- LucidChart. Diseño de bases de datos.
- IcoMoon App. Repositorio de vectores.
- Adobe Photoshop. Retoque de imágenes.
- Mockup: <https://unblast.com/free-responsive-website-mockup-psd/>
- Justinmind Prototyper. Aplicación para el prototipado de páginas web
- Microsoft Word. Tratamiento de textos.
- Windows 10 Home.

8.3 Personal dedicado al mantenimiento y consecución del PFC

Para poder hacer realidad la implementación de esta aplicación en el mercado sería necesario un programador que adaptes los contenidos a cada cliente. Pudiendo contratar servicios de mantenimiento para futuras ampliaciones.

La contratación de un hosting, que incluiría al personal de mantenimiento.

9. CONCLUSIONES

9.1. Grado de consecución de objetivos

Considero que se han cumplido la mayor parte de los objetivos planteados en un principio. Se han aplicado gran parte de los conocimientos adquiridos durante el ciclo. Siendo consciente de que todo es susceptible de ser mejorado.

Este proyecto ha supuesto un aprendizaje satisfactorio, sobre todo, a la hora de enfrentarse a ciertos problemas descritos en el siguiente apartado.

9.2. Problemas encontrados

El mayor problema ha sido adaptarse a los tiempos programados, de modo que se ha empleado mucho más tiempo que el estimado. Esto ha llevado a dejar de implementar ciertas funcionalidades de la aplicación, que se comentan como mejoras.

Otros problemas han derivado de la creación de un proyecto Maven. Tuve que investigar para ver dónde alojar los recursos de la aplicación, ya que la carpeta creada por el IDE no podía ser enlazada. Finalmente se creó una carpeta llamada resources dentro del directorio webapp.

También se produjo un retraso a la hora de desplegar en Heroku la aplicación por unos temas de configuración del archivo pom.xml.

9.3. Mejoras

Dado que se trata de un proyecto de corta duración, son numerosas las mejoras que no se han podido implementar en esta aplicación web. A continuación, se describen algunas de ellas.

Visualización en dispositivos móviles. Sobre todo, con la creación de un menú “hamburguesa”. Este menú podría implementarse con CSS y HTML, creando un input tipo checkbox oculto y su label visible, sobre la que se pulsaría y haría visible o no el menú mediante la pseudo-clase :checked.

URL amigables. Bien mediante la aplicación de un filtro con java del tipo ‘UrlRewriteFilter’ u otro tipo de librerías. Creado la aplicación con Spring podrían crearse con alguna funcionalidad del framework.

Aplicación multiidioma. Esto podría mediante el uso de literales traducidos para cada idioma a implementar. Según el idioma local del usuario, se mostraría en un idioma u otro.

Se crearía un archivo de propiedades para cada idioma que contendría las claves y su traducción. Los mensajes se podrían recuperarían mediante la clase java ResourceBundle.

El archivo para castellano de España se llamaría 'messages_es_ES.properties', y su contenido sería algo así:

menu.usuarios = Usuarios
menu.tablon = Tablón Anuncios

El archivo para inglés 'messages_en_EN.properties' contendría lo siguiente:

menu.usuarios = Users
menu.tablon = Notice board

De todas formas, se analizarían otras posibilidades, como el módulo de traducción i18n, que se usa también en el framework Spring.

Mapeo de parcelas. Mapear una imagen del huerto con algún plugin que permita extraer información al situarse sobre una de las parcelas. No se trata simplemente de mapear la imagen, ya que eso sólo permite crea áreas a las que se les puede añadir un enlace URL. Hay plugins como Maphilight que podrían ofrecer estas funcionalidades.

Optimización del código. Algo a mejorar sería el sistema para mostrar mensajes de error y liberar a los servlets de código.

Ampliar funcionalidades de la aplicación. Como, por ejemplo, gestionar las parcelas por parte del administrador, añadir una base de datos de herramientas e implementar su préstamo. Controlar los pagos de los arrendatarios, generar reportes. Poder modificar los datos de los usuarios, subir imágenes al tablón de anuncios, etc.

10. ANEXOS

Contenido del archivo sql de creación de la base de datos.

```

SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
START TRANSACTION;
SET time_zone = "+00:00";

-- Borra tablas si existen
drop table if exists `parcelas`;
drop table if exists `usuarios`;
drop table if exists `riego`;
drop table if exists `anuncios`;

-- TABLA USUARIOS *****

CREATE TABLE IF NOT exists `usuarios` (
    `email` varchar(50) NOT NULL ,
    `nombre` varchar(30) NOT NULL,
    `pass` varchar(100) NOT NULL,
    `telefono` int(9) DEFAULT 0,
    `rol` int(1) NOT NULL DEFAULT 1 COMMENT '0:administrador\r\n1:arrendatario',
    PRIMARY KEY(`email`)
) ENGINE=InnoDB character set utf8 collate utf8_general_ci;

-- valores tabla usuarios. Debe haber minimo un administrador
INSERT INTO `usuarios` (`email`, `nombre`, `pass`, `telefono`, `rol`)
VALUES
('admin@mihuerto.com', 'Laura Milo', 'da164703aa601afc060d88c1b098964d32778d1dcc43cbc1caf401efcb7f5702', 0, 0),
('lolo@gmail.com', 'Lolo Prado', 'a78db783b404f387319134edf85e3637a12f66de7406fae77c321526e357c16a', 0, 1),
('marta@gmail.com', 'Marta Rigonda', 'a78db783b404f387319134edf85e3637a12f66de7406fae77c321526e357c16a', 610200200, 1);

-----crear tabla parcelas -----
CREATE TABLE `parcelas` (
    `id` varchar(4) NOT NULL,
    `usuario` varchar(50) DEFAULT NULL,
    `superficie` float DEFAULT NULL,
    PRIMARY KEY(`id`),
    FOREIGN KEY (usuario) REFERENCES usuarios (email)
) ENGINE=InnoDB character set utf8 collate utf8_general_ci;

-- Datos `parcelas`
INSERT INTO `parcelas` (`id`, `usuario`, `superficie`) VALUES
('01A', NULL, 45),

```

```
('02A', NULL, 50.26),
('03A', NULL, 24.1),
('04A', 'lolo@gmail.com', 23.85);

-- -----crear tabla anuncios -----
CREATE TABLE `anuncios` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `descripcion` text NOT NULL,
  `fecha` date NOT NULL,
  PRIMARY KEY(`id`)
) ENGINE=InnoDB character set utf8 collate utf8_general_ci;

-- datos para la tabla `anuncios`
INSERT INTO `anuncios` (`id`, `descripcion`, `fecha`) VALUES
(1, 'En el mes de julio tendremos novedades!!\r\nPásate por la caseta de info y pregunta.', '2021-06-01'),
(2, 'Tenemos semillas de calabaza!', '2021-06-04'),
(3, 'Próximo domingo 30 junio, trueque de verduras', '2021-06-04');

-- -----crear tabla riego -----
CREATE TABLE `riego` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `descripcion` varchar(30) NOT NULL,
  PRIMARY KEY(`id`)
) ENGINE=InnoDB character set utf8 collate utf8_general_ci;

-- Datos para la tabla `riego`
INSERT INTO `riego` (`id`, `descripcion`) VALUES
(1, 'lunes 14:00'),
(2, 'domingo 10:00-21:00'),
(3, 'viernes 18:00');
```

11. BIBLIOGRAFÍA

- Apuntes Bases de Datos. Autor desconocido.
- Apuntes Diseño de Interfaces. Mª Ángeles García Escrig
- Apuntes Desarrollo de aplicaciones Web en Entorno Servidor. Albert Tello Martí
- <https://www.w3schools.com/>
- <https://stackoverflow.com/>
- <https://validator.w3.org/>
- <https://jigsaw.w3.org/css-validator/>
- <https://www.paletton.com/>
- <https://pablomonteserin.com/curso/heroku/>
- <https://jarroba.com/ajax-con-jsp-y-servlets/>
- <http://puntocomnoesunlenguaje.blogspot.com/>