Lorena Jiménez Tejada

Víctor Manuel Faría Chávez

2º CFGS Informática y Comunicaciones - Desarrollo de Aplicaciones Web

Curso académico 2020/21

Proyecto de Desarrollo de Aplicaciones Web

Resumen

En los establecimientos hoteleros, a día de hoy, el control de limpieza de las habitaciones entre el check-out y el siguiente check-in es difícil de organizar.

Nuestro objetivo principal es crear una aplicación web responsive que agilice la comunicación entre camareros de piso y recepción, consiguiendo así que la información del estado de las habitaciones llegue de manera inmediata. Para ello las camareras de piso dispondrán de un listado con las habitaciones que deben limpiar ese día y al chequearlas como limpias, la información se actualizará en recepción.

De esta forma, conseguiremos facilitar la labor de recepción a la hora de indicar a los clientes cuándo pueden ocupar su habitación.

Abstract

In hotel establishments nowadays, controlling room service between check-out and the next check-in is difficult to organize.

Our main objective is to create a responsive web application that streamlines communication between chambermaids and reception, thus ensuring that the information on the status of the rooms arrives immediately. For this, the chambermaids will have a list of the rooms that must be cleaned that day. Once they're checked off as cleaned, the information will be updated at the reception.

This way we will be able to facilitate reception work when it comes to telling customers when they can occupy their room.

1. Índice

1. Índice **2**

2. Índice de figuras **5**

3. Introducción **6**

4. Análisis del problema **6**

5. Definición del proyecto **7**

5.1. Requisitos que debe cumplir nuestra aplicación **7**

5.1.1. Login 7

5.1.2. Añadir, modificar y eliminar usuario 7

5.1.3. Una pantalla de inicio para la recepción y el mánager 7

5.1.4. Rápida visualización del estado de las habitaciones 8

5.1.5. Ordenar por diferentes opciones 8

5.1.6. Añadir nuevas reservas y eliminar las anuladas 8

5.1.7. Alterar el estado de las habitaciones 8

5.1.8. Pantalla para Jefa de camareras 9

5.1.9. Pantalla para las camareras de piso 9

5.1.10. Crear habitaciones nuevas, editarlas y eliminarlas 9

5.2. Objetivos y alcance **9**

6. Diseño **10**

6.1. Propuestas de posibles interfaces **10**

6.1.1. Pantalla de login 10

6.1.2. Pantalla de inicio de recepción 11

6.1.3. Pantalla de reservas de recepción 12

6.1.4. Pantalla añadir reservas 13

6.1.5. Pantalla de inicio del mánager 14

6.1.6. Pantallas de usuarios de mánager 15

6.1.7. Pantallas de habitaciones de mánager 16
6.1.8. Pantalla de la jefa de camareras de piso 17
6.1.9. Pantalla de camarera de piso 18
6.1.10. Pantallas móviles de las empleadas de limpieza 19
6.2. Casos de uso 20
7. Planificación 25
7.1. Recursos 25
7.1.1. Software necesario para ejecutar en local 25
7.1.2. Herramientas 25
7.1.2.1. Visual Studio Code 25
7.1.2.2. Spring Tool Suite 25
7.1.2.3. Balsamiq Cloud 25
7.1.2.4. Maven 25
7.1.2.5. Docker 26
7.1.3. Tecnologías 26
7.1.3.1. Java 26
7.1.3.2. Spring boot 26
7.1.3.3. Vue.js 27
7.1.4. Servicios 27
7.1.5. Humanos 27
7.1.6. Temporal 27
7.2. Tareas a llevar a cabo 28
7.3. Diagrama de flujo 29
7.4. Estructura de la base de datos 30
7.5. Cronograma 31
7.6. Metodología 32
7.7. Estimación de costes 33

7.8. Validación 33

8. Ejecución 33

9. Conclusiones 34

9.1. ¿Cumple con lo previsto? 34

9.2. Propuestas de mejora 34

11. Bibliografía 35

12. Anexos 35

2. Índice de figuras

Figura 1: Pantalla de login **10**

Figura 2: Pantalla de inicio de recepción **11**

Figura 3: Pantalla de reservas de recepción **12**

Figura 4: Pantalla nueva reserva de recepción **13**

Figura 5: Pantalla de inicio del mánager **14**

Figura 6: Pantalla de listado de usuarios de mánager **15**

Figura 7: Pantalla de usuario en detalle de mánager **15**

Figura 8: Pantalla de listado de habitaciones de mánager **16**

Figura 9: Pantalla de organización de limpieza de la jefa de camareras de piso **17**

Figura 10: Pantalla de listado de la camarera de piso **18**

Figura 11: Pantalla móvil de la jefa de camareras de piso **19**

Figura 12: Pantalla móvil de la camarera de piso **19**

Figura 13: Casos de uso 'Usuario no identificado' **20**

Figura 14: Casos de uso 'Camarera de piso' **20**

Figura 15: Casos de uso 'Jefa de camareras de piso' **20**

Figura 16: Casos de uso 'Recepción' **21**

Figura 17: Casos de uso 'Manager' **22**

Figura 18: Casos de uso 'Manager con rol de recepción' **23**

Figura 19: Casos de uso 'Manager con funciones de Jefa de camareras' **24**

Figura 20: Diagrama de una petición **29**

Figura 21: Estructura de la base de datos **30**

3. Introducción

En los establecimientos hoteleros, a día de hoy, el control de limpieza de las habitaciones entre el check-out y el siguiente check-in es difícil de organizar, de modo que hemos creado una aplicación web responsive que se encargará de manejar este aspecto de la vida diaria del hotel y así mejorar su rendimiento.

4. Análisis del problema

En la actualidad para que en la recepción de un hotel sepan si una habitación está lista para que un cliente nuevo pueda entrar, cuando el anterior salió ese mismo día, lo habitual es llamar por teléfono, o walkie-talkie, a la jefa de las camareras de piso. Ésta pasará por la habitación para comprobarlo y llamará a recepción con la respuesta.

También sucede que las camareras de piso no suelen tener seguridad de si una habitación que deben limpiar, ya sea por un check-out, o por la limpieza diaria a un cliente que permanece, se encuentra en ese momento libre, debiendo llamar a la puerta, lo cual puede molestar al cliente si está descansando, o llamando a recepción para comprobar si la llave de la habitación está allí, y por tanto el cliente está fuera.

No es algo muy eficiente y pensamos que podemos ayudar a mejorarlo.

Sabemos de la existencia de este problema debido a la experiencia en el sector, hace unos años, de uno de los componentes del grupo, además de haber contactado con diferentes camareras de piso que están trabajando en la actualidad en diferentes hoteles. Comprobando que la situación no ha cambiado a lo largo de los años.

5. Definición del proyecto

Para solucionar el problema que hemos detectado, hemos pensado crear una aplicación web que gestionará el estado de limpieza y ocupación real de las habitaciones del hotel.

5.1. Requisitos que debe cumplir nuestra aplicación

5.1.1. Login

El usuario podrá loguearse y salir de la sesión de forma fácil.

5.1.2. Añadir, modificar y eliminar usuario

A este apartado sólo podrá accederse siendo administrador.

Deberá haber una página encargada de dar de alta a los usuarios con sus respectivos roles.

Se mostrará también un listado con los usuarios actuales y, al seleccionar uno, mostrar la información más detallada y permitir la modificación y eliminación de los mismos.

5.1.3. Una pantalla de inicio para la recepción y el mánager

Esta pantalla debe mostrar una gráfica con el porcentaje de ocupación del hotel en el día, otra con el porcentaje de las habitaciones limpias y una tercera con la ocupación a lo largo del mes.

La única diferencia en esta pantalla será el menú lateral que variará según el rol.

5.1.4. Rápida visualización del estado de las habitaciones

A primera vista la recepcionista debe poder visualizar qué habitaciones faltan por limpiar y si está ocupada o no dicha habitación en el momento. Esta vista debe actualizarse cada vez que cambie el estado de una habitación.

Desde recepción marcarán la habitación como no ocupada a la entrega de las llaves del cliente al salir del hotel, de este modo la camarera de piso sabrá que puede pasar a realizar la limpieza sin tener que llamar a la puerta, con la consiguiente molestia al cliente si se encuentra descansando.

5.1.5. Ordenar por diferentes opciones

Debe poderse ordenar por tipo de habitación, estado de limpieza, ocupada, fecha de entrada y fecha de salida.

5.1.6. Añadir nuevas reservas y eliminar las anuladas

Deben poderse añadir nuevas reservas con su número de habitación, fecha de check-in y fecha de check-out, y poder eliminarse en caso de ser anuladas dichas reservas.

5.1.7. Alterar el estado de las habitaciones

Al acabar la limpieza, la camarera de piso marcará la habitación como limpia, para que la recepción sea consciente del cambio de estado.

Desde recepción debe poderse cambiar el estado de una habitación en sus campos ocupados, urgente y check-out.

5.1.8. Pantalla para Jefa de camareras

Debe haber una pantalla para que la jefa de camareras de piso pueda asignar a cada empleada las habitaciones que debe limpiar, indicando el tipo de limpieza (check-out o permanencia en el hotel) y si la limpieza es urgente porque el nuevo cliente ya llegó y está esperando.

La información será actualizada desde la base de datos para adaptarse a la fecha solicitada.

Al finalizar la organización, cada camarera recibirá su listado individualizado.

5.1.9. Pantalla para las camareras de piso

Esta pantalla debe mostrar los números de las habitaciones a limpiar, si es o no un check-out, si el cliente aún se encuentra dentro de la habitación o ya entregó la llave en recepción y si es urgente su limpieza. Además, deberá tener un checkbox para indicar que la habitación ya se encuentra limpia y un botón actualizar estado, para enviar la información a recepción.

5.1.10. Crear habitaciones nuevas, editarlas y eliminarlas

Se podrán crear habitaciones si es la primera vez que se utiliza el programa y si hubieran reformas en el hotel, poder eliminarlas o editarlas o añadir más.

5.2. Objetivos y alcance

Para la demostración tendremos las funcionalidades básicas, pero si al final vemos que la aplicación cumple realmente con las necesidades, tenemos la intención de terminarla en su totalidad. Esto quiere decir que una vez terminemos el ciclo intentaremos seguir programando este proyecto con la idea de acabar vendiéndolo a los hoteles

6. Diseño

6.1. Propuestas de posibles interfaces

6.1.1. Pantalla de login

La primera pantalla que aparecerá ante el usuario será la de login, donde se identificará para acceder a la aplicación. En caso de haber olvidado la contraseña, se mostrará un mensaje para que se ponga en contacto con el mánager y éste se encargará de solucionarlo.

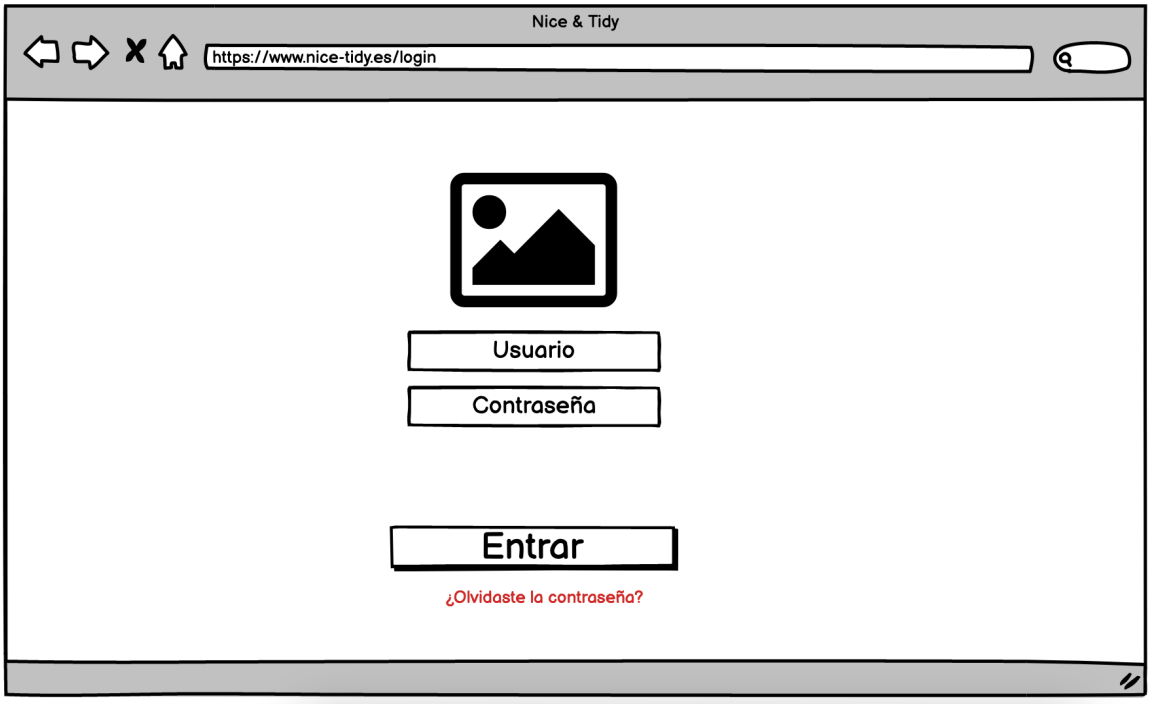

Figura 1: Pantalla de login

6.1.2. Pantalla de inicio de recepción

Si el usuario que accede tiene el rol de recepción, aparecerá una pantalla de inicio que mostrará los gráficos indicados en los requisitos. En el menú lateral tendrá acceso a la pantalla de reservas y podrá cerrar sesión.

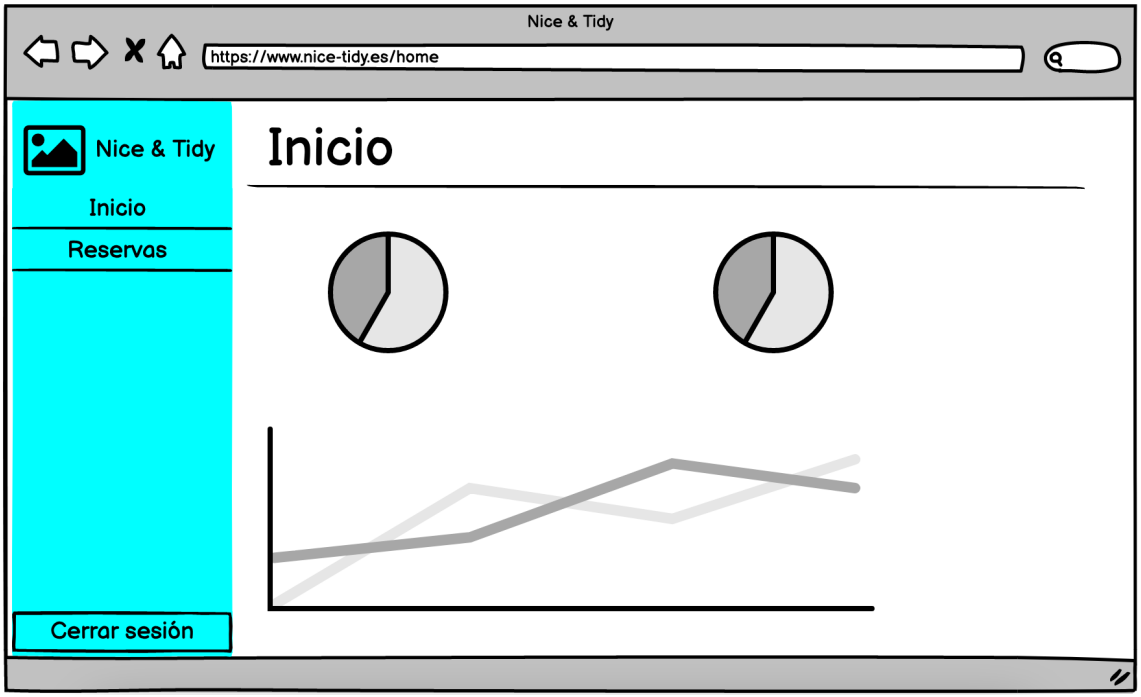

Figura 2: Pantalla de inicio de recepción

6.1.3. Pantalla de reservas de recepción

Cuando accede, aparece la siguiente pantalla, donde se mostrará el listado de las habitaciones con su información correspondiente, además de la opción de ordenarlas por tipo de habitación, estado de limpieza, ocupada o no, fecha de entrada y fecha de salida.

Además tendrá un botón para añadir nuevas reservas y la posibilidad de eliminar las existentes.

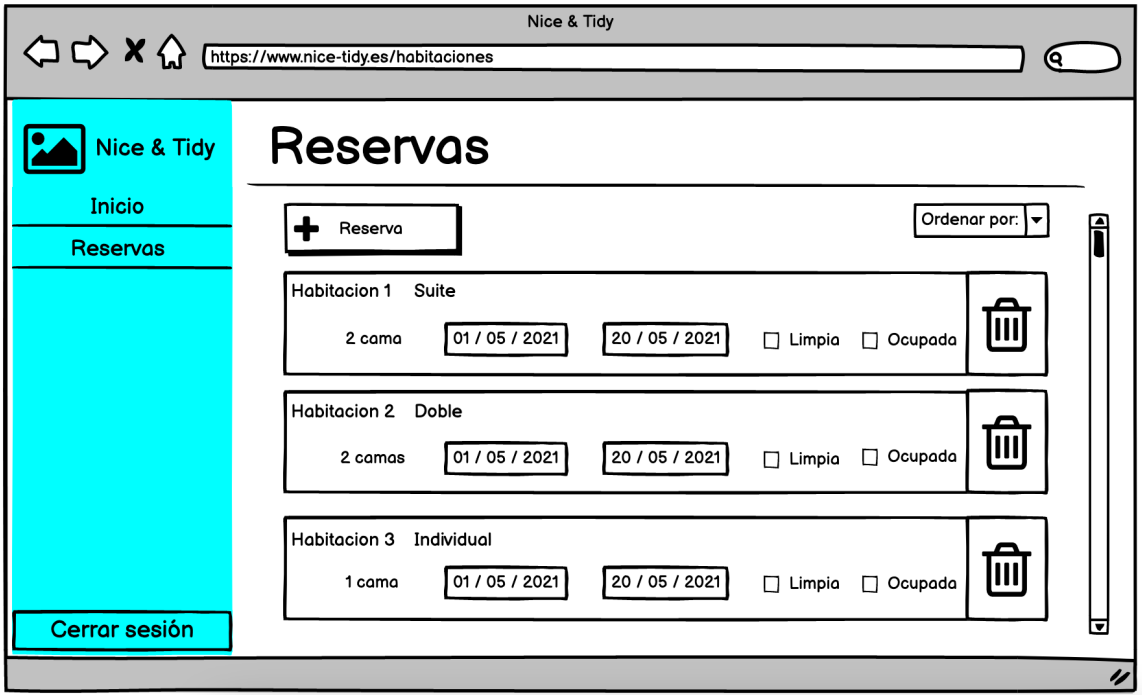

Figura 3: Pantalla de reservas de recepción

6.1.4. Pantalla añadir reservas

Cuando se pulsa el botón de añadir reserva, mostraremos la siguiente pantalla. Si la habitación seleccionada estuviese ocupada en esa fecha, mostraría un mensaje de aviso para que se seleccione otra.

Figura 4: Pantalla nueva reserva de recepción

6.1.5. Pantalla de inicio del mánager

Si el rol de usuario que accede es de mánager, aparecerá la misma pantalla que al recepcionista, pero con más opciones en el menú lateral. Estas opciones son para el control y administración de usuarios y habitaciones.

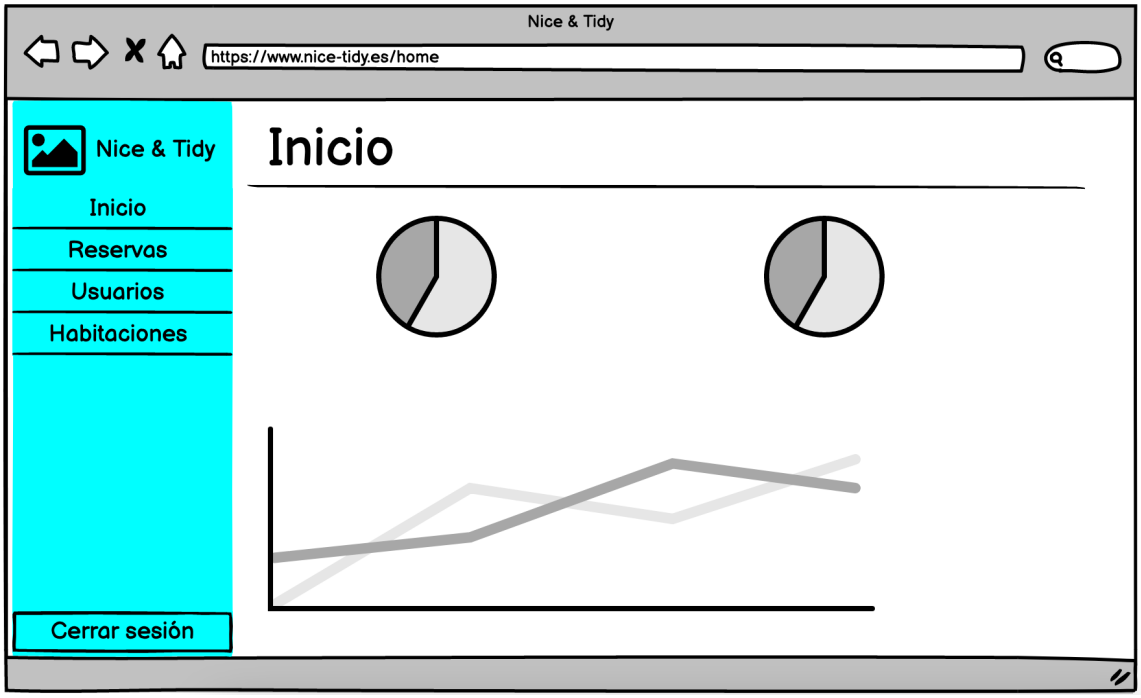

Figura 5: Pantalla de inicio del mánager

6.1.6. Pantallas de usuarios de mánager

Cuando se accede, aparecerá una pantalla con el listado de los mismos, con una opción para ordenar por rol y por nombre y un botón para añadir nuevos usuarios al sistema.

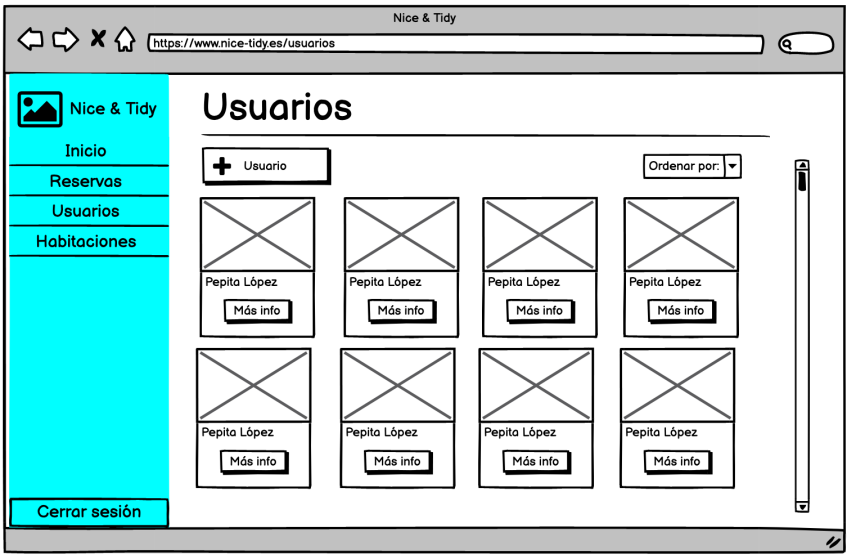

Figura 6: Pantalla listado de usuarios de mánager

Al pulsar en un usuario, se mostrará una nueva ventana con información más detallada, además de las opciones de modificar y eliminar usuario

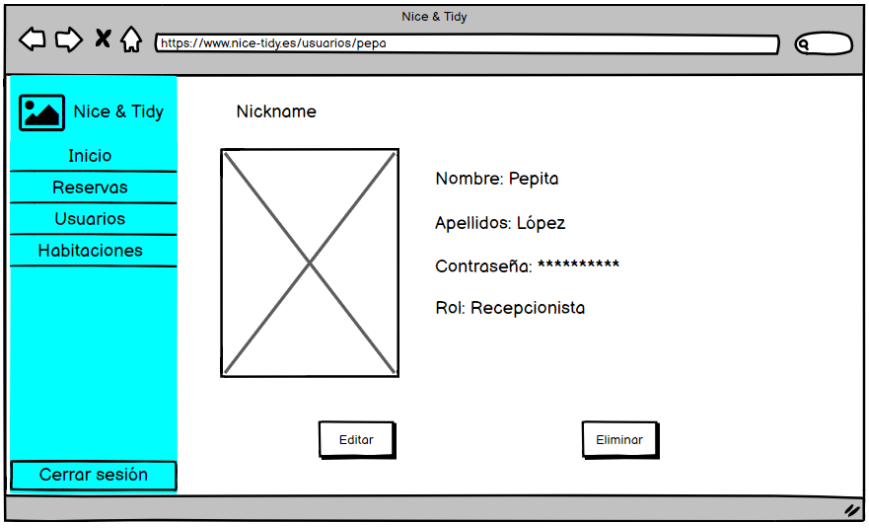

Figura 7: Pantalla de usuario en detalle de mánager

6.1.7. Pantallas de habitaciones de mánager

Cuando se accede por el menú lateral, aparecerá la siguiente pantalla para poder introducir en el sistema las habitaciones, con su número, tipo y número de camas.

Además podrá modificar las habitaciones existentes o eliminarlas si ya no existen en el hotel por motivo de alguna reforma

Figura 8: Pantalla de listado de habitaciones del mánager

6.1.8. Pantalla de la jefa de camareras de piso

Si el usuario que accede a la aplicación tiene el rol de jefa de camareras de piso, tendrá acceso a la siguiente pantalla, donde se cargarán las habitaciones a limpiar ese día, con sus datos correspondientes y sólo tendrá que asignar la camarera de piso que realizará la limpieza. Una vez finalizada la asignación de habitaciones, pulsará el botón enviar, y cada camarera recibirá su listado.

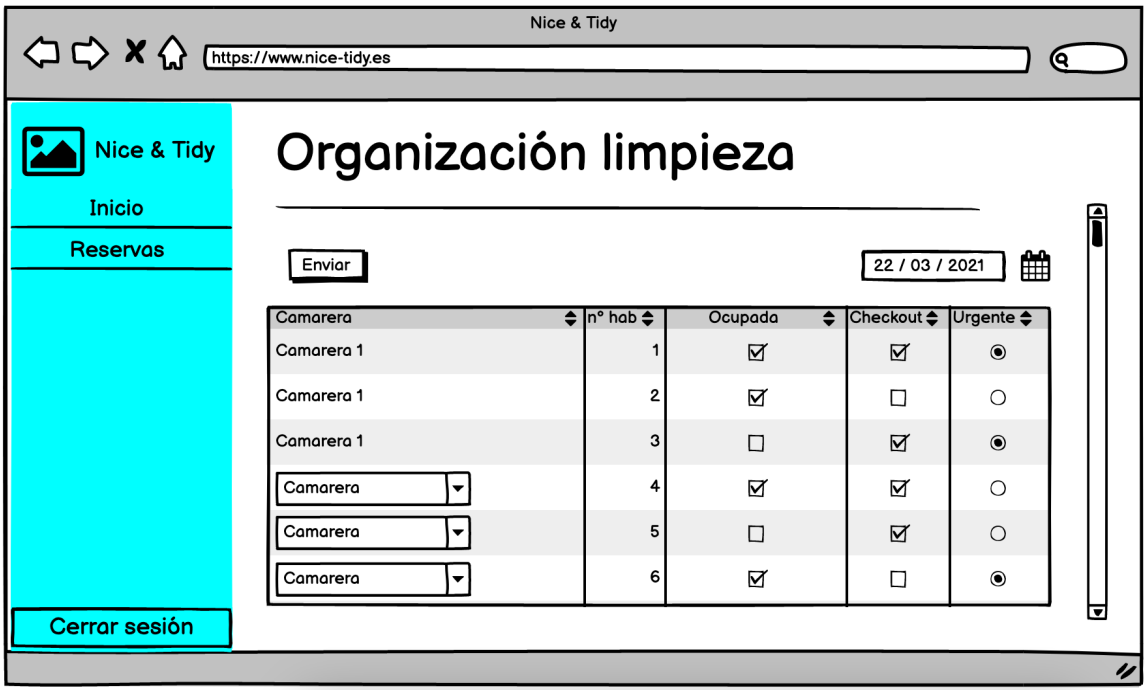

Figura 9: Pantalla de organización de limpieza de la jefa de camareras de piso

6.1.9. Pantalla de camarera de piso

Si el rol del usuario que accede es de camarera de piso, le aparecerá una única pantalla, donde visualizará el listado de habitaciones que debe limpiar ese día y su estado. Cuando haya terminado la limpieza de cada habitación marcará el checkbox y pulsará actualizar para enviar la información a recepción.

Podrá también ordenar las habitaciones por ocupada, urgente y check-out.

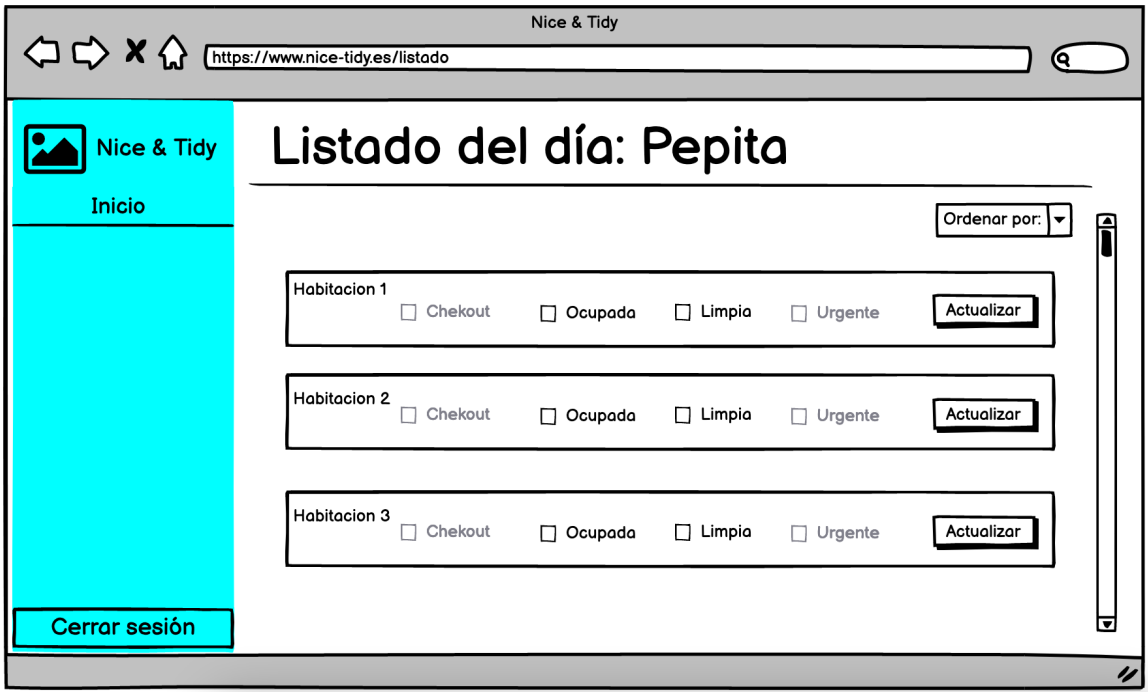

Figura 10: Pantalla de listado de limpieza de la camarera de piso

6.1.10. Pantallas móviles de las empleadas de limpieza

Tanto la pantalla de la jefa de camareras como las de las camareras de piso tendrán su versión móvil como se ve a continuación, con la misma funcionalidad que en la versión de escritorio

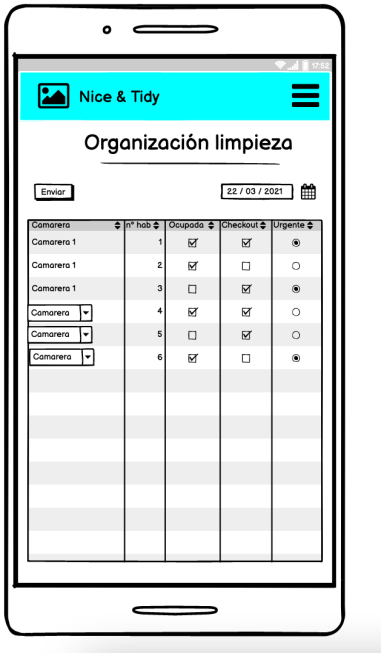

Figura 11: Pantalla móvil de la jefa de camareras de piso

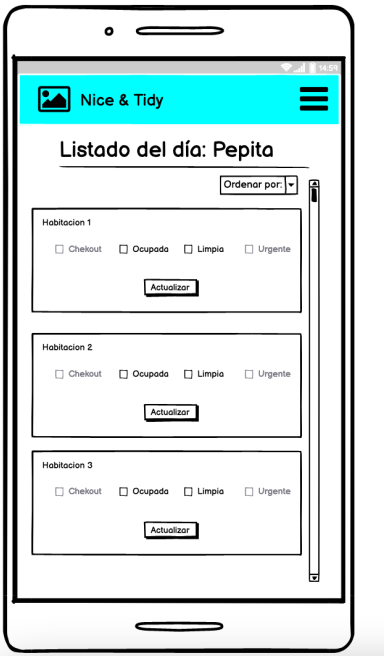

Figura 12: Pantalla móvil de la camarera de piso

6.2. Casos de uso

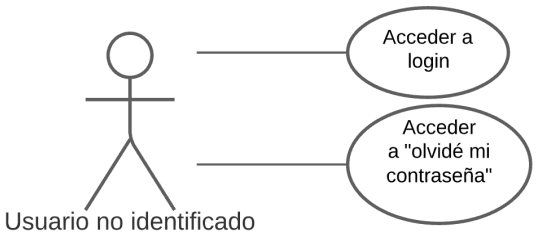

Figura 13: Casos de uso 'Usuario no registrado'

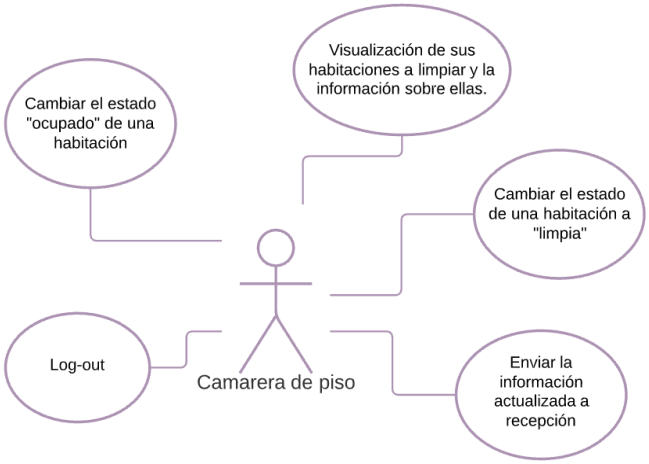

Figura 14: Casos de uso 'Camarera de piso'

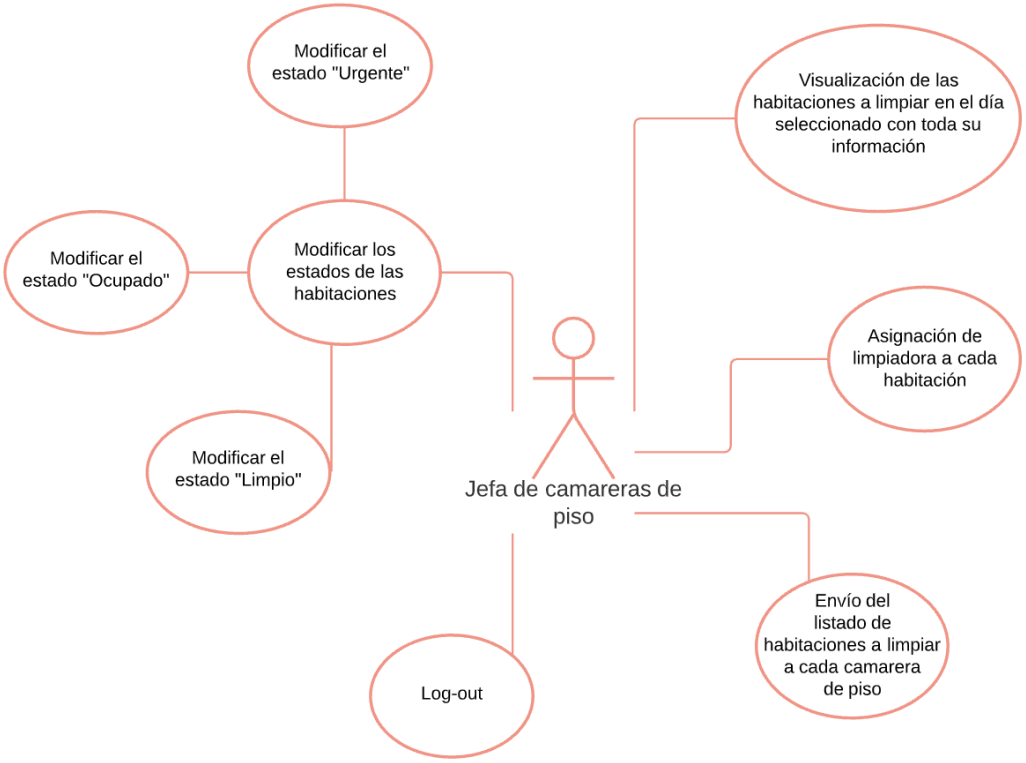

Figura 15: Casos de uso 'Jefa de camareras de piso'

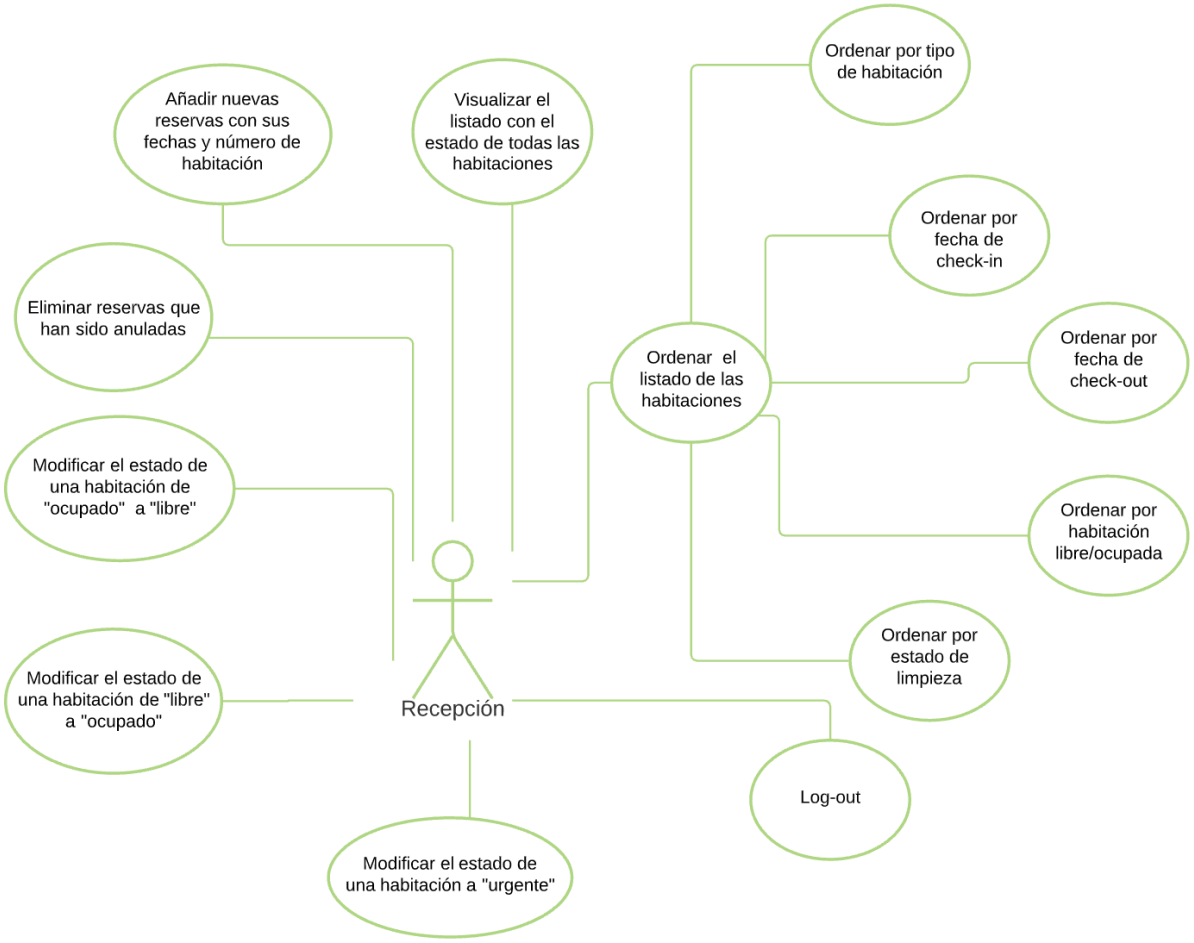

Figura 16: Casos de uso ‘recepción’

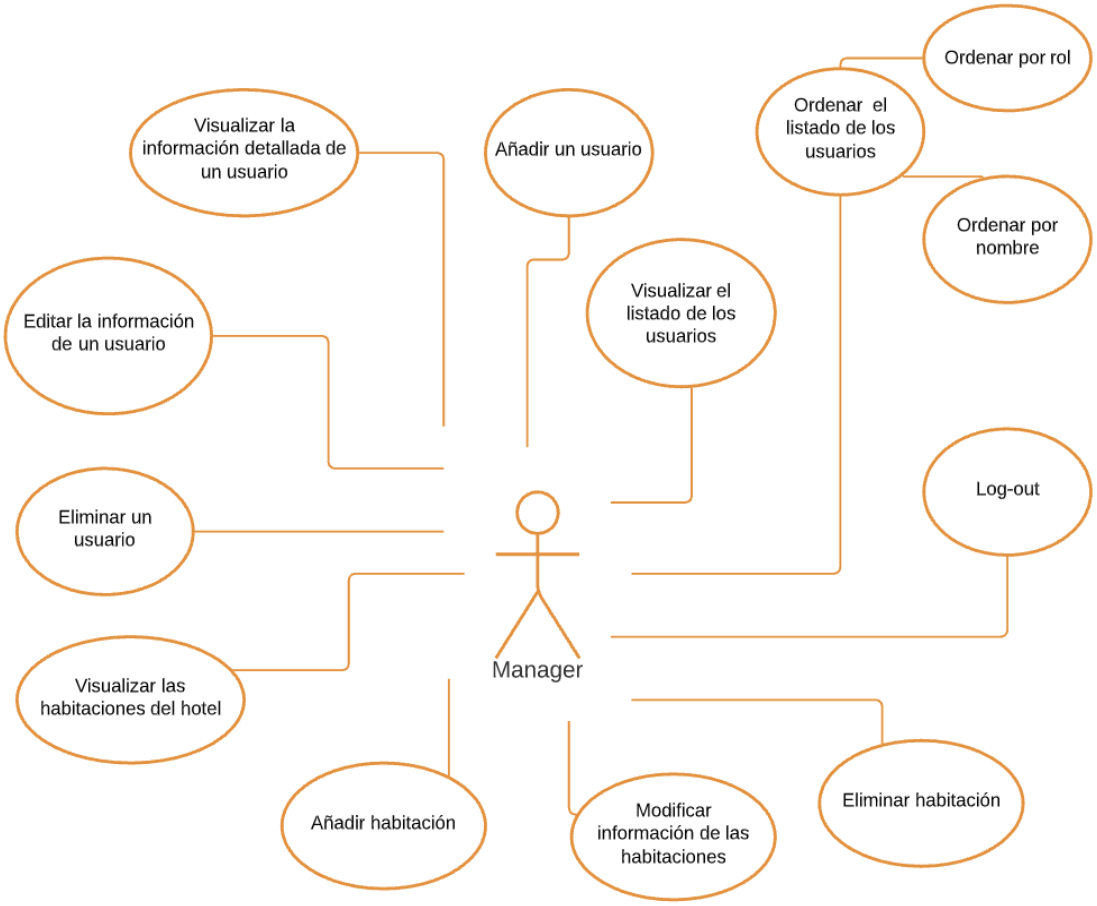

Figura 17: Casos de uso ‘manager’

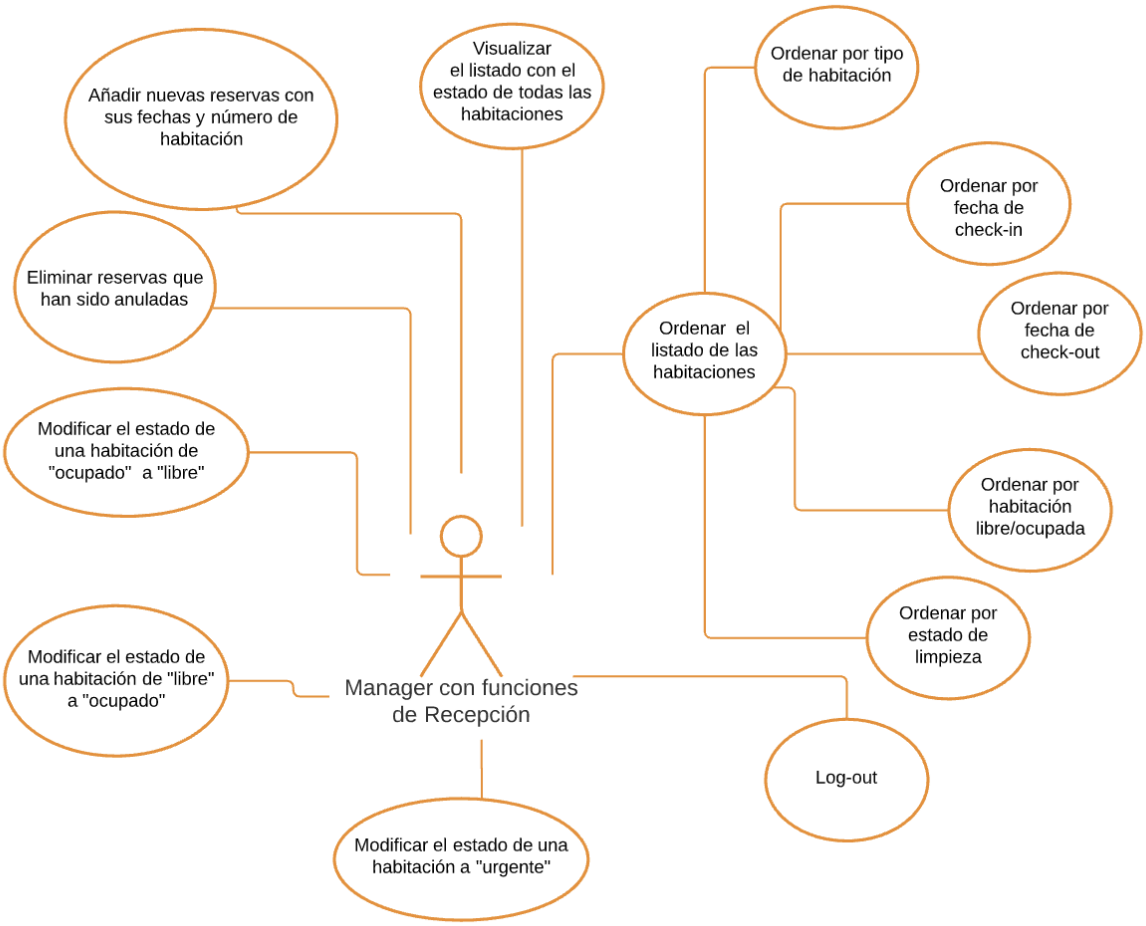

Figura 18: Casos de uso ‘manager con rol de recepción’

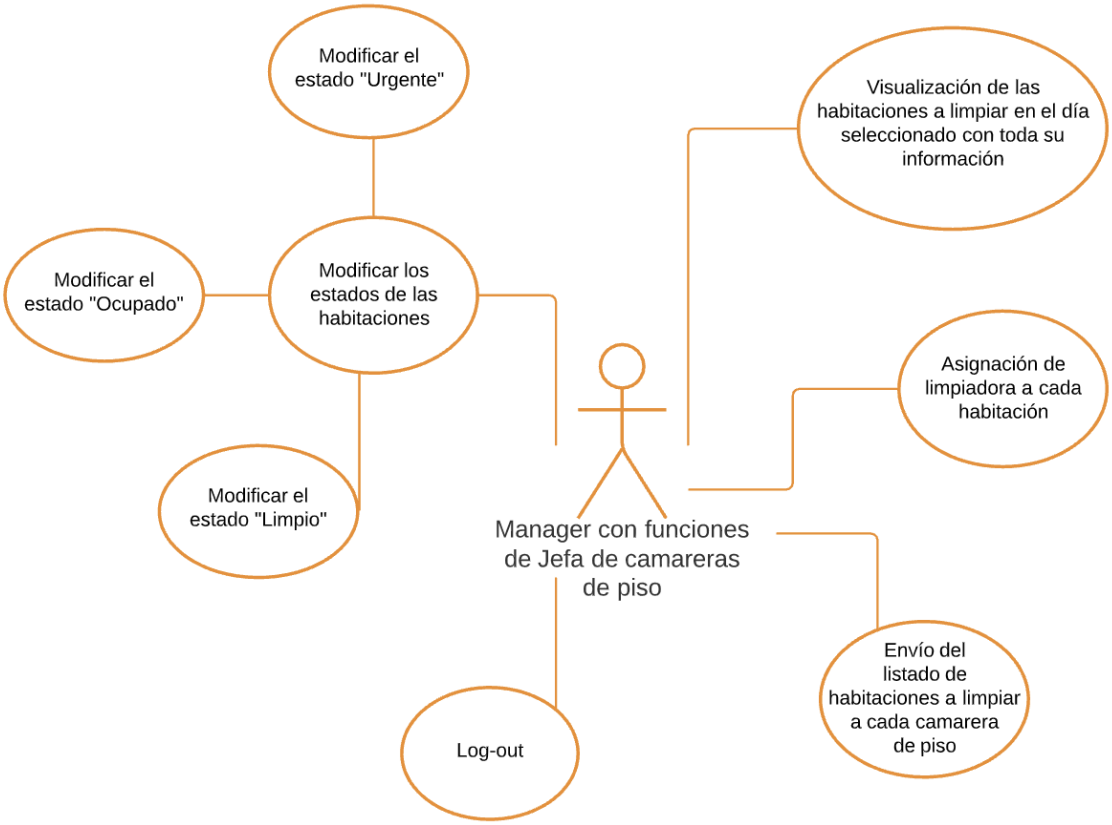

Figura 19: Casos de uso ‘manager con rol de jefa de camareras’

7. Planificación

7.1. Recursos

7.1.1. Software necesario para ejecutar en local

- Java JDK 1.8 o OpenJDK en su defecto
- Maven
- Servidor MySQL con una base de datos llamada nice
- Vue.js

7.1.2. Herramientas

7.1.2.1. Visual Studio Code

Usamos este editor de texto debido a la cantidad de plugins, herramientas, snippets que tiene y lo intuitivo que es su uso.

7.1.2.2. Spring Tool Suite

Spring Tool Suite (STS) es un IDE basado en la versión Java EE de Eclipse, pero altamente customizado para trabajar con Spring Framework y el recomendado para su uso.

7.1.2.3. Balsamiq Cloud

Esta herramienta la hemos usado para diseñar los mockup del proyecto.

7.1.2.4. Maven

Es una herramienta de software libre que ayuda a simplificar la construcción y administración de proyectos basados en Java.

La hemos usado para ahorrarnos tiempo a la hora de compilar el código, empaquetarlo e instalar las dependencias necesarias, entre ellas un servidor apache tomcat, el jpa, hibernate y el driver para la conexión a la base de datos.

7.1.2.5. Docker

Durante el proceso de desarrollo local tuvimos una serie de problemas, el principal fue que el servidor de base de datos de XAMPP funcionaba en un ordenador pero no en el otro, impidiendo así la conexión del lado del servidor con los datos. Por este motivo decidimos usar Docker. Es de código abierto y sirve para crear contenedores que funcionan como máquinas virtuales extremadamente livianas y modulares. De este modo creamos un contenedor de mysql que sabremos que nos funcionará a ambos

7.1.3. Tecnologías

7.1.3.1. Java

Una de las razones por las que elegimos éste lenguaje de programación es porque es fiable, rápido y además es un lenguaje fuertemente tipado, por lo que tenemos un mayor control sobre qué tipo de dato nos llega y cómo procesarlo correctamente

7.1.3.2. Spring boot

Spring es un framework para el desarrollo de aplicaciones de código abierto para la plataforma JAVA. La razón principal para elegir este framework fue que ambos dimos Java el primer año del ciclo y sacamos muy buenas notas. Además es tremendamente rápido y sencillo para crear el backend. Cabe mencionar que por debajo tira de Apache Tomcat para poder funcionar como servidor.

Aunque parezca extraño usar esta tecnología pues no se ha dado en el curso, resulta que dispone de un montón de herramientas que nos ha facilitado muchísimo el desarrollo.

Con hibernate hemos conseguido convertir las clases en entidades para la base de datos usando solo anotaciones.
Con JPA hemos conseguido las acciones CRUD sin necesidad de crear ninguna consulta SQL. Consiguiendo así que nuestro backend sea agnóstico al servidor sql que se vaya a utilizar.

7.1.3.3. Vue.js

A la hora de decidir qué tecnología usaríamos para el front Vue.js fue un “must”, ya que lo dimos en el ciclo, es fácil de usar, elegante y el código queda bien organizado con los componentes, así que no tuvimos dudas, Vue debía de estar sí o sí.

7.1.4. Servicios

Github lo utilizamos como plataforma para el control de versiones y tanto el dominio como el hosting para nuestra web www.nice-tidy.es estarán bajo el proveedor de servicios GoDaddy.

7.1.5. Humanos

Actualmente el equipo lo formamos nosotros dos, aunque existe la posibilidad de incluir un diseñador cuando vayamos a comercializar el software.

7.1.6. Temporal

Nuestra estimación para la realización del diseño web y tener las funcionalidades mínimas sería de un mes.

7.2. Tareas a llevar a cabo

- Hacer la página del login
- Menú lateral
- Añadir Spring Security
- Securizar rutas
- Lógica del login
- Crear las entidades usuario y habitación
- Hacer el CRUD de habitaciones
- Hacer el CRUD de usuarios
- Instalar el módulo Vue Router y añadirlo al front
- Hacer las vistas de habitaciones con su respectivo formulario
- Hacer la vista de listado de usuarios
- Hacer la vista detallada del usuario
- Vista de las reservas y editarlas
- Vista de gobernanta
- Vista de camarera
- Lógica gobernanta
- Lógica camarera de piso
- Lógica de las vistas de reservas
- Gráficos de la pantalla de inicio
- Las ordenaciones

7.3. Diagrama de flujo

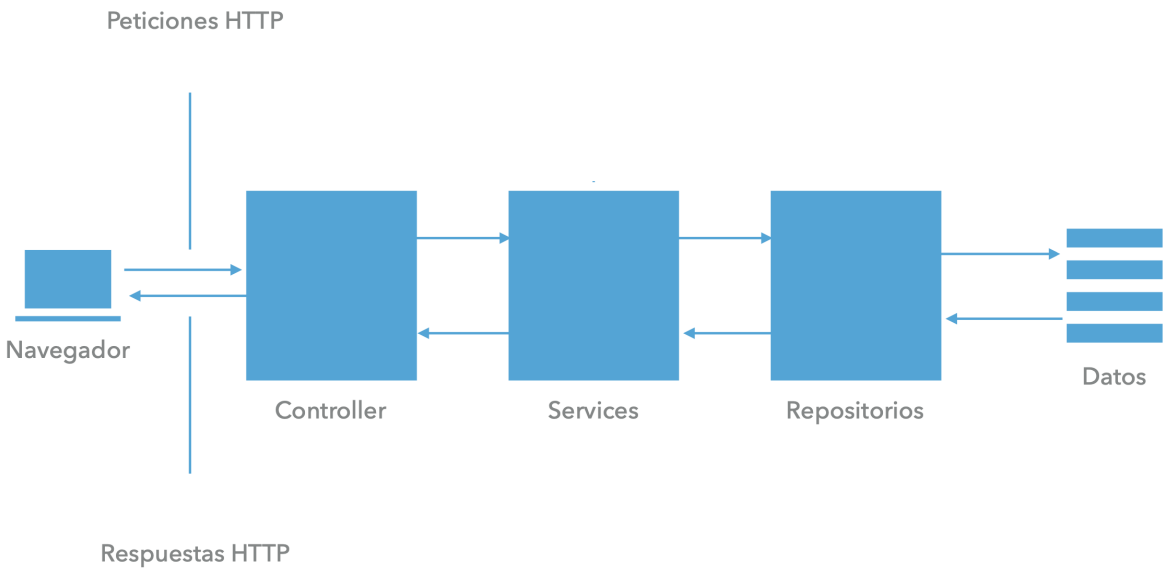

Figura 20: Diagrama de una petición

Una petición que vaya al backend seguirá el siguiente esquema:

En el controlador estarán relacionados los endpoints con sus respectivos métodos, éstos llamaran al servicio, interfaz, en cuyas implementaciones estará descrito qué lógica habrá que hacer, y por último llamará al repositorio, que no es más que una interfaz que implementa de JpaRepository, que ya tiene lógica interna para hacer un CRUD.

Hemos optado por seguir esta estructura, aunque sea más compleja, porque queremos que sea fácilmente escalable. Aunque al inicio nos lleve más tiempo montarla, nos va a simplificar mucho los cambios en el futuro.

7.4. Estructura de la base de datos

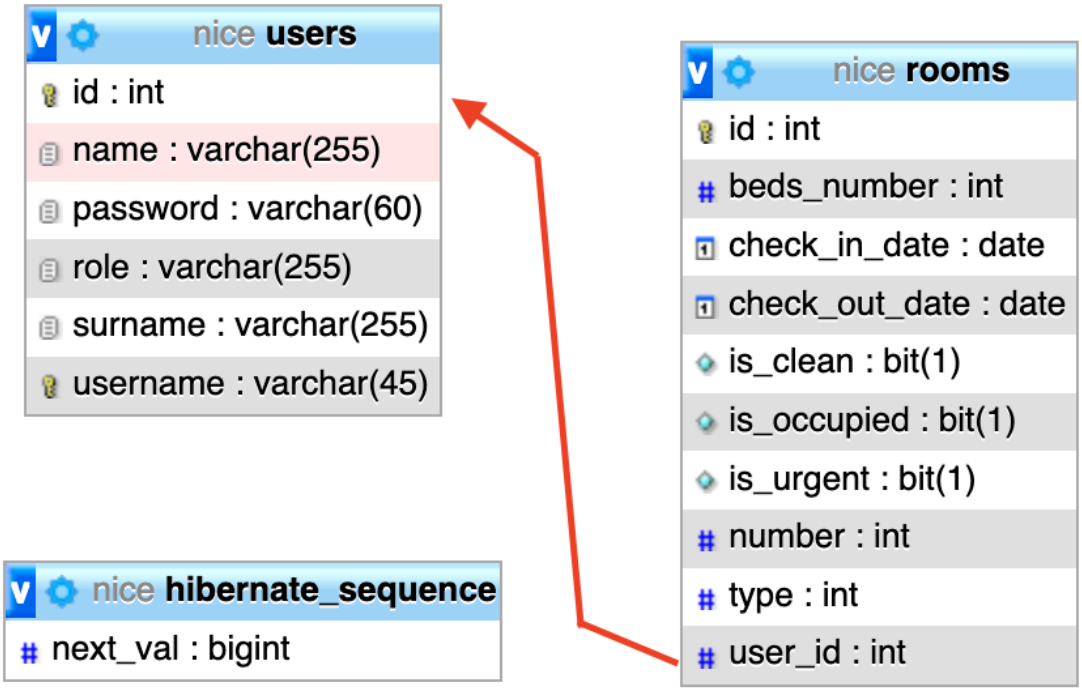

Figura 21: Estructura de la base de datos

Tenemos tres tablas en nuestra base de datos llamada nice.

Hibernate_sequence es creada por Spring Boot y se usa para controlar el campo id de los registros del resto de las tablas.

Rooms tiene la información referente a las habitaciones. En ella se guarda, entre otros datos, el número de camas que tiene, si está limpia o no, el id de la camarera que la limpia, etc.

Además tiene otra función, ya que si el campo fecha de entrada está relleno, se entiende que ya hay una reserva para dicha habitación en esa fecha.

Users es la tabla encargada de los usuarios que tienen acceso a la aplicación. En ella guardamos tanto su nombre, apellidos, nombre de usuario y contraseña. Además disponemos de 4 roles: admin, recepcionista, camarera y gobernanta.

7.5. Cronograma

Tarea	1	2	3	4	5	6	7	8	9	10	11	12
Definir el proyecto												
Diseño de pantallas												
Diseño de casos de uso												
Planificación												
Preparar el área de trabajo												
Crear la entidad habitación												
Crear la entidad usuario												
Instalar y configurar vue router												

Tarea	13	14	15	16	17	18	19	20	21
Crear el controlador de habitación									
Crear el controlador de usuario									
Crear menú lateral									
CRUD en servicio de usuario									
CRUD en servicio de habitación									
Instalar y configurar vue router									
Hacer la vista de habitaciones									
Hacer la vista de usuarios									
Hacer la vista del login									
Lógica en el backend para el login									

Tarea	22	23	24	25	26	27	28	29	30	31	32	33
Formulario de Usuarios												
Vista detallada de usuario												
Formulario de habitaciones												
Vista de reservas												
Creación y modificación de reservas												
Vista gobernanta												
Vista camarera de piso												
Lógica de gobernanta												
Lógica de camarera de piso												
Lógica de modificación de reservas												

7.6. Metodología

A la hora de programar hemos seguido la metodología colaborativa, donde dividimos el proyecto en pequeños objetivos de programación que subíamos a GitHub una vez finalizado.

Cada vez que se terminaba un objetivo decidíamos conjuntamente cuál sería el siguiente.

7.7. Estimación de costes

El despliegue para la demostración se hará en plataformas gratuitas como Heroku y Netlify.

Sin embargo, a la hora de comercializarlo utilizaremos GoDaddy, cuya tarifa incluye el dominio, email de microsoft 365, 25 bases de datos, almacenamiento ilimitado y webs ilimitadas por 120€/año

7.8. Validación

Teniendo en cuenta el breve periodo de tiempo del que disponemos, somos conscientes de que no nos dará tiempo para programar las validaciones necesarias, pero en un futuro a la hora de hacer los test unitarios y de integración seguiremos los siguientes principios:

- Que la ejecución de los tests sea rápida (menos de 1 o 2 minutos para ejecutarlas todas).
- Evitar tests frágiles usando el patrón ObjectMother que permite que los tests estén menos acoplados al código de producción.
- Seguir la pirámide de tests (que los test unitarios ocupen la gran parte de los tests y que simulen la arquitectura de las conexiones a bases de datos, etc)
- Tener una versión beta que pueda usar el personal y que así nos cuente su opinión

8. Ejecución

El siguiente enlace lleva a nuestro repositorio de GitHub donde se podrá clonar el proyecto:

https://github.com/victorfch/nice-tidy-project

9. Conclusiones

9.1. ¿Cumple con lo previsto?

Durante el proceso de desarrollo hemos logrado la funcionalidad básica, de modo que la aplicación ya es funcional, aunque quedaron algunos detalles como las ordenaciones y los gráficos de la pantalla de inicio que terminaremos una vez finalizado el ciclo.

9.2. Propuestas de mejora

Con el paso del tiempo nos gustaría hacer una serie de mejoras, entre ellas están:

- Que el lado del servidor se pueda conectar con la base de datos del hotel para sincronizar las entradas y salidas que tienen en real.
- Usar Vuex para gestionar el estado de los datos.
- Usar JWT para el login
- Para el lado del cliente cambiar Vue.js por Nuxt.js para solucionar problemas de SEO y así conseguir un Server Side Rendered.
- Poder elegir el idioma en la web.
- Guardar un histórico del listado de limpieza del día
- Opción para habitaciones no disponibles temporalmente por reforma o reparaciones.
- Uso de WebSockets para la actualización automática

11. Bibliografía

Colaboradores de Wikipedia. (2020a, enero 16). Spring Framework. Wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Spring_Framework

Colaboradores de Wikipedia. (2020b, octubre 10). Maven. Wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Maven

¿Qué es Docker? (2021). Red Hat. https://www.redhat.com/es/topics/containers/what-is-docker

Integración de eclipse y SpringSource Tool Suite. David Marco. http://www.davidmarco.es/articulo/integracion-de-eclipse-y-springsource-tool-suite

12. Anexos

Como memoria de la ejecución adjuntamos el enlace a GitHub:
https://github.com/victorfch/nice-tidy-project