***GestApart (Microservicio de Gestion de Apartamentos)***

**Índice.**

[1.Información general de la Aplicación](#_Toc534889113)

[2.Bases de Datos y Modelos](#_Toc534889114)

[3.Microservicios finalizados](#_Toc534889115)

[3.a ClienteWeb](#_Toc534889116)

[3.b UsuariosMicroservice](#_Toc534889117)

[3.c ApartamentosMicroservice](#_Toc534889118)

[3.d ReservasMicroservice](#_Toc534889119)

[3.e ConversorMicroservice](#_Toc534889120)

[3.f FotosMicroservice](#_Toc534889121)

[4.Microservicios Pendientes](#_Toc534889122)

[4.a Parking](#_Toc534889123)

[4.b Limpieza](#_Toc534889124)

[4.c Servicios](#_Toc534889125)

[5.Biografía y búsquedas](#_Toc534889126)

[6.Herramientas utilizadas](#_Toc534889127)

# 1.Información general de la Aplicación

Se trata de una aplicación con la que poder gestionar apartamentos vacacionales. No se trata de una gestión de reservas (aunque se podría incluir si fuera necesario). Pensado para empresas que gestione una cantidad elevada de apartamentos o pisos. Valido también para hoteles y hostales (con pocas modificaciones).

A parte del apartamento, es posible gestionar otros servicios como Parkings, Servicio Limpieza y escalonar la aplicación para cualquier otro servicio necesario.

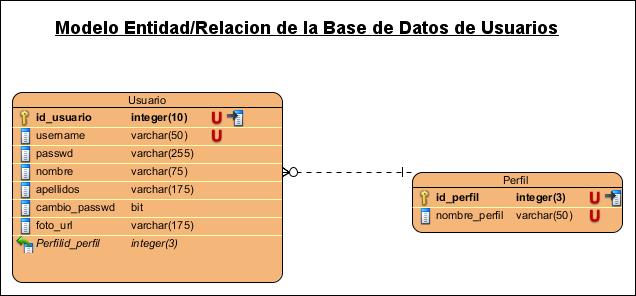
La aplicación está orientada a funcionar con Microservicios. Debido a limitaciones con el hardware, no se ha podido fraccionar la aplicación en el proyecto, pero en la memoria se indican los distintos microservicios que habría.

# 2.Bases de Datos y Modelos

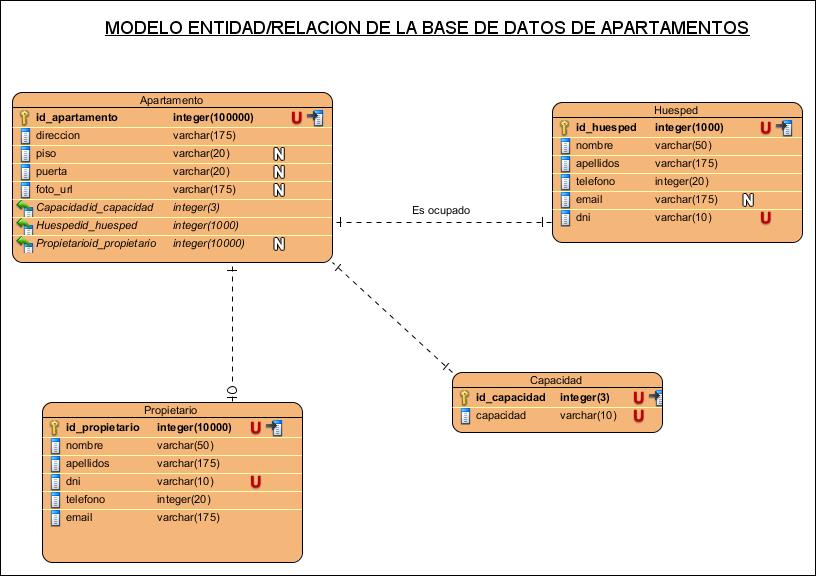
Las bases de datos se han realizado en H2(in memory) para que se pueda probar mejor. Cada una de ellas incluye un archivo **data.sql** con instrucciones para la inserción de datos de prueba.

1. **UsuariosDB:** Base de datos de los usuarios que tendrán acceso y su nivel de acceso o perfil;

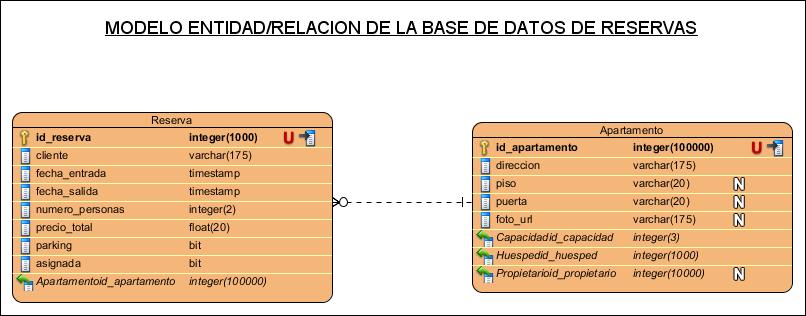
ROLE\_ADMIN, ROLE\_USER, ROLE\_CLEAN, etc.



1. **ApartamentosDB:** Base de datos de los apartamentos/pisos que dispone el cliente para su ocupación. Incluye información del huésped que estaría al cargo de la habitación, la capacidad del apartamento y el propietario del mismo.



1. ReservasDB: Base de datos de las reservas que tiene el cliente y los apartamentos relacionados.



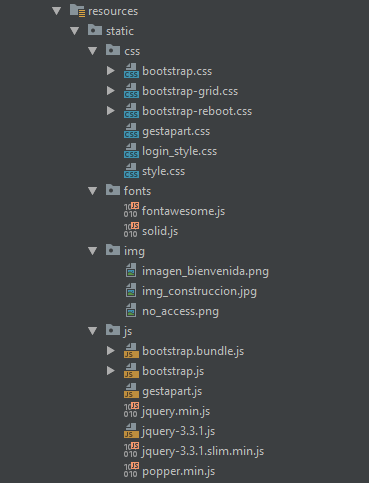
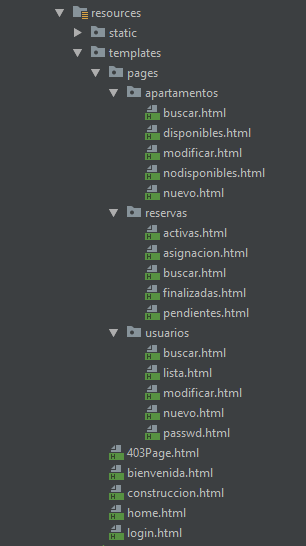
# 3.Microservicios finalizados

1. ClienteWeb
   * Login
   * SpringSecurity
   * Página de gestión de usuarios
   * Página de gestión de Apartamentos
   * Página de gestión de Reservas
2. Microservicio de Usuarios(UsuarioMicroservice)
3. Microservicio de Apartamentos(ApartamentosMicroservice)
4. Microservicio de Reservas(ReservasMicroservice)
5. Microservicio de Conversión de entidades(ConversorMicroservice)
6. Microservicio de gestión de fotos(FotosMicroservice)

A continuación, se desglosa todo el contenido.

## 3.a ClienteWeb

Código SpringMVC en el que se define todo el Front de la aplicación. La gestión de los formularios y la implementación de la seguridad. A continuación, se muestra el árbol de directorio del Front:



A destacar:

* El uso de Thymeleaf para la comunicación con los controladores y carga de resultados. También se utiliza JavaScript principalmente para cargar listas. Ya que el Máster no incluía mucho temario de este lenguaje, puede ser un poco burdo o mal estructurado, pero realiza la función para lo que está escrito. Más información en el archivo ***gestapart.js***
* Utilización de Feign para la comunicación con el resto de microservicios.
* Configuración de la carpeta de los recursos con la clase ***WebResourceConfig***:

public void addResourceHandlers(ResourceHandlerRegistry registry) {  
 registry.addResourceHandler(  
 "/img/\*\*",  
 "/css/\*\*",  
 "/js/\*\*",  
 "/fonts/\*\*",  
 "/pages/\*\*")  
 .addResourceLocations(  
 "classpath:/static/img/",  
 "classpath:/static/css/",  
 "classpath:/static/js/",  
 "classpath:/static/fonts/",  
 "classpath:/templates/pages/");  
}

* Configuración del Login y acceso a las páginas mediante roles o perfiles con SpringSecurity;
  1. ***WebSecurityConfig***, para el acceso a las páginas.

protected void configure(HttpSecurity http) throws Exception {  
  
 http.csrf().disable();  
  
 http.authorizeRequests().antMatchers("/login", "/logout", "/403page").permitAll();  
  
 http.authorizeRequests().antMatchers("/home", "/").access("hasAnyRole('ROLE\_ADMIN','ROLE\_USER','ROLE\_CLEAN')");  
  
 http.authorizeRequests().antMatchers("/apartamentos/\*", "/reservas/\*", "/parkins/\*", "/servicios/\*").access("hasAnyRole('ROLE\_USER','ROLE\_ADMIN')");  
  
 http.authorizeRequests().antMatchers("/limpieza/\*").access("hasAnyRole('ROLE\_CLEAN','ROLE\_ADMIN')");  
  
 http.authorizeRequests().antMatchers("/\*", "/usuarios/\*").access("hasAnyRole('ROLE\_ADMIN')");  
  
 http.authorizeRequests().and().exceptionHandling().accessDeniedPage("/403");  
  
 // Configuracion para el formulario de Login  
 http.authorizeRequests().and().formLogin()//  
 // Configuracion página login.  
 .loginProcessingUrl("/j\_spring\_security\_check") // URL del action del formulario  
 .loginPage("/login")//  
 .defaultSuccessUrl("/home")//  
 .failureUrl("/login?error=true")//  
 .usernameParameter("username")//  
 .passwordParameter("password")  
 // Configuracion pagina logout  
 .and().logout()  
 .logoutUrl("/logout")  
 .logoutSuccessUrl("/login")  
 .deleteCookies("JSESSIONID")  
 .invalidateHttpSession(true)  
 .clearAuthentication(true);  
  
 // Control de sesión ---------------------------  
 http.sessionManagement()  
 .sessionCreationPolicy(SessionCreationPolicy.*ALWAYS*)  
 .maximumSessions(1)  
 .expiredUrl("/login");  
}

* 1. ***UserDetailServiceImpl***, para el Login de acceso a través de una consulta con la Base de Datos del Microservicio de Usuarios

public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {  
 UserDetails userDetails = null;  
 try {  
 //Obtenemos un objeto usuario de la base de datos  
 List<UsuarioDTO> appUser = usuariosFeign.buscarUsuario(username);  
  
 //Obtenemos una lista de los roles que tiene el usuario  
 List<String> roleNames = new ArrayList<>();  
 roleNames.add(appUser.get(0).getPerfil().getNombrePerfil());  
  
 //Añadir los roles a SpringSecurity  
 List<GrantedAuthority> grantList = new ArrayList<>();  
 if (roleNames != null) {  
 GrantedAuthority authority = new SimpleGrantedAuthority(roleNames.get(0));  
 grantList.add(authority);  
 }  
 userDetails = new User(appUser.get(0).getUsername(), appUser.get(0).getPasswd(), grantList);  
 }catch (InternalAuthenticationServiceException e){  
 *log*.fatal("Error al acceder a la aplicación. Nombre de usuario y contraseña incorrectos o inexistentes");  
 } catch (Exception e){  
 *log*.fatal("Error inesperado de inicio de sesión: "+e.getMessage());  
 }  
 return userDetails;  
}

## 3.b UsuariosMicroservice

Microservicio encargado de la gestión de los usuarios y almacenamiento en la Base de Datos, realizada en H2 en memoria. Podría segmentarse en otros dos microservicios a parte, un Microservicio para la gestión de los perfiles y otro para la generación del usuario y password inicial.

A destacar:

* Conexión Feign para el almacenamiento de las fotos de los usuarios y la conversión de Entidades a DTO y viceversa.
* Clase GenerarUserPasswdService, para la generación de un nombre de usuario y un password inicial.
  1. Generar nombre de usuario:

public String generarUsername(String nombre, String apellidos) {  
 nombre = StringUtils.*stripAccents*(nombre).trim();  
 apellidos = StringUtils.*stripAccents*(apellidos).trim();  
 while (apellidos.length() < 6) apellidos += "0";  
 String usernameGenerado = (nombre.substring(0, 1) + apellidos.substring(0, 6).trim()).toLowerCase();  
 int count = 0;  
 while (existeUsuario(usernameGenerado)) {  
 count++;  
 usernameGenerado += count;  
 }  
 return usernameGenerado;  
}

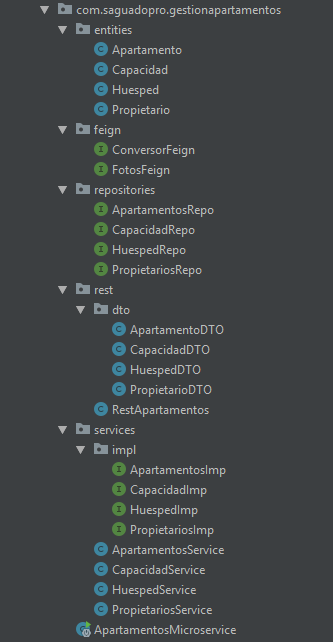
* 1. Generar password inicial:

public String generarPasswd(String username) {  
 StringBuilder passwdAutogenerado = new StringBuilder(username.replace((username.charAt(0)), (username.toUpperCase().charAt(0))));  
 LocalTime tiempo = LocalTime.*now*();  
 StringBuilder horaSb = new StringBuilder();  
 horaSb.append(tiempo.getHour()).append(tiempo.getMinute()).append(tiempo.getSecond());  
 passwdAutogenerado.append(horaSb);  
 return passwdAutogenerado.toString();  
}

* Documentado con javadoc y swagger.

## 3.c ApartamentosMicroservice

Microservicio encargado de la gestión de los apartamentos/pisos/habitaciones que pueda tener el cliente y su correspondiente almacenamiento en una Base de Datos H2 en memoria para pruebas. También podría segmentarse en varios microservicios; Microservicio para la gestión de Propietarios, otro para la gestión de Huéspedes y por ultimo uno para las Capacidades. Debido a limitaciones de hardware no se han podido realizar, pero el código está estructurado para que sea fácil dividirlo en microservicios.



A destacar:

* Conexión Feign para el almacenamiento de las fotos de los usuarios y la conversión de Entidades a DTO y viceversa.
* Documentado con javadoc y swagger

## 3.d ReservasMicroservice

Microservicio encargado de gestionar las reservas. Base de datos propia H2 en memoria. También se encargaría de procesar las reservas entrantes de otros sitios, he utilizado por ejemplo de Booking.com, utilizando un response de prueba y gracias a un conversor de XML a objeto puedo tratar y transformar esas reservas a las de la propia aplicación.

A destacar:

* Conexión Feign para la conversión de Entidades a DTO y viceversa.
* Método para la conversión de reservas de booking.com utilizándola librería **JAXB API** para transformar la respuesta del servidor de **booking** a una reserva del apropia aplicación.

public void getBookinReservas(String xmlFile){  
 Reservations reservations;  
 try {  
 InputStream inputStream = new ByteArrayInputStream(xmlFile.getBytes());  
 JAXBContext jaxbContext = JAXBContext.*newInstance*(Reservations.class);  
 Unmarshaller unmarshaller = jaxbContext.createUnmarshaller();  
 reservations = (Reservations) unmarshaller.unmarshal(inputStream);  
 System.*out*.println(reservations);  
 for (Reservation reservation : reservations.getListaReservas()) {  
 Reserva reserva = new Reserva();  
 reserva.setIdReserva(reservation.getId());  
 reserva.setIdApartamento(reservation.getRoom().getId());  
 reserva.setCliente(reservation.getCustomer().getFirst\_name()+" "+reservation.getCustomer().getLast\_name());  
 reserva.setNumero\_personas(reservation.getRoom().getNumberofguests());  
 reserva.setFechaEntrada(LocalDate.*parse*(reservation.getRoom().getArrival\_date()));  
 reserva.setFechaSalida(LocalDate.*parse*(reservation.getRoom().getDeparture\_date()));  
 reserva.setPrecioTotal(reservation.getRoom().getTotalprice());  
 reservasRepo.save(reserva);  
 }  
 } catch (JAXBException e) {  
 *logger*.warn("Las reserva procedentes de Booking.com no se han podido parsear: "+e.getErrorCode());  
 }  
}

* Documentado con javadoc y swagger

## 3.e ConversorMicroservice

Microservicio encargado de convertir todas las entidades de la aplicación a objetos DTO y viceversa y poder así trabajar mejor con las Bases de Datos. Solo necesita conexión por Feign con el Microservicio de gestión de fotos y está documentado con javadoc y swagger.

## 3.f FotosMicroservice

Microservicio encargado de codificar y decodificar las imágenes para su mejor manejo en las comunicaciones http y en guardar dicha imagen. En este caso se guardarían dentro de una carpeta en **resources** dependiendo de su origen. Modifica también el nombre de la imagen a guardar para que sea identificativo de cada usuario o apartamento.

A destacar:

* Uso de la librería **java-image-scaling** para escalar las imágenes a un tamaño adecuado, que no sea la imagen muy pesada y así agilizar la comunicación.

private BufferedImage resize(BufferedImage img, int newW, int newH) {  
 ResampleOp resizeOp = new ResampleOp(newW, newH);  
 resizeOp.setFilter(ResampleFilters.*getLanczos3Filter*());  
 return resizeOp.filter(img, null);  
}

* Documentado con javadoc y swagger

# 4.Microservicios Pendientes

## 4.a Parking

Microservicio encargado de la gestión de las plazas de parking y su almacenamiento en una Base de Datos que indique si está disponible o no. El usuario tendría la posibilidad de ver los parkings que estuviesen disponibles y asignar una plaza a un apartamento o huésped. Se comunicaría con los Microservicios Apartamentos y Reservas.

## 4.b Limpieza

Microservicio encargado de la gestión de la limpieza de los apartamentos. Habría un encargado que solo tendría acceso a esta parte de la aplicación y podría gestionar si el apartamento, una vez desocupado, ya está limpio y disponible o aún no se ha preparado para su uso. De esta forma se podría gestionar mejor el servicio de limpieza. Este Microservicio estaría orientado a hoteles o edificio de apartamentos.

## 4.c Servicios

Microservicio encargado de gestionar productos de valor añadido. Contendría una serie de servicio de ocio o restauración con los datos necesarios para poder informar a los clientes si piden algún servicio o tener los datos necesarios para contactar con ellos, por ejemplo, taxis o restaurantes.

# 5.Biografía y búsquedas

Lo que he tenido que buscar y de donde lo he buscado para hacerlo.

* Spirng Security: [*https://o7planning.org/en/11705/create-a-login-application-with-spring-boot-spring-security-jpa*](https://o7planning.org/en/11705/create-a-login-application-with-spring-boot-spring-security-jpa)
* Añadir recursos estáticos para Thymeleaf: [*https://memorynotfound.com/adding-static-resources-css-javascript-images-thymeleaf/*](https://memorynotfound.com/adding-static-resources-css-javascript-images-thymeleaf/)
* JAXB para un XML en Java: [*https://springframework.guru/using-jaxb-for-xml-with-java/*](https://springframework.guru/using-jaxb-for-xml-with-java/)
* Manejar foto desde formulario:[*http://www.mkyong.com/spring-boot/spring-boot-file-upload-example/*](http://www.mkyong.com/spring-boot/spring-boot-file-upload-example/)
* Comunicación con booking.com: [*https://connect.booking.com/user\_guide/site/en-US/b\_xml-reservationssummary/*](https://connect.booking.com/user_guide/site/en-US/b_xml-reservationssummary/)

# 6.Herramientas utilizadas

* **Intellij** con plugins Lombok y Bootstrap
* **POSTMAN** para pruebas
* **VisualParadigm UML** para los diagramas de Bases de Datos
* **Microsoft Word** para la memoria