

Universidad de Valladolid

ESCUELA DE INGENIERÍA INFORMÁTICA



DISEÑO DE PROYECTO PYDSC2022

PLANIFICACIÓN Y DISEÑO DE SISTEMAS COMPUTACIONALES

Jhon Steeven Cabanilla Alvarado

Miguel Chaveinte García

Carlos Martín Sanz

Alejandro Pulido Sánchez

Héctor Toribio González

Índice

1. Introducción	3
2. Documentación de decisiones	3
2.1. Métodos, herramientas y técnicas	3
3. Arquitectura lógica	4
3.1. Modules and Style General	5
3.2. Uses Style	6
3.3. Inheritance Style	7
3.4. Components and Connectors	8
3.5. Diagrama de despliegue	9
4. Caso de Uso “Consultar Alojamientos Disponibles”	10
4.1. Diagrama de clases	10
4.2. Diagrama de secuencia principal	11
4.3. Obtener alojamientos disponibles	12
4.4. Consultar Servicios	13
5. Caso de uso “Registrar nuevos precios”	14
5.1. Diagrama de Clases	14
5.2. Diagrama de secuencia principal	15
6. Interfaz	16
6.1. Menú principal	16
6.2. Log in	17
6.3. Logueado	17
6.4. Alojamientos disponibles	18
6.5. Mostrar información - Alojamientos disponibles	18
6.6. Reservar alojamiento	19
6.7. Mostrar información - Reservar alojamiento	20
6.8. Confirmar reserva - Reservar alojamiento	20
6.9. Registrar nuevos precios	21

6.10. Modificar precios	21
7. Diseño base de datos	22
7.1. Modelo de dominio reducido	22
7.2. Diagrama entidad relación	23
7.3. Diagrama relacional	24
7.4. Revisión final y script	26
8. Implementación	26
9. Test y pruebas de la implementación	27
9.1. Caso de uso “IDENTIFICARSE”	27
9.2. Caso de uso “CONSULTAR ALOJAMIENTOS”	28
9.3. Caso de uso “REGISTRAR NUEVOS PRECIOS”	29
9.4. Caso de uso “REALIZAR RESERVA”	31
10.Herramientas y bibliografía	32

1. Introducción

En este documento se presenta el desarrollo del diseño del proyecto *Vacation As Home*. Contiene las decisiones del entorno tecnológico, los diagramas pertinentes, la explicación de la interfaz de la aplicación, sus mockups y detalles sobre la implementación relacionados con los casos de uso planteados en el enunciado.

2. Documentación de decisiones

Las decisiones del entorno tecnológico para este proyecto son las siguientes:

2.1. Métodos, herramientas y técnicas

El sistema se basará en una aplicación web accesible desde internet, por lo cual el conjunto de tecnologías que se usarán en el desarrollo estarán orientadas a dicho propósito. A continuación se describe el conjunto de lenguajes y frameworks utilizados:

- **HTML:** Hace referencia al lenguaje de marcado para la elaboración de páginas web. Es un estándar que sirve de referencia del software que conecta con la elaboración de páginas web en sus diferentes versiones, define una estructura básica y un código (denominado código HTML) para la definición de contenido de una página web, como texto, imágenes, vídeos, juegos, entre otros.
- **CSS:** CSS es un lenguaje usado para definir y crear la presentación de un documento estructurado escrito en HTML.
- **Javascript :** Javascript es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico. Se utiliza principalmente en su forma del lado del cliente, implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas⁴ aunque existe una forma de JavaScript del lado del servidor.
- **Java Servlet:** Es una tecnología de servidor utilizada en el desarrollo de aplicaciones web para procesar y gestionar las solicitudes de los usuarios y proporcionar una respuesta adecuada. Los servlets son programas Java que se ejecutan en un servidor web y se utilizan para manejar las peticiones de los clientes y proporcionar una respuesta. Uno de los principales beneficios de los servlets es que se ejecutan en el servidor, lo que permite a las aplicaciones web procesar datos y lógica de negocio en el lado del servidor en lugar de en el lado del cliente. Los servlets también se integran fácilmente con otros componentes del servidor, como bases de datos y archivos, lo que permite a las aplicaciones web acceder a estos recursos de manera sencilla. Los servlets son especialmente útiles para aplicaciones web que requieren un procesamiento intensivo en el servidor o el acceso a recursos del servidor, como bases de datos o archivos.
- **Sistema gestor de bases de datos.** Al elegir Java como lenguaje de programación, optamos por emplear MySQL como sistema gestor de bases de datos. Se trata de un sistema relacional de licencia pública y comercial (Oracle). Es sencillo de usar, soporta múltiples plataformas y tanto la instalación como configuración no llevan mucho tiempo.

- **Bibliotecas.** Usaremos las bibliotecas básicas de Java, `java.sql` para las bases de datos y conexiones con los modelos, `javax.javaee-web-api` para las interfaces y `javax.servlet` para los controladores puedan interactuar con las *response/request*.
- **Plataformas de desarrollo.** Usaremos **NetBeans** como IDE para programar la interfaz gráfica, ya que conocemos que la integración de las tecnologías en su plataforma es adecuada. En cuanto a la herramienta escogida como control de versiones se ha decidido utilizar **Gitlab** de la escuela por la facilidad de uso y robustez que proporciona a la hora de desarrollo informático colaborativo.

3. Arquitectura lógica

En esta sección mostraremos algunos diagramas en los que mostramos la arquitectura lógica del sistema, usando diversos diagramas y estilos dados en clase.

En la figura 1 (arquitectura genérica) presentamos el patrón que hemos elegido de manera más abstracta y sin entrar en detalles.

Para la elaboración de la descomposición se siguió el patrón Modelo-Vista-Controlador (MVC), con un modelo pasivo. En este patrón, la vista se encarga de mostrar los datos y recoger los datos introducidos por el usuario. El controlador valida los datos de la vista y con ellos modifica el modelo e informa a la vista que el modelo ha cambiado y que debe ser “refrescada”. Por último el controlador solo se encarga de introducirlos o recuperar los datos almacenados en la Base de Datos, cabe decir que el modelo, al ser pasivo no puede notificar cambios de estado.

Como se puede ver, nosotros hemos escogido implementar el sistema usando un patrón Modelo-Vista-Controlador por capas (Layered MVC). Tenemos cuatro capas:

- **Capa Vista «capa estricta»:** es la capa encargada de almacenar las vistas y los controladores para dichas vistas del patrón MVC.
- **Capa controlador «capa estricta»:** esta capa se encarga de almacenar los controladores para los diferentes casos de uso, que son como hemos dicho antes, servlets.
- **Capa Persistencia «capa estricta»:** en esta capa hemos decidido implementar un patrón *Data Access Object* (DAO) que, junto con *Data Transfer Object* que se presentarán en la siguiente capa, para realizar la comunicación con la base de datos. Esta comunicación se lleva a cabo mediante su conexión con el paquete `DataAccess`. Este paquete contiene una clase `ConnectionPool`, con patrón *Singleton* que se utiliza para garantizar que una clase tenga solo una instancia y proporcionar un punto de acceso global a ella. La capa de acceso a datos se encarga de realizar las operaciones de lectura y escritura en la base de datos, mientras que el pool de conexiones gestiona un conjunto de conexiones a la base de datos que pueden ser reutilizadas por la aplicación en lugar de crear una nueva conexión cada vez que se necesita acceder a la base de datos. Esto permite reducir la sobrecarga en la base de datos y mejorar el rendimiento de la aplicación.
- **Capa Utils «capa relajada»:** en esta capa metemos los *Data Transfer Object* para transferir datos entre la capa de acceso a datos (DAO) y la capa de presentación, paso previo por los controladores. Los DTOs se pueden utilizar como modelos para representar los datos que se muestran al usuario, y se ubican en un paquete “util” para que puedan ser accedidos por todas las capas de la aplicación.

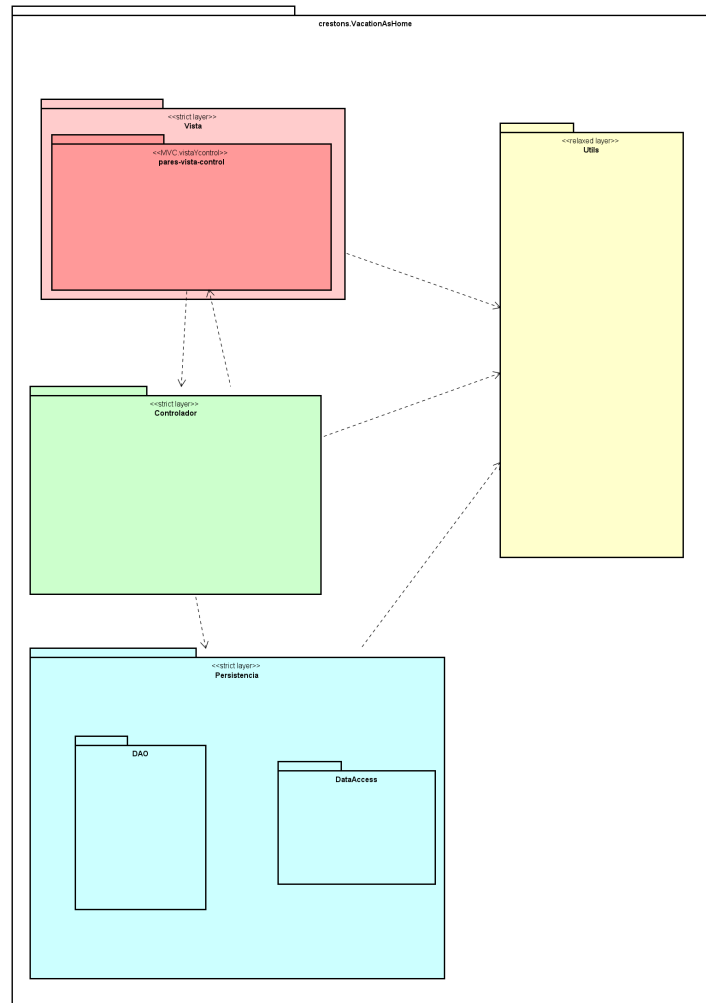


Figura 1: Diagrama de despliegue

3.1. Modules and Style General

En la figura 2 aparece el diagrama del *Modules and Style General*. En él podemos ver más detalladamente la descomposición de los módulos del sistema propuestos anteriormente, con sus respectivas clases y paquetes. Aparecen las cuatro capas expuestas con anterioridad.

En el módulo **Vista.pares-vista-control**, hemos incluido los paquetes de cada uno de los cuatro casos de uso a desarrollar.

En el módulo **Controladores**, hemos incluido los controladores de los casos de uso utilizados. En el resto de diagramas observaremos las relaciones de estos.

En el módulo **Persistencia** hemos creado dos paquetes: **DAO** Y **DataAccess**. En el primero hemos representados las dos clases **DAO** que utilizaremos. Estas, para recuperar los datos de la Base de Datos, harán uso de la clase *Singleton Connection Pool* que es la que gestionará las conexiones.

En el módulo **Utils**, hemos incluido los **DTO** como modelos que utilizarán el resto de paquetes.

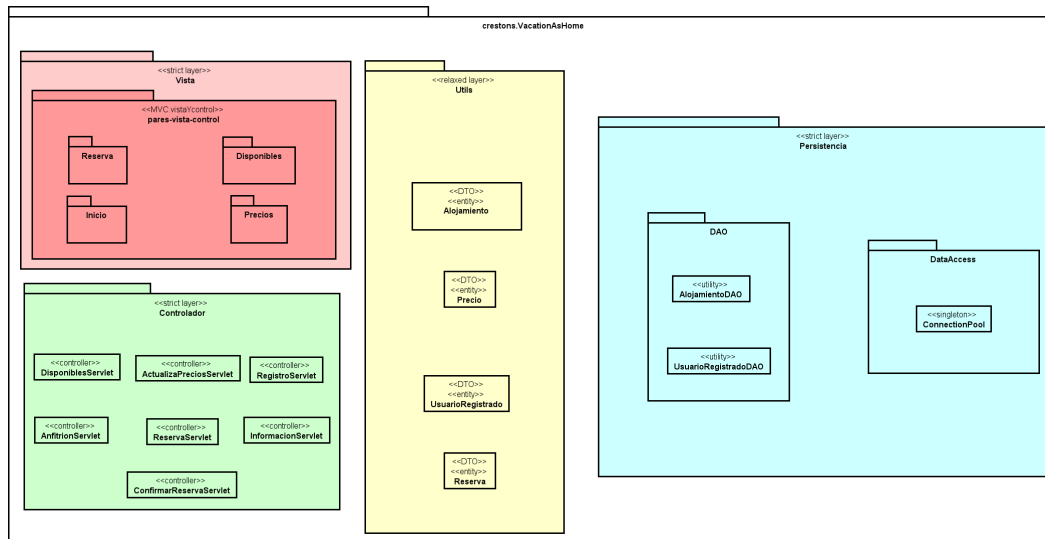


Figura 2: Diagrama de despliegue

3.2. Uses Style

Para poder ver como se relacionan los diferentes módulos del sistema entre ellos, así como los módulos externos que utilizan/necesitan, hemos realizado el diagrama para el *Uses Style*.

Para las bases de datos y conexiones con los modelos hemos utilizado `java.sql`, `javax.javaee-web-api`, basada en Java EE (Enterprise Edition), para las interfaces, el control de la sesión y el acceso a datos persistentes; y `javax.servlet` para los controladores puedan interactuar con las *response/request*.

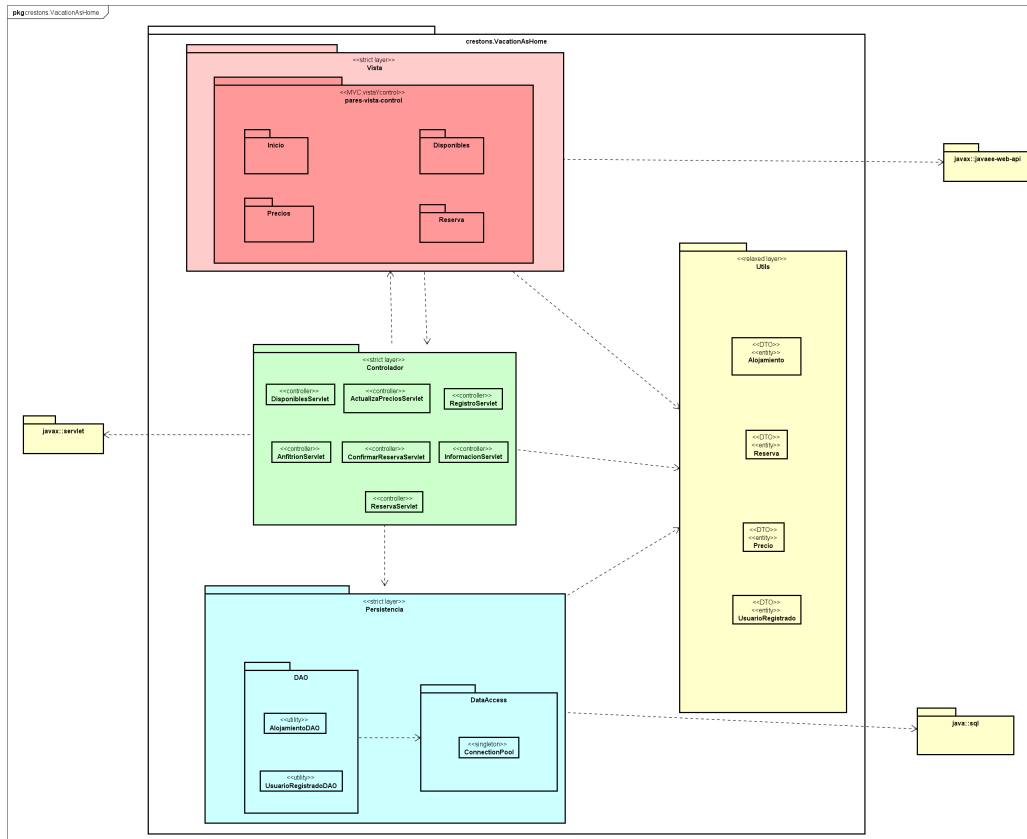


Figura 3: Uses Style General

3.3. Inheritance Style

En el *Inheritance Style*, en su diagrama, se detallan las herencias de las clases del sistema.

En nuestro caso, la herencia viene dada por `HttpServlet` es una clase abstracta que se encuentra en el paquete `javax.servlet`. `HttpServlet` proporciona una implementación básica de un `Servlet` que maneja solicitudes HTTP. Es abstracta porque requiere que implementes algunos métodos abstractos que deben ser implementados por las subclases.

`HttpServlet` proporciona varios métodos para manejar diferentes tipos de solicitudes HTTP, como GET o POST. Estos métodos incluyen:

- **doGet:** Este método se llama cuando se envía una solicitud GET a un servlet.
- **doPost:** Este método se llama cuando se envía una solicitud POST a un servlet.

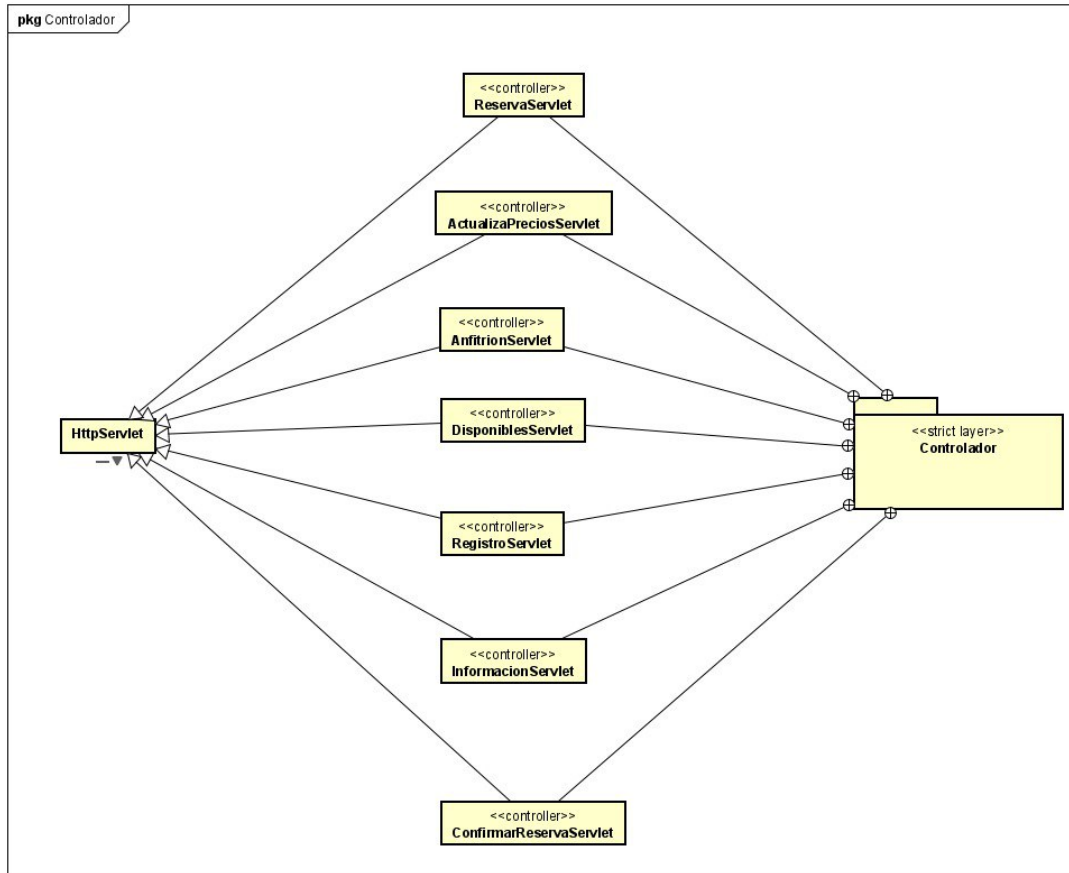


Figura 4: Diagrama de herencia

3.4. Components and Connectors

En nuestro caso hemos visto que el sistema que hemos diseñado puede dividir sus componentes de maneras distintas.

- Sistema distribuido en un par cliente-servidor con el cliente encargándose de la interfaz gráfica y el servidor encargándose de la lógica de negocio. Que requiere interfaz (Request) y solicita mediante (Response) al HttpServlet, mediante http, alojado en Servidor
- Tríada MVC con acceso a la base de datos por parte del modelo.

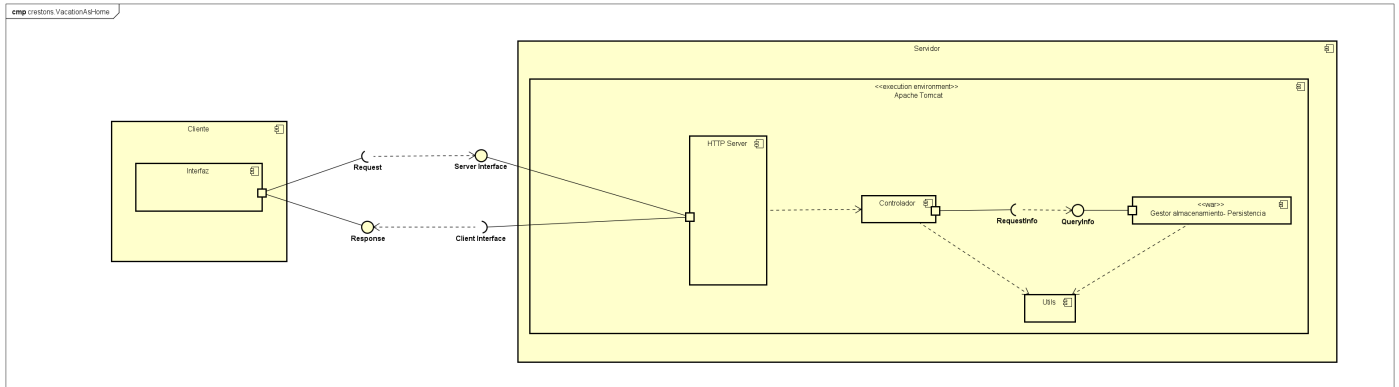


Figura 5: Components and Connectors

3.5. Diagrama de despliegue

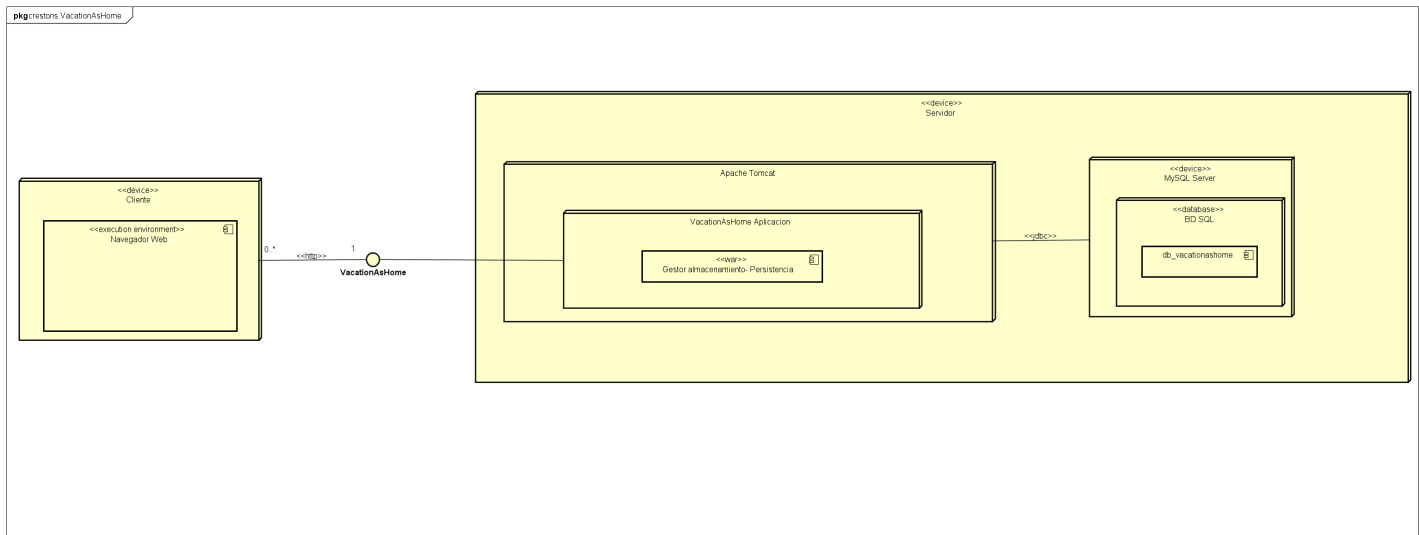


Figura 6: Diagrama de despliegue

En la Figura 6, se representa el diagrama de despliegue de la aplicación. Contamos con una arquitectura cliente-servidor. En este caso, tenemos nuestra aplicación desplegada en un Apache Tomcat, que se conecta con el servidor de MySQL para acceder a la base de datos. Por otro lado, el cliente, que puede ser un ordenador, una tablet o un teléfono inteligente, se conecta a nuestro servidor a través de un navegador web por http.

4. Caso de Uso “Consultar Alojamientos Disponibles”

4.1. Diagrama de clases

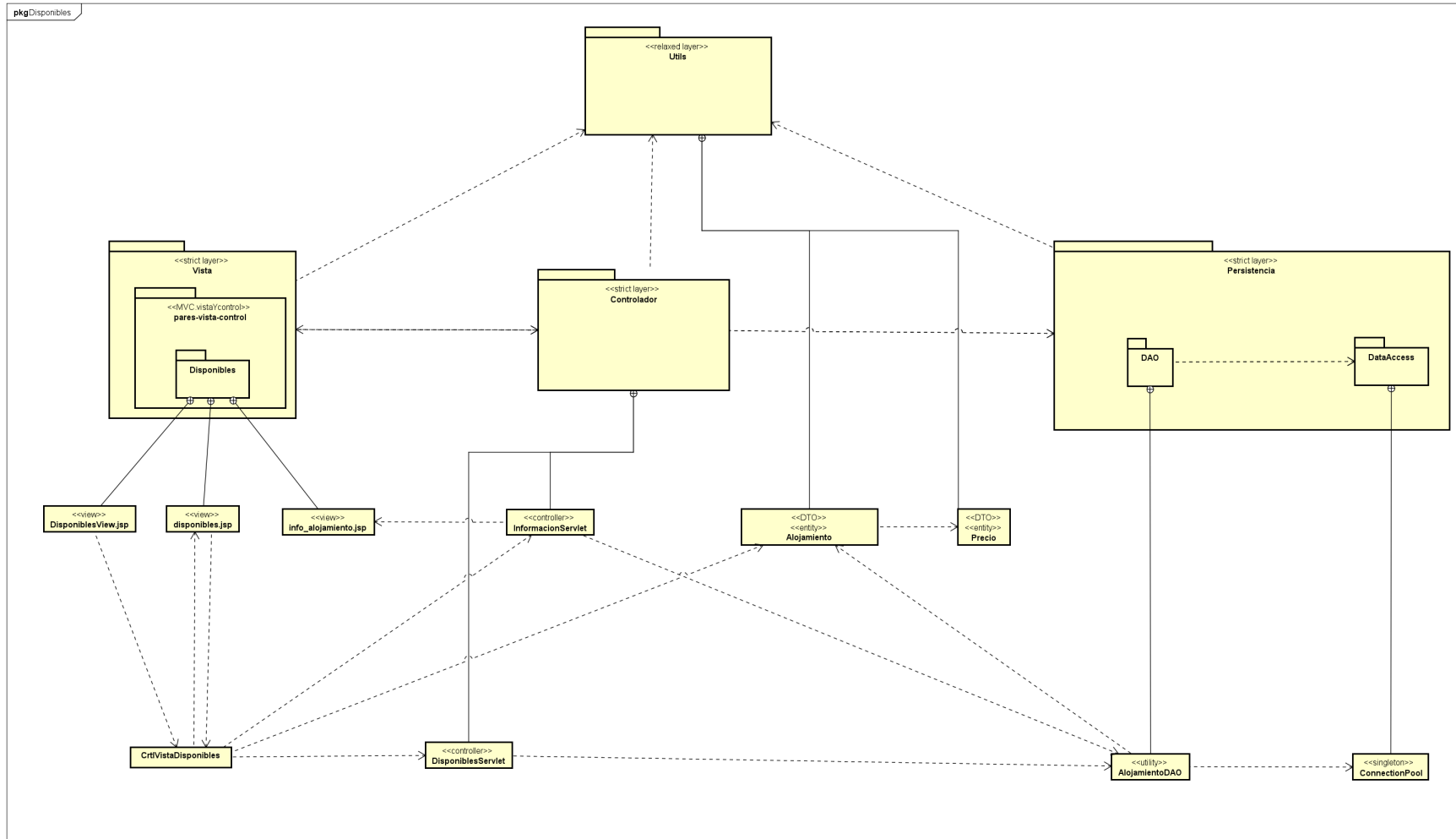
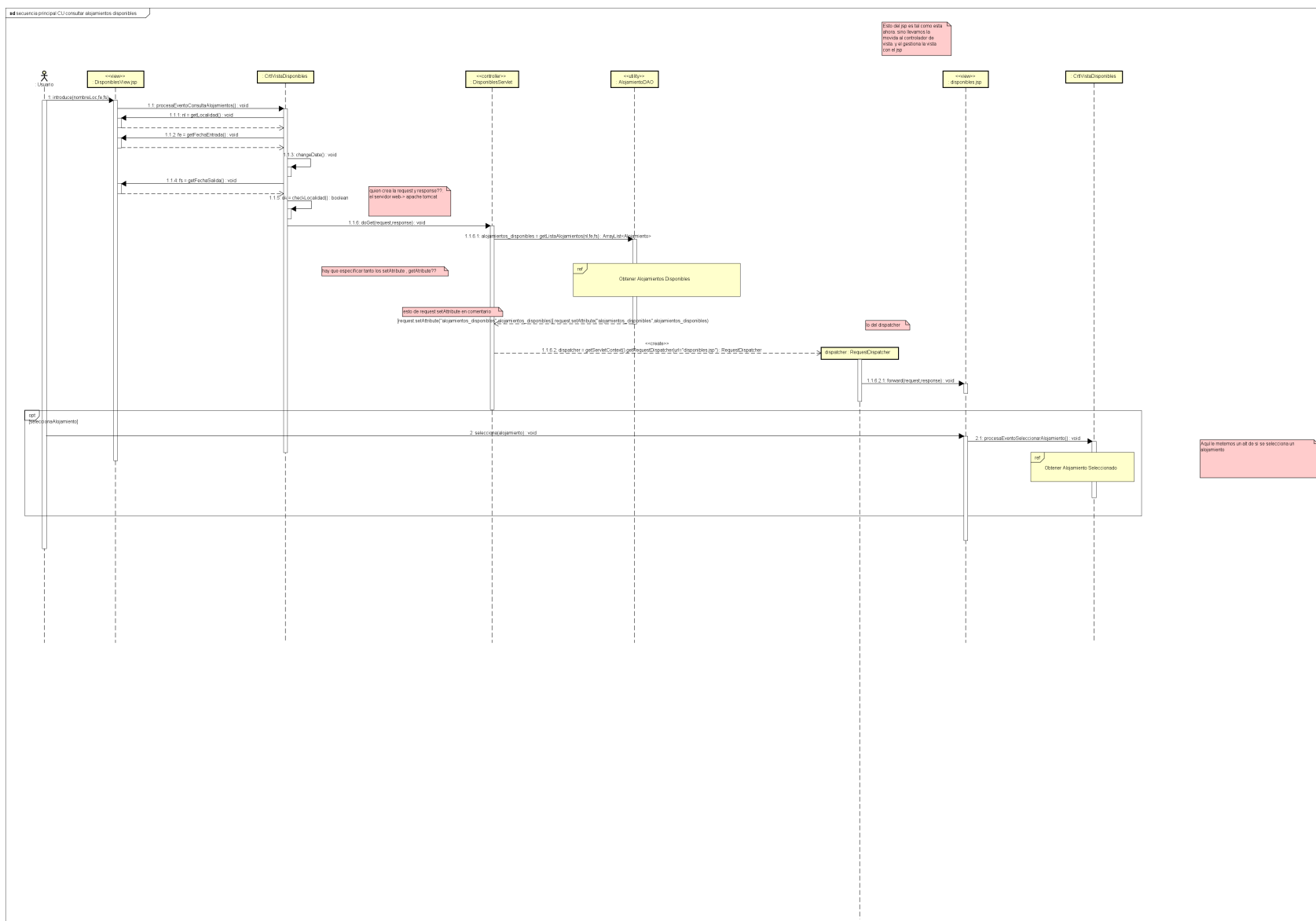


Figura 7: Diagrama Clases CUAlojamientosDisponibles

Universidad de Valladolid



4.3. Obtener alojamientos disponibles

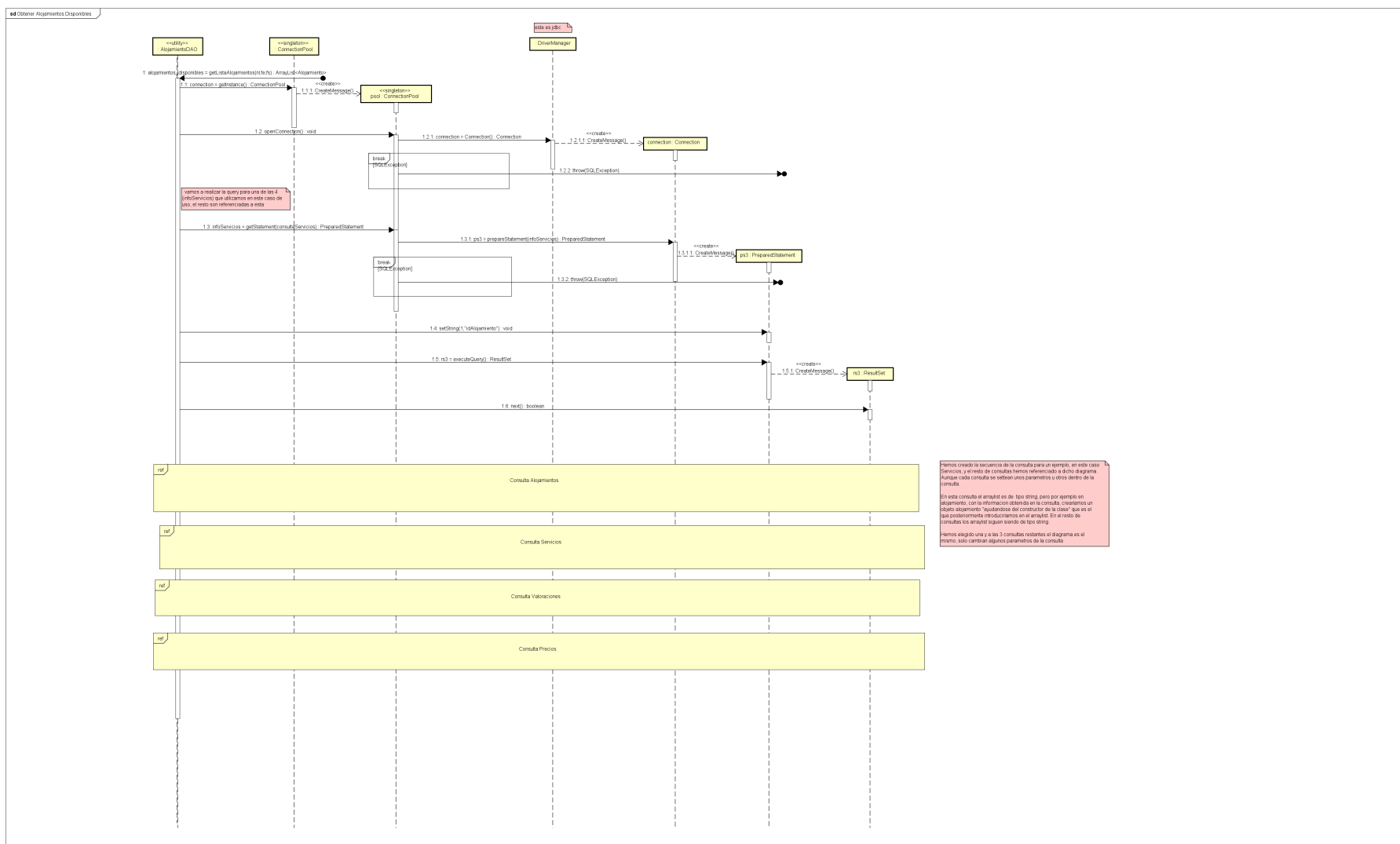


Figura 9: Obtener alojamientos disponibles

4.4. Consultar Servicios

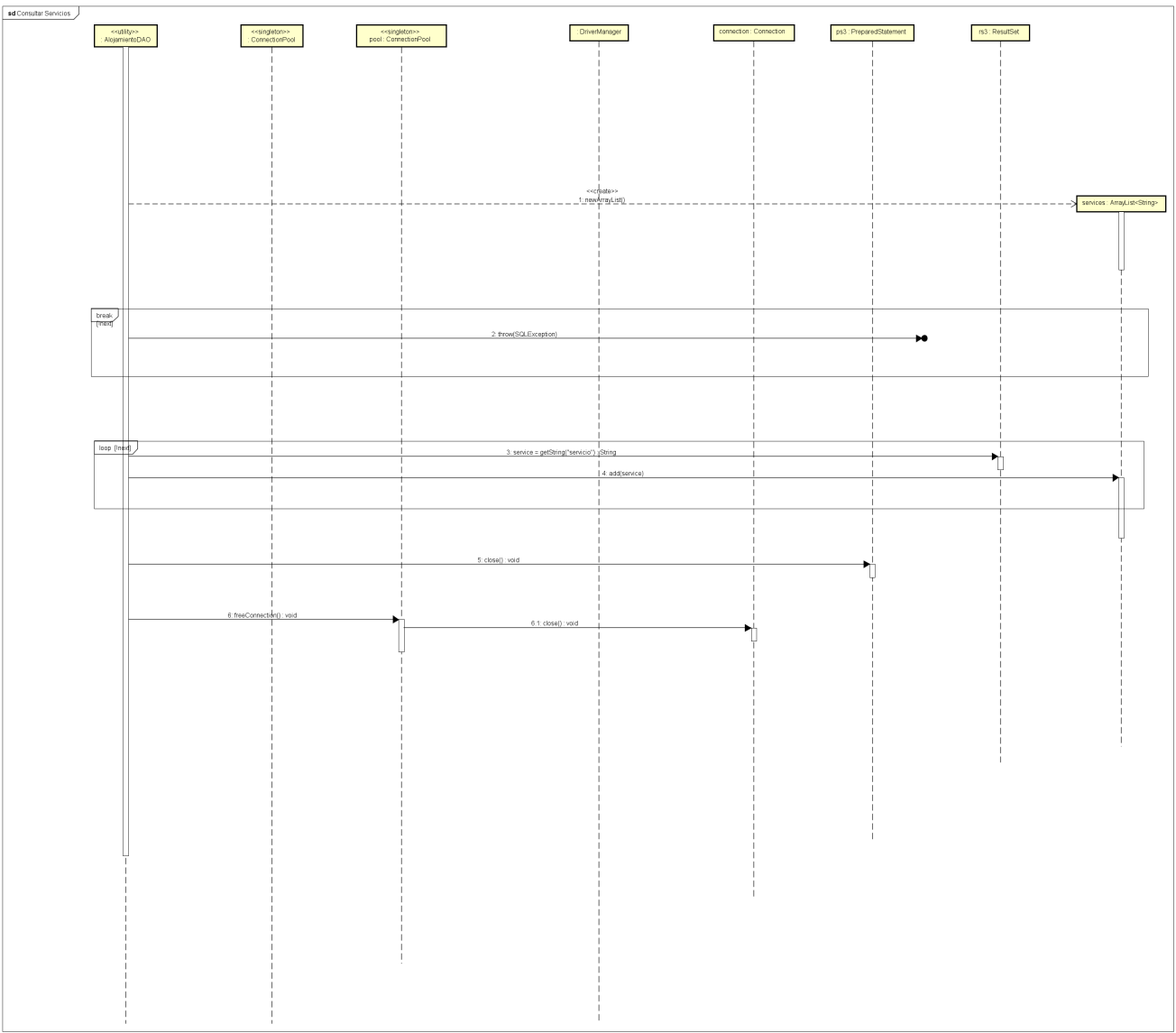


Figura 10: Consultar servicios

5. Caso de uso “Registrar nuevos precios”

5.1. Diagrama de Clases

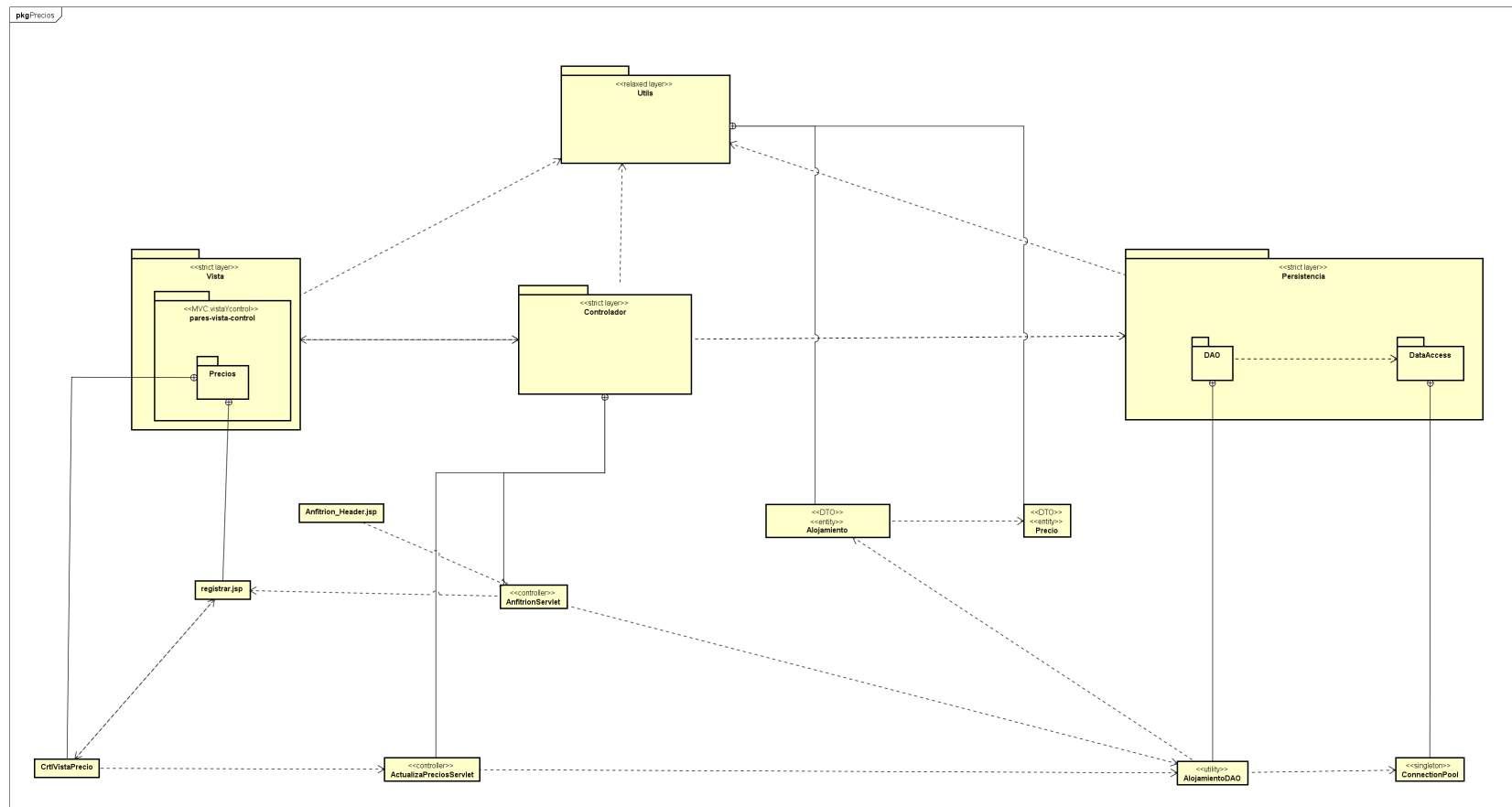


Figura 11: Diagrama de clases CURegistrarNuevosPrecios

5.2. Diagrama de secuencia principal

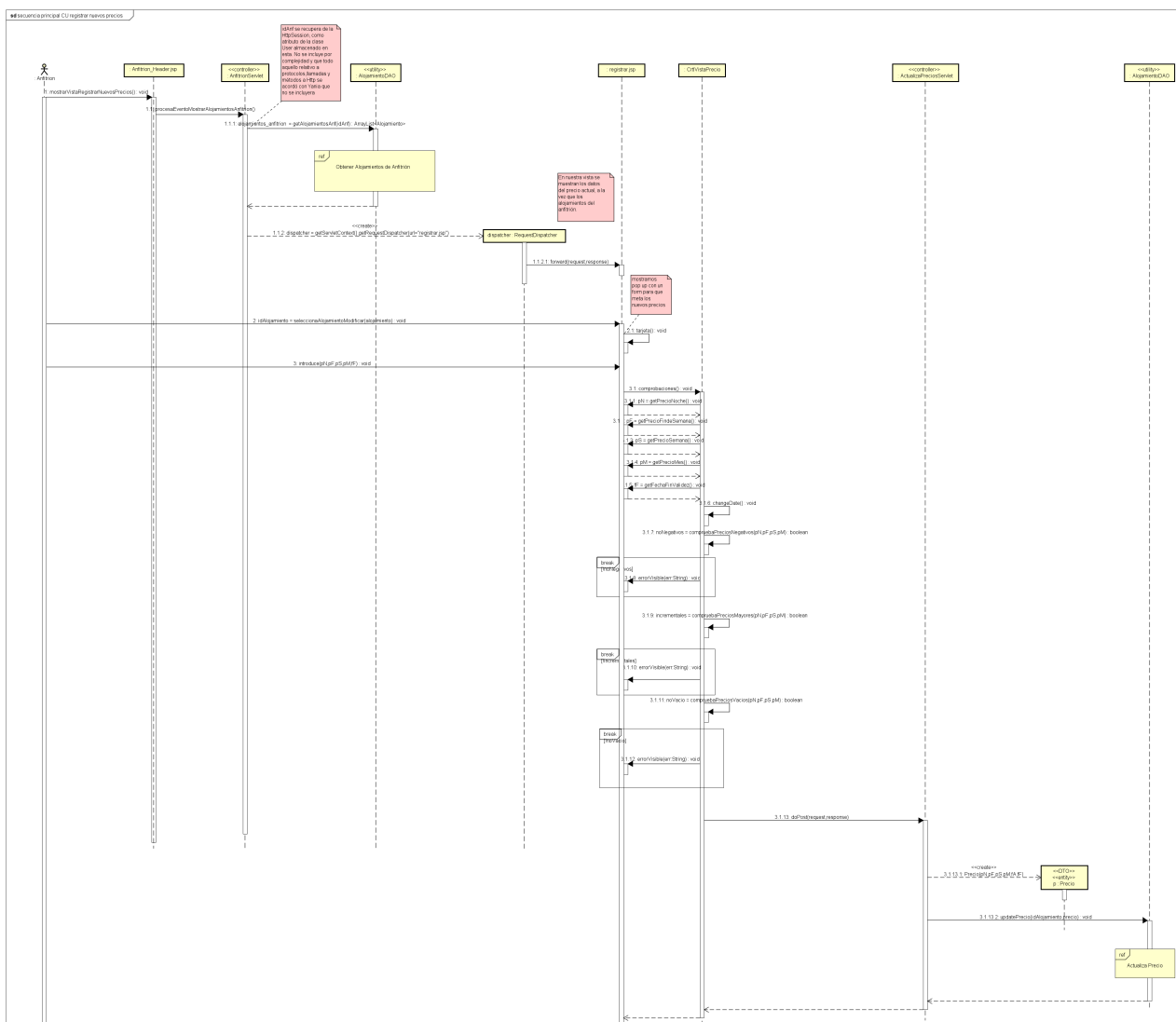


Figura 12: Diagrama de secuencia principal CURegistrarNuevosPrecios

No tenemos hemos realizado el diagrama de Obtener Alojamientos Anfitrión en este caso de uso porque se hace de la misma manera que la Obtención de Alojamientos Disponibles.

6. Interfaz

Para el posterior desarrollo de nuestra aplicación, primero hemos optado por la realización de un diseño orientativo a través de bocetos o mockups. La aplicación web que hemos utilizado para ello se llama (<https://mockups.com/>). La finalidad principal de estos es elaborar la vista de la aplicación teniendo un aspecto de referencia previo, y que la vista final presentada varíe levemente con respecto a los mockups realizados.

A continuación, los mostraremos acompañados de una breve descripción sobre el caso de uso que representan. En los bocetos únicamente hemos tratado las situaciones en las que el caso de uso se realiza exitosamente, sin fallos en las entradas, ni casos alternativos. Aunque el tratamiento de estos se podrá ver en el apartado “Test y pruebas de la implementación” donde a parte de la batería de pruebas, incluiremos capturas de la salida obtenida en caso de fallos o errores.

6.1. Menú principal

El menú principal no representa como tal ninguno de los casos de uso propuestos, pero es la primera vista o pantalla que el usuario encontrará cuando este acceda a la aplicación.

Podemos ver que se ofrecen las dos posibilidades que un usuario no registrado puede realizar, la primera *consultar los alojamientos disponibles*, y la segunda el *log in*.



Figura 13: Menú principal

6.2. Log in

Si el usuario selecciona Log in en el menú principal anterior, le llevaría a la siguiente vista. En esta podemos observar los campos que ha de rellenar para loguearse en el sistema y acceder al resto de funcionalidades.

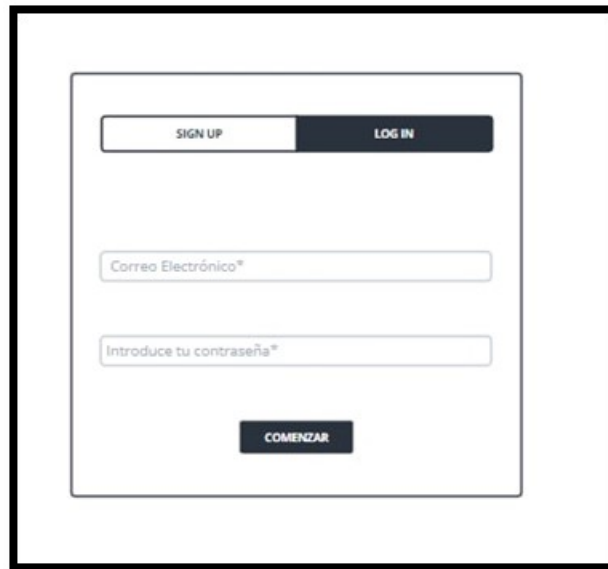
A login form interface with a white background and a black border. At the top, there are two buttons: 'SIGN UP' in a light gray box and 'LOG IN' in a dark gray box. Below these are two input fields: 'Correo Electrónico*' and 'Introduce tu contraseña*'. At the bottom, there is a dark gray button labeled 'COMENZAR'.

Figura 14: Log in

6.3. Logueado

Una vez que el usuario ha introducido correctamente sus credenciales, y por tanto, se ha logueado en la aplicación verá la siguiente vista. Similar a la del menú principal, pero con más opciones, y donde ponía log in ahora muestra un mensaje “Bienvenido”.

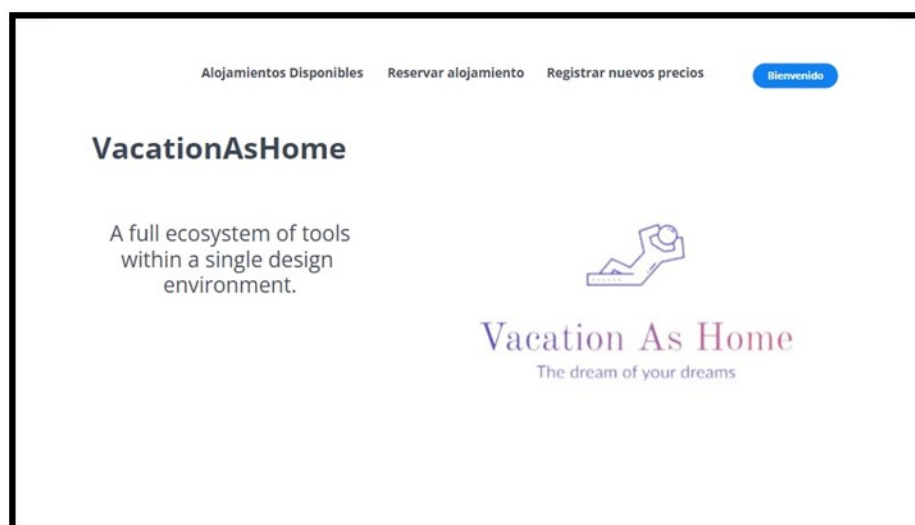


Figura 15: Logueado

6.4. Alojamientos disponibles

Esta vista es compartida tanto para un usuario registrado de la aplicación como para un usuario no logueado, ya que el sistema debe permitir a ambos consultar los alojamientos disponibles como se indica en el caso de uso.

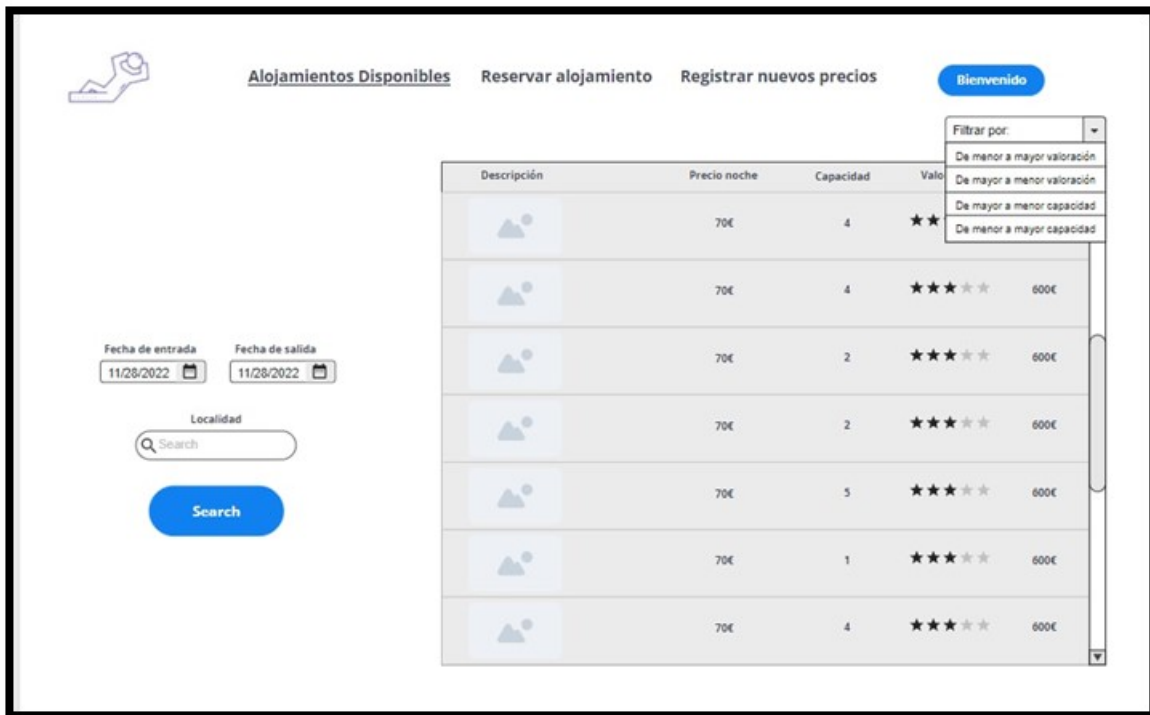


Figura 16: Alojamientos dispobles

6.5. Mostrar información - Alojamientos disponibles

Consultando los alojamientos disponibles también podremos ver más información de cada uno de ellos, clicando en la foto de dicho alojamiento.

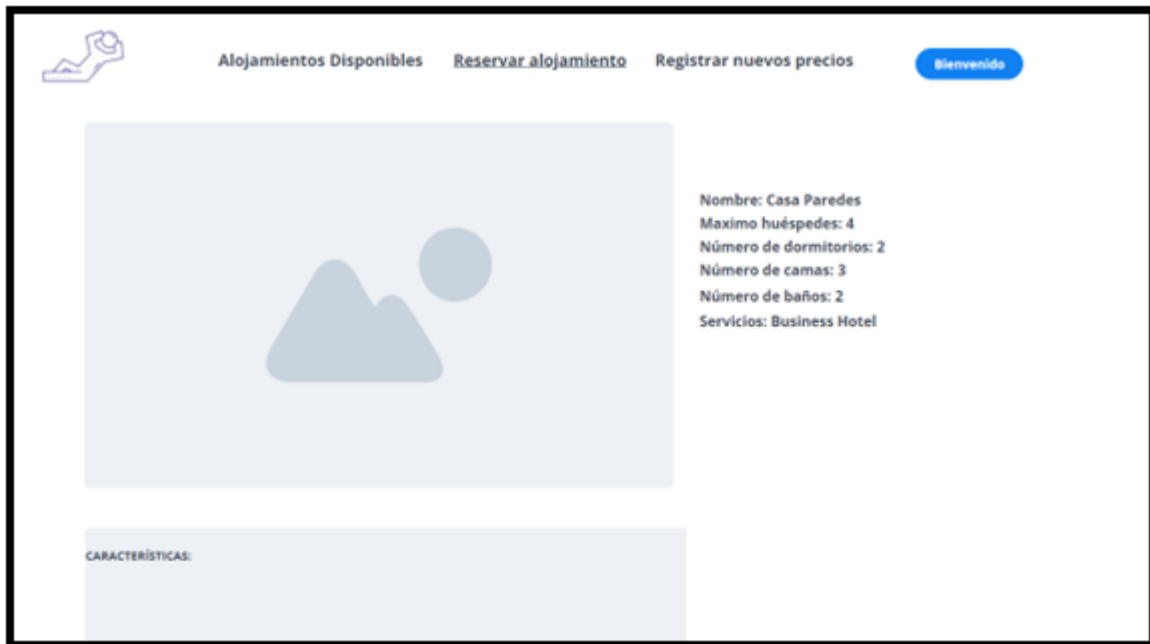


Figura 17: Información de cada alojamiento

6.6. Reservar alojamiento

Una vez el usuario se ha logueado, en las opciones de la parte superior se ve “Reservar alojamiento”. Esta vista solo es accesible para usuarios logueados.

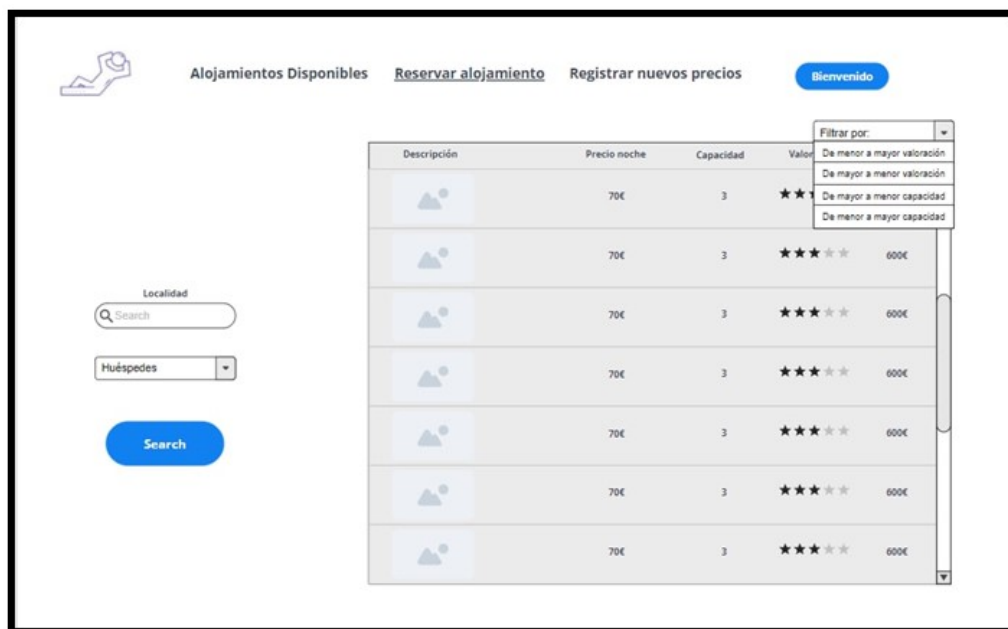


Figura 18: Reservar alojamiento

6.7. Mostrar información - Reservar alojamiento

Al igual que en “consultar alojamiento” la forma de acceder a “mostrar información” es la misma. Cuando buscamos para “reservar alojamiento” clicamos en la foto de este y nos muestra información sobre él. Nos abre esta vista con la información y la posibilidad de realizar la reserva.

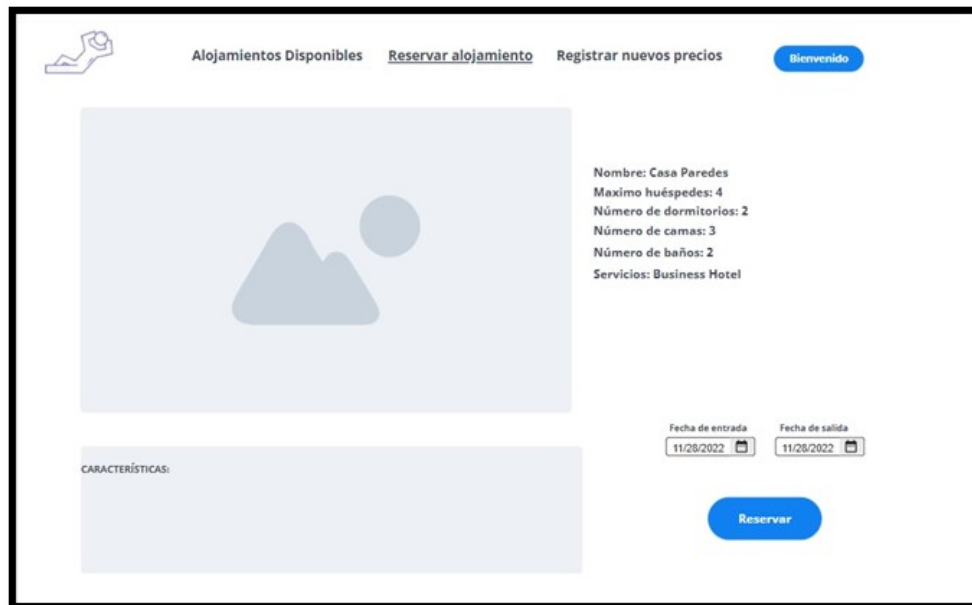


Figura 19: Información en las reservas

6.8. Confirmar reserva - Reservar alojamiento

Y tras darle en el botón de reservar del mockup anterior nos muestra el siguiente pop-up donde terminar de realizar la reserva, pudiendo ofrecer un mensaje para el anfitrión y decidir si fraccionar o no el pago.



Figura 20: Reservar alojamiento

6.9. Registrar nuevos precios

Esta vista está disponible para el tipo de usuario anfitrión, y permite modificar los precios de sus alojamientos.



The screenshot shows a web interface for a user named 'Bienvenido'. At the top, there are navigation links: 'Alojamientos Disponibles', 'Reservar alojamiento', and 'Registrar nuevos precios' (which is highlighted). Below the navigation bar, the title 'Sus alojamientos' is displayed. The main content is a table listing seven properties. Each row includes a placeholder image for the property, a 'Modificar precios' button, the property name, location, address, and a list of prices for different durations.

Foto		Nombre	Localidad	Ubicación precisa	Precios
	Modificar precios	Casa Pedro	Boecillo	Calle cositas, 3, 1ª	Noche: 70€ F.Semana: 200€ Semana: 500€ Mes: 3000€
	Modificar precios	Torcuato Menendez	Dueñas	Calle cositas, 3, 1ª	Noche: 70€ F.Semana: 200€ Semana: 500€ Mes: 3000€
	Modificar precios	Puludos House	Laguna de Duero	Calle cositas, 3, 1ª	Noche: 70€ F.Semana: 200€ Semana: 500€ Mes: 3000€
	Modificar precios	Xavo del 8 Land	Ávila	Calle cositas, 3, 1ª	Noche: 70€ F.Semana: 200€ Semana: 500€ Mes: 3000€
	Modificar precios	Charleston Hotel	Laguna de Duero	Calle cositas, 3, 1ª	Noche: 70€ F.Semana: 200€ Semana: 500€ Mes: 3000€
	Modificar precios	Yoni Macaroni Place	Sanabria	Calle cositas, 3, 1ª	Noche: 70€ F.Semana: 200€ Semana: 500€ Mes: 3000€
	Modificar precios	Toripetos Home	Laguna de Duero	Calle cositas, 3, 1ª	Noche: 70€ F.Semana: 200€ Semana: 500€ Mes: 3000€

Figura 21: Registrar precio

6.10. Modificar precios

Como se puede ver en la vista anterior, permite modificar los precios para cada alojamiento a través de un botón. Este botón nos lleva a otra vista donde poder modificar cada uno de los precios asociados a el alojamiento y tiempo de la estancia.

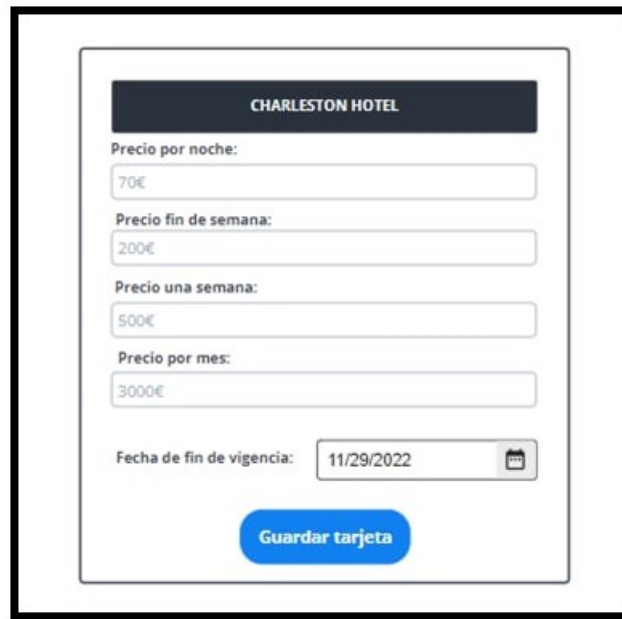
El formulario, titulado 'CHARLESTON HOTEL', permite modificar los precios por diferentes tipos de estancia. Incluye campos de entrada para: 'Precio por noche' (70€), 'Precio fin de semana' (200€), 'Precio una semana' (500€) y 'Precio por mes' (3000€). También hay un campo para la 'Fecha de fin de vigencia' (11/29/2022) con un icono de calendario. Un botón azul 'Guardar tarjeta' está situado al final.

Figura 22: Modificar precio

7. Diseño base de datos

Puesto que hemos decidido utilizar una base de datos para la persistencia y almacenamiento de los datos, ahora vamos a hacer un pequeño desglose de los diagramas o modelos realizados para la construcción de la base de datos.

7.1. Modelo de dominio reducido

Antes de comenzar con el propio diseño y realización de la base de datos optamos por reducir la complejidad del modelo de dominio ofrecido por los profesores. De esta forma conservábamos únicamente las clases y las enumeraciones que nos iban a ser útiles para la implementación final.

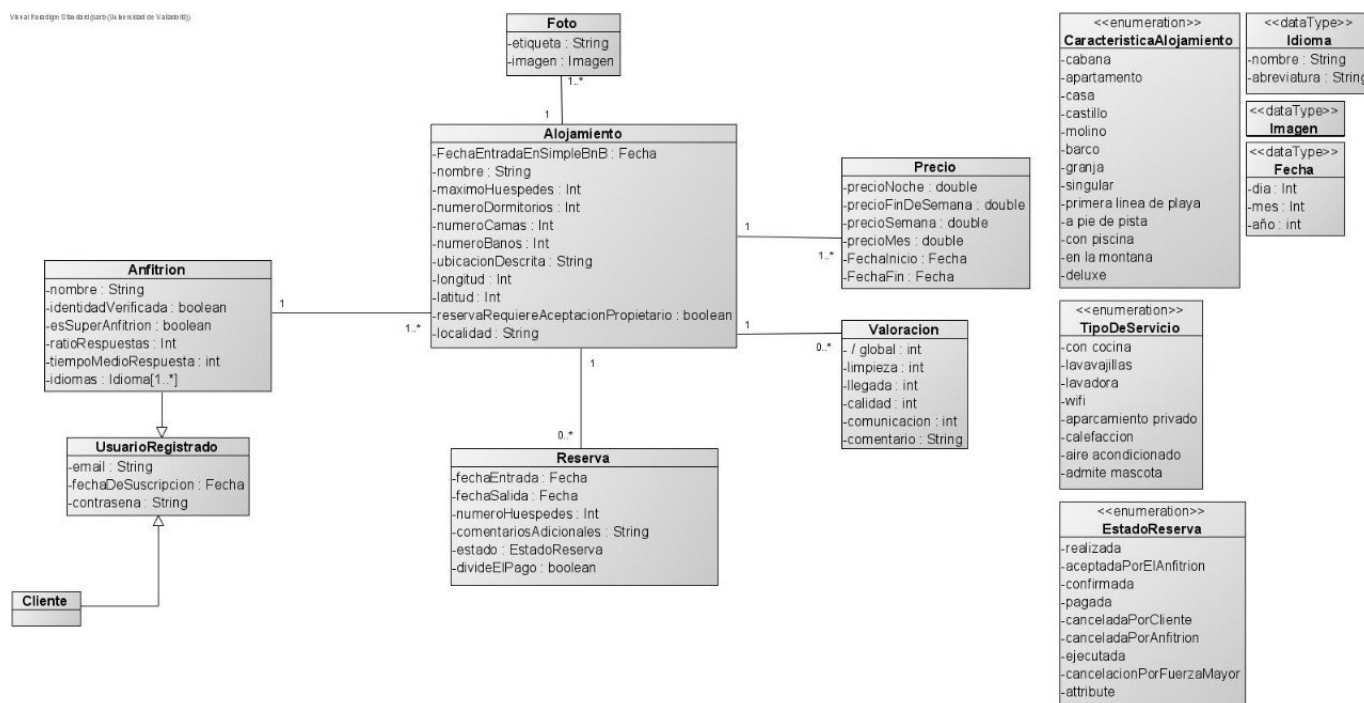


Figura 23: Modelo de dominio reducido

7.2. Diagrama entidad relación

Una vez ya hemos reducido la complejidad del modelo de dominio, hemos creado un diagrama entidad relación. Donde podemos observar las claves foráneas y primarias de cada una de las entidades, así como las relaciones entre ellas. De esta forma, ya podríamos elaborar el diagrama relacional y llevar a cabo la creación de la base de datos en un script SQL.

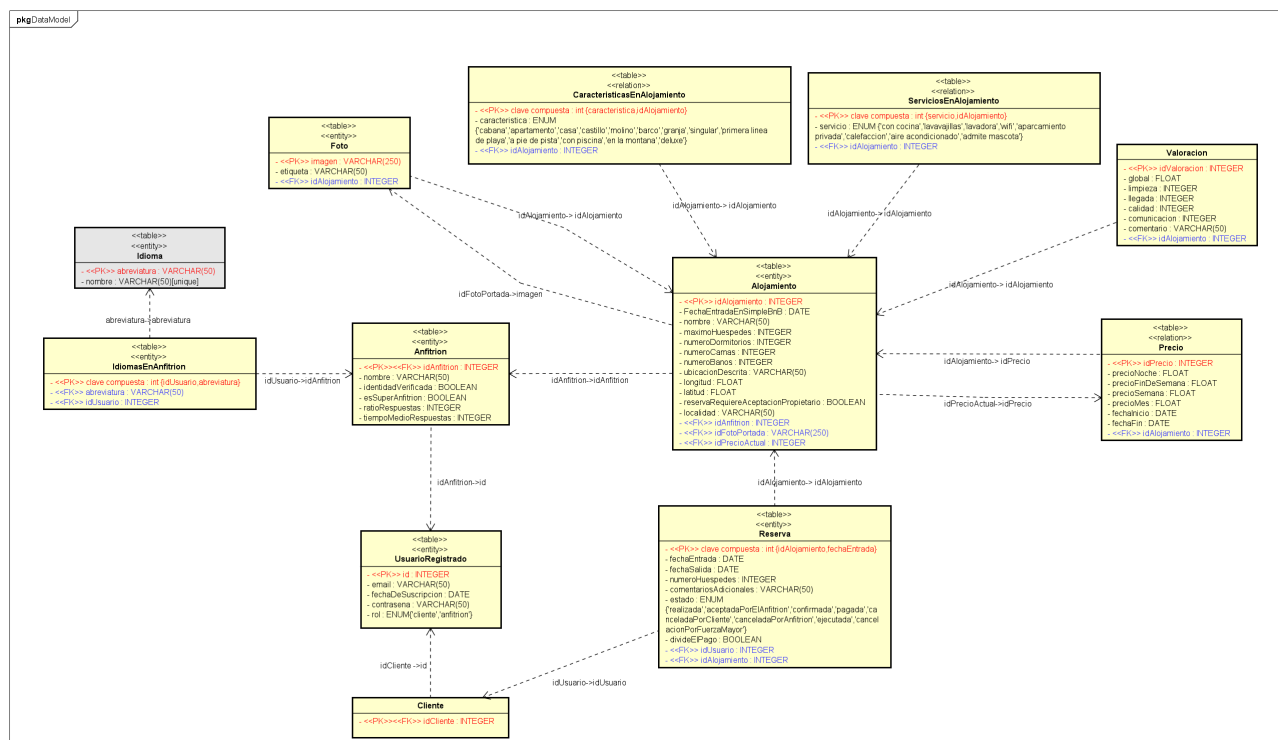


Figura 24: Diagrama E-R

7.3. Diagrama relacional

Hemos realizado también un diagrama relacional donde se muestran las tablas junto a sus atributos o campos, sus claves primarias foráneas y las relaciones de forma más visual. Esto facilita la creación del script.

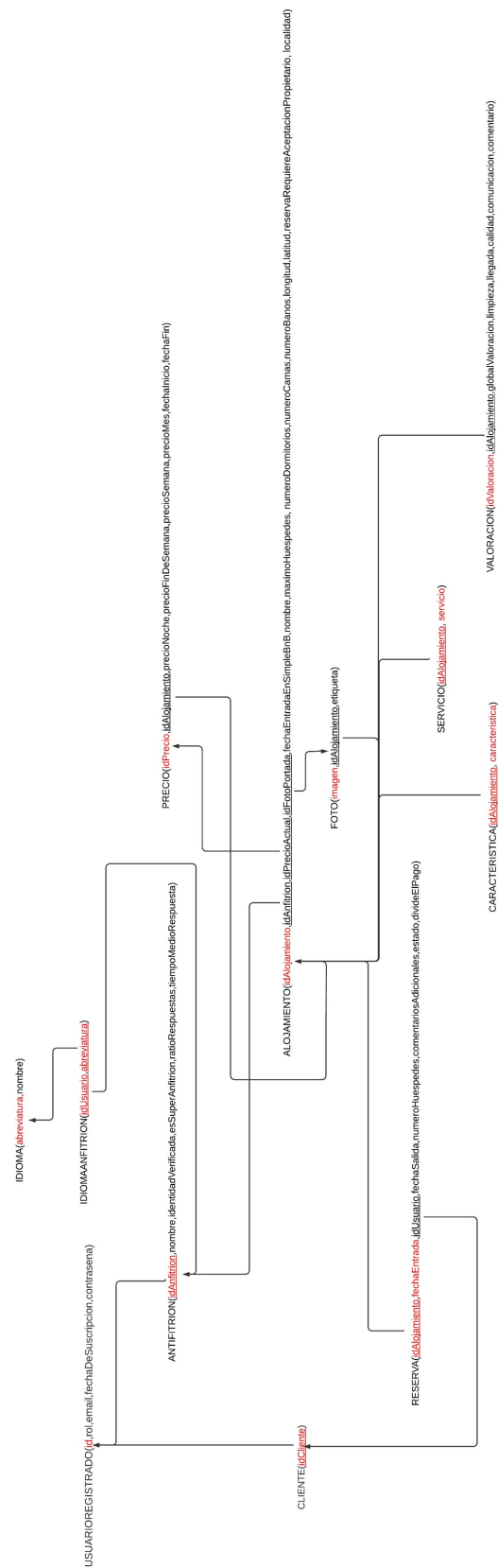


Figura 25: Diagrama E-R

7.4. Revisión final y script

Con los diagramas anteriormente documentados, hemos elaborado un script sql, con la creación de las tablas campos, atributos y relaciones. También la correspondiente población de la base de datos, con el fin de cubrir el mayor número de casos de prueba y probar el funcionamiento de la aplicación, que posteriormente llevaremos a cabo en el apartado “Test y pruebas de la implementación”.

En el siguiente enlace dejamos donde se encuentra el script dentro del proyecto, así como el contenido de dicho script:

https://gitlab.inf.uva.es/hectori/pdsc_proyecto_g04/-/blob/Entrega/app/VacationAsHome.sql

8. Implementación

La distribución de los paquetes y clases que hemos utilizado es la siguiente:

- webapp
 - Anfitrion_Header.jsp
 - Cliente_Header.jsp
 - ConfirmarReserva.jsp
 - CtrlVistaDisponibles.jsp
 - CtrlVistaLogin.jsp
 - CtrlVistaPrecios.jsp
 - CtrlVistaReservas.jsp
 - DisponiblesView.jsp
 - Logged.jsp
 - ReservasView.jsp
 - Usuario_Header.jsp
 - disponible_NO_login.html
 - disponibles.jsp
 - info_alojamiento.jsp
 - inicio_1.html
 - localidades_spain.js
 - registrar.jsp
 - reservas.jsp
 - style.css
- java
 - Control
 - ActualizaPreciosServlet.java

- AnfitrionServlet.java
- ConfirmarReservaServlet.java
- DisponiblesServlet.java
- InformacionServlet.java
- RegistroServlet.java
- ReservasServlet.java
- Persistencia
 - DAO
 - ◊ AlojamientosDAO.java
 - ◊ UsuarioRegistradoDAO.java
 - DataAccess
 - ◊ ConnectionPool.java
- Utils
 - Alojamiento.java
 - Precio.java
 - Reserva.java
 - UsuarioRegistrado.java

9. Test y pruebas de la implementación

Con el fin de comprobar que la implementación llevada a cabo cumple debidamente con el funcionamiento especificado en la descripción de cada caso de uso. Hemos elaborado una batería de pruebas que cubre la totalidad de los casos planteados en cada caso de uso.

Para la realización de las pruebas hemos utilizado el script SQL, que se encuentra en el repositorio de GitLab que hemos utilizado para la realización del proyecto, que se encuentra en este enlace https://gitlab.inf.uva.es/hectori/pdsc_proyecto_g04/-/blob/main/VacationAsHome.sql

9.1. Caso de uso “IDENTIFICARSE”

Condiciones entrada	Salida esperada	Salida obtenida
Email= "" Contraseña= ""	Muestra una alerta que indica que los campos no pueden ser vacíos	Muestra una alerta que indica que los campos no pueden ser vacíos
Email= manolo@gmail.com Contraseña= abcdefg	Mensaje que indica que la contraseña no corresponde	Mensaje que indica que la contraseña no corresponde
Email= 124@123.com Contraseña= abcdefg	Mensaje que indica que el email no existe en la BD	Mensaje que indica que el email no existe en la BD
Email= 124@123.com Contraseña= 1234	Mensaje que indica que el email no existe en la BD	Mensaje que indica que el email no existe en la BD
Email= manolo@gmail.com Contraseña= 1234	Cambio de vista correspondiente al usuario logueado	Cambio de vista correspondiente al usuario logueado (5)

Cuadro 1: IDENTIFICARSE

9.2. Caso de uso “CONSULTAR ALOJAMIENTOS”

En este caso de uso existen 2 comprobaciones previas que se realizan con código JavaScript del controlador de la vista en el HTML.

- Comprobar que la fecha de entrada siempre es anterior a la fecha de salida:

Fecha de Entrada: 21/12/2022

Fecha de Salida: 22/12/2022

Localidad: pruebaerror

Buscar

Figura 26: Comprobación 1 “CONSULTAR ALOJAMIENTOS”

- Comprobar que la duración máxima de la estancia o alquiler no es superior a 30 días:

Fecha de Entrada: 19/12/2022

Fecha de Salida: 22/12/2022

Localidad: pruebaerror

Buscar

Figura 27: Comprobación 2 “CONSULTAR ALOJAMIENTOS”

Condiciones entrada	Salida esperada	Salida obtenida
Fecha entrada = 15/12/2022 Fecha salida = 20/12/2022 Localidad= oiuyhnj	Alerta o mensaje de error indicando que la localidad no existe	Alerta o mensaje de error indicando que la localidad no existe
Fecha entrada = 15/12/2022 Fecha salida = 20/12/2022 Localidad= Portugalete	Mensaje de error o aviso en pantalla indicando que no hay alojamientos disponibles, ya que no existen inserts para esa localidad	Mensaje de error o aviso en pantalla indicando que no hay alojamientos disponibles ya que no existen inserts para esa localidad
Fecha entrada = 20/12/2022 Fecha salida = 25/12/2022 Localidad= Aldeamayor de San Martín	Mensaje de error o aviso en pantalla, ya que no hay disponibles debido a que están reservados	Mensaje de error o aviso en pantalla, ya que no hay disponibles debido a que están reservados
Fecha entrada = 02/01/2023 Fecha salida = 07/01/2023 Localidad= Laguna de Duero	Muestra el alojamiento disponible, para esa localidad y fechas	Muestra el alojamiento disponible, para esa localidad y fechas
Fecha entrada = 25/12/2022 Fecha salida = 23/12/2022 Localidad= Laguna de Duero	Muestra una alerta en la casilla de fecha de salida indicando que esta debe de ser posterior a la fecha de entrada.	Muestra una alerta en la casilla de fecha de salida indicando que esta debe de ser posterior a la fecha de entrada.

Cuadro 2: CONSULTAR ALOJAMIENTOS

9.3. Caso de uso “REGISTRAR NUEVOS PRECIOS”

De nuevo en este caso de uso existen 2 comprobaciones previas que se realizan con código JavaScript del controlador de la vista en el HTML.

- Comprobar que la fecha es correcta, ya que se introduce con un input tipo date, que permite seleccionar la fecha a través de un calendario, por tanto, el formato siempre va a ser correcto.
- Comprobar que, en la selección de fechas, no permita fechas anteriores a la actual.

Figura 28: Comprobaciones “REGISTRAR NUEVOS PRECIOS”

Condiciones entrada	Salida esperada	Salida obtenida
Precio por noche= -5 Precio fin de semana= 150 Precio una semana= 350 Precio por un mes= 1050 Fecha fin de vigencia= 28/12/2022	Mensaje de error indicando que los valores de los nuevos precios han de ser positivos	Mensaje de error indicando que los valores de los nuevos precios han de ser positivos
Precio por noche= 150 Precio fin de semana= 150 Precio una semana= 350 Precio por un mes= 1050 Fecha fin de vigencia= 28/12/2022	Mensaje de error indicando que los valores de los nuevos precios han de ser incrementales, precio por noche más barato que finde se semana etc	Mensaje de error indicando que los valores de los nuevos precios han de ser incrementales, precio por noche más barato que finde se semana etc
Precio por noche= 5 Precio fin de semana= 360 Precio una semana= 350 Precio por un mes= 1050 Fecha fin de vigencia= 28/12/2022	Mensaje de error indicando que los valores de los nuevos precios han de ser incrementales, precio por noche más barato que finde se semana etc	Mensaje de error indicando que los valores de los nuevos precios han de ser incrementales, precio por noche más barato que finde se semana etc
Fecha entrada = Precio por noche= 0 Precio fin de semana= 0 Precio una semana= 0 Precio por un mes= 0 Fecha fin de vigencia= 29/12/2022	Mensaje de error indicando que los valores de los nuevos precios han de ser incrementales, precio por noche más barato que finde se semana etc	Mensaje de error indicando que los valores de los nuevos precios han de ser incrementales, precio por noche más barato que finde se semana etc
Precio por noche= 75 Precio fin de semana= 150 Precio una semana= 350 Precio por un mes= 1050 Fecha fin de vigencia= 29/12/2022	Actualiza el valor de la tarjeta de precios y muestra un mensaje notificando de ello al usuario	Actualiza el valor de la tarjeta de precios y muestra un mensaje notificando de ello al usuario

Cuadro 3: REGISTRAR NUEVOS PRECIOS

Gracias a las pruebas de la parte inferior, conseguimos detectar tres errores de implementación relacionados con la entrada introducida por el usuario, que no habían sido tratados y gracias a las pruebas pudimos corregirlo y mejorar el funcionamiento del sistema.

Cuando el usuario introducía en alguno de los campos de los nuevos precios un espacio, dejaba el campo vacío o introducía letras (caracteres no numéricos), no mostraba ningún mensaje de error al usuario, y no actualizaba con los precios introducidos.

Condiciones entrada	Salida esperada	Salida obtenida
Precio por noche= " " Precio fin de semana= 150 Precio una semana= 350 Precio por un mes= 1050 Fecha fin de vigencia= 29/12/2022	Muestra un mensaje de error indicando que ninguno de los campos de los nuevos precios ha de ser vacío	Muestra un mensaje de error indicando que ninguno de los campos de los nuevos precios ha de ser vacío
Precio por noche= "" Precio fin de semana= 150 Precio una semana= 350 Precio por un mes= 1050 Fecha fin de vigencia= 29/12/2022	Muestra un mensaje de error indicando que ninguno de los campos de los nuevos precios ha de ser vacío	Muestra un mensaje de error indicando que ninguno de los campos de los nuevos precios ha de ser vacío
Precio por noche= hola Precio fin de semana= 150 Precio una semana= 350 Precio por un mes= 1050 Fecha fin de vigencia= 29/12/2022	Muestra alerta indicando que el precio nuevo ha de ser un número, no letras	Muestra alerta indicando que el precio nuevo ha de ser un número, no letras

Cuadro 4: REGISTRAR NUEVOS PRECIOS 2

9.4. Caso de uso “REALIZAR RESERVA”

Para la realización de este caso de uso, primero se realiza una búsqueda de los alojamientos disponibles en la localidad y para el número de huéspedes indicado, por ello primero se ejecuta esta búsqueda y después se comenzaría a efectuar la reserva en sí.

Para ello en las pruebas vamos a distinguir “dos partes” una primera que se refiere a la búsqueda de los alojamientos disponibles/candidatos para la reserva similar a “consultar alojamientos” pero con otros campos de entrada. Y posteriormente pruebas relacionadas con la realización de la reserva en concreto, una vez ya elegido el alojamiento que el usuario quiere reservar. Además dentro de las pruebas las dividimos según el tipo de usuario, ya que únicamente pueden realizar una reserva los clientes, los anfitriones no.

Para esta primera parte, que constituye la búsqueda de los alojamientos disponibles, el funcionamiento es el mismo independientemente del tipo de usuario, es decir, sea cliente o anfitrión la salida espera y obtenida es la misma.

■ Primera parte del Caso de Uso

Condiciones entrada	Salida esperada	Salida obtenida
Tipo usuario= cliente o anfitrión Localidad= Aldeamayor de San Martín Huéspedes= 3	Muestra los alojamientos disponibles para esa entrada.	Muestra los alojamientos disponibles para esa entrada.
Tipo usuario= cliente o anfitrión Localidad= Aldeamayor de San Martín Huéspedes= -1	Muestra una alerta indicando que el valor debe ser superior o igual a 1	Muestra una alerta indicando que el valor debe ser superior o igual a 1
Tipo usuario= cliente o anfitrión Localidad= Aldeamayor de San Martín Huéspedes= 21	Muestra una alerta indicando que el valor debe ser inferior o igual a 20	Muestra una alerta indicando que el valor debe ser inferior o igual a 20
Tipo usuario= cliente o anfitrión Localidad= Puebla de Sanabria Huéspedes= 3	Mensaje indicando que no existen alojamientos disponibles para ese municipio	Mensaje indicando que no existen alojamientos disponibles para ese municipio
Tipo usuario= cliente o anfitrión Localidad= ghjklkj Huéspedes= 3	Muestra una alerta indicando que introduzca una localidad válida	Muestra una alerta indicando que introduzca una localidad válida

Cuadro 5: REALIZAR UNA RESERVA PARTE 1

■ Segunda parte del Caso de Uso

Condiciones entrada	Salida esperada	Salida obtenida
Tipo usuario= cliente Fecha de Entrada= 21/12/2022 Fecha de Salida= 25/12/2022 Mensaje para anfitrión=”Buenas” Fraccionar pago= true	Muestra un mensaje indicando que el alojamiento está reservado en las fechas indicadas	Muestra un mensaje indicando que el alojamiento está reservado en las fechas indicadas

Cuadro 6: REALIZAR RESERVA PARTE 2.1

La siguiente prueba nos ayudó de nuevo a detectar un fallo en la implementación, que no era acorde a lo que solicitaba el caso de uso del enunciado de la práctica. Ya que nosotros permitíamos

la creación de una reserva con el campo “Mensaje para anfitrión” vacío. Y se solicita que este no sea vacío, por tanto, modificamos el código para que cumpliera con el requerimiento de la secuencia del caso de uso.

Condiciones entrada	Salida esperada	Salida obtenida
Tipo usuario= cliente Fecha de Entrada= 27/12/2022 Fecha de Salida= 28/12/2022 Mensaje para anfitrión= “” Fraccionar pago= false	Muestra un mensaje indicando que el mensaje para el anfitrión no puede ser vacío	Muestra un mensaje indicando que el mensaje para el anfitrión no puede ser vacío
Tipo usuario= cliente Fecha de Entrada= 05/01/2023 Fecha de Salida= 04/01/2023 Mensaje para anfitrión= “Reserva” Fraccionar pago= true	Muestra un mensaje indicando que la fecha de entrada no puede ser posterior a la de salida	Muestra un mensaje indicando que la fecha de entrada no puede ser posterior a la de salida
Tipo usuario= cliente Fecha de Entrada= 02/01/2023 Fecha de Salida= 04/01/2023 Mensaje para anfitrión= “Reserva” Fraccionar pago= true	Crea la reserva para ese usuario, ese alojamiento y los parámetros de entrada establecidos	Crea la reserva para ese usuario, ese alojamiento y los parámetros de entrada establecidos
Tipo usuario= anfitrión Fecha de Entrada= 05/01/2023 Fecha de Salida= 07/01/2023 Mensaje para anfitrión= “Reserva” Fraccionar pago= true	Muestra un mensaje indicando que el anfitrión no puede efectuar una reserva	Muestra un mensaje indicando que el anfitrión no puede efectuar una reserva

Cuadro 7: REALIZAR UNA RESERVA PARTE 2.2

10. Herramientas y bibliografía

Para la realización del proyecto hemos utilizado distintas herramientas y apuntes, los cuales son:

- Apuntes de la asignatura. Campus Virtual UVa
- JDK 1.8
- NetBeans 12.1. Desarrollo de código e implementación.
- Visual Paradigm 16.2. Modelos de dominio.
- Astah Professional. Diagramas de diseño.
- MySQL. Creación base de datos
- GitLab. Constancia del tiempo invertido y seguimiento
- Gantt Proyect. Realización del diagrama de Gantt
- Moqups. Para la realización de los bocetos de las vistas
- Draw.io. Realización de diagramas de planificación (PFD, PBS, WBS...)