

# Trabajo Práctico Integrador

**Materia:** Desarrollo de aplicaciones cliente servidor

**Grupo:** 1

**Integrantes:**

- Rios, Lucas
- Fernández, Ian

**Año:** 2022

# INTRODUCCIÓN:

En el presente resumen vamos a mostrar el trabajo realizado por el grupo, el cual consiste en una aplicación integrada para esta primera entrega por el backend. El desarrollo de esta aplicación fue realizado en el lenguaje Java con el framework Spring. Mostraremos la arquitectura para la aplicación elegida, los diagramas de entidad relación y las herramientas de desarrollo elegida.

# ESCENARIO:

En vísperas del mundial de Qatar 2022 la empresa Fantastic Tour (FANTUR S.A.) ha solicitado a los alumnos de la materia Desarrollo de Aplicaciones Cliente-Servidor el desarrollo de un sistema para la venta y administración de paquetes turísticos.

El sistema debe permitir a los clientes registrar en forma electrónica, realizar consultas de paquetes, reservar paquetes y abordar los mismos por diferentes medios (tarjetas de crédito u otros sistemas de pago on-line) y enviar publicidad (vía e-mail) a sus clientes.

La mayoría de los paquetes turísticos que la empresa comercializa están compuestos por pasajes aéreos o en micro, estadía en hoteles, seguros médicos COVID-19 y entradas a espectáculos. La aplicación debe contemplar el armado y cotización de estos paquetes turísticos por parte de los administradores de la empresa, pudiendo establecerse cuáles paquetes están disponibles o hasta que fecha pueden adquirirse.

Debido a las imposiciones impuestas por los organismos de control que rigen la actividad del sector, las agencias de turismo (incluida FANTUR) deben solicitar permisos al organismo de contralor antes de confirmar las operaciones a sus clientes. Este tipo de solicitudes deber ser realizar en forma on-line y por medio de un web-service que el organismo provee.

## **Información del servicio**

Este servicio no fue desarrollado por los alumnos de la cátedra Desarrollo de Aplicaciones Cliente-Servidor, por lo cual la tasa de error del mismo es aleatoria y deberá ser tomada en cuenta a la hora de utilizarlo.

Contar con una aplicación para teléfonos móviles donde los usuarios puedan consultar y operar sería una muy buena ventaja competitiva para esta empresa.

## **Consideraciones:**

Consideramos que para cada paquete creado por algún administrador en el sistema este deberá crearse las entidades las cuales componen dicho paquete que son, residencia y las habitaciones a las cuales hospedarán los clientes, la cobertura médica y los servicios que propone, los partidos del mundial y las entradas, por último el medio de viaje y los pasajes.

# Arquitectura seleccionada

Para la resolución del problema se optó por un servicio web, desarrollado con la tecnología de Java en Spring boot. El servicio web seguirá una arquitectura de API REST a partir de microservicios.

## Motivo de arquitectura seleccionada

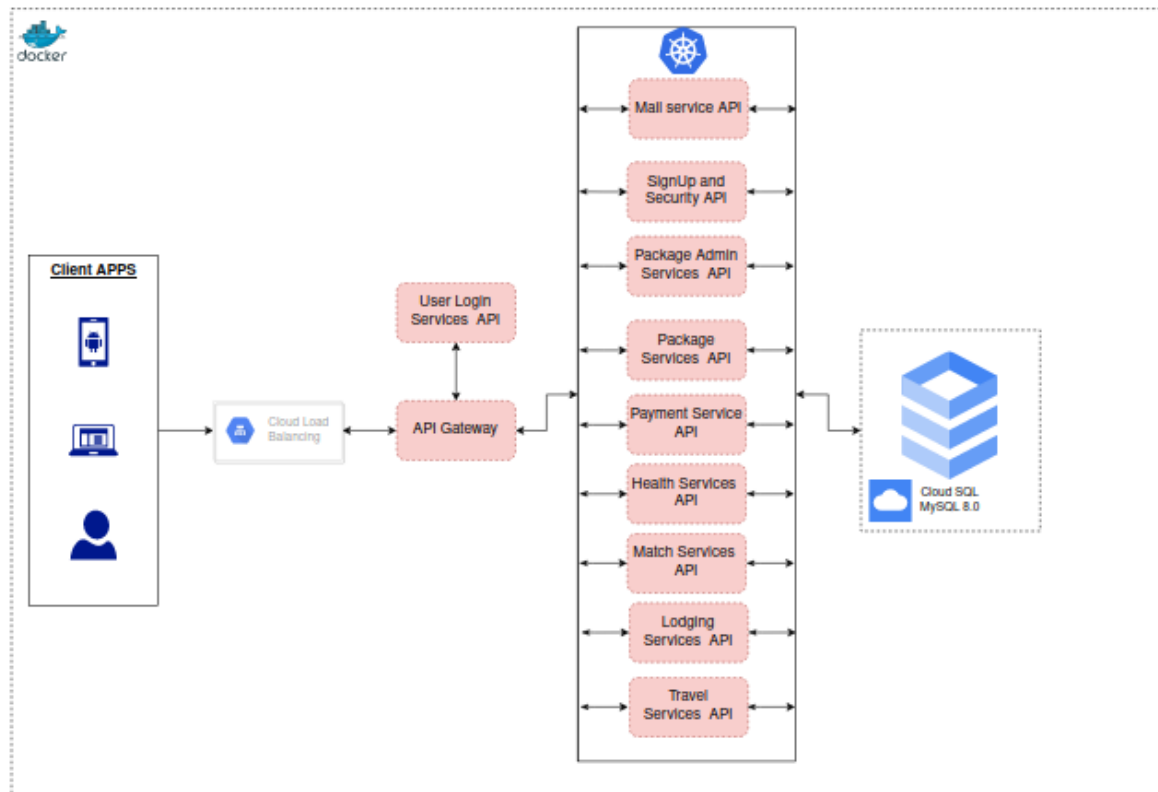
La utilización de microservicios nos propone poder centrarnos en pequeñas aplicaciones, pequeños módulos o servicios que va componer mi aplicación completa. De tal manera que optamos por la realización de estos servicios:

- Servicio de partidos de fútbol
- Servicio de cobertura médica
- Servicio de viajes
- Servicio de alojamiento
- Servicio de administración de paquetes
- Servicio de paquetes para el cliente
- Servicio de mail
- Servicio de formas de pago
- Servicio de sing up and login
- Servicio de seguridad

El servicio de administración de paquetes es el servicio utilizado por la adminitracion de pagina web, la misma se va a servir de otros servicios (Partidos de futbol, cobertura medica, viajes y alojamiento) las cuales tambien son utilizadas por los administradores del servicio web. EL servicio de mail será utilizado para notificar al cliente de distintos eventos y promociones de viajes pero la misma no repercute en nada con respecto al servicio de paquetes de administración de paquetes. El servicio de paquetes para el cliente, será el servicio que utilizará el cliente, donde podrá consultar y comprar los paquetes que necesite. El servicio de paquetes para el cliente se servirá del servicio de pago.

Como podemos ver, pensar en esta arquitectura nos lleva a poder individualizar cada módulo, haciéndolos trabajar de forma individual y separada, interactuando solo en algunas ocasiones.

## Gráfico del microservicio



El API Gateway es la puerta de entrada, la cual se sirve del servicio de Login y Seguridad para permitir autenticar al usuario y conseguir el rol que este cumple dentro del sistema. De acuerdo al rol que cumple tendrá distintos permisos, lo cual le permitirá o no acceder a distintos servicios como por ejemplo, el ROL de administrador podrá acceder a todos los servicios, pero el cliente solo podrá acceder a los del paquete destinados al usuario. Una persona que no se encuentra autenticada podrá acceder al servicio de registro o SingUp y también al servicio de Login.

## Tecnología utilizada

### Herramientas de desarrollo

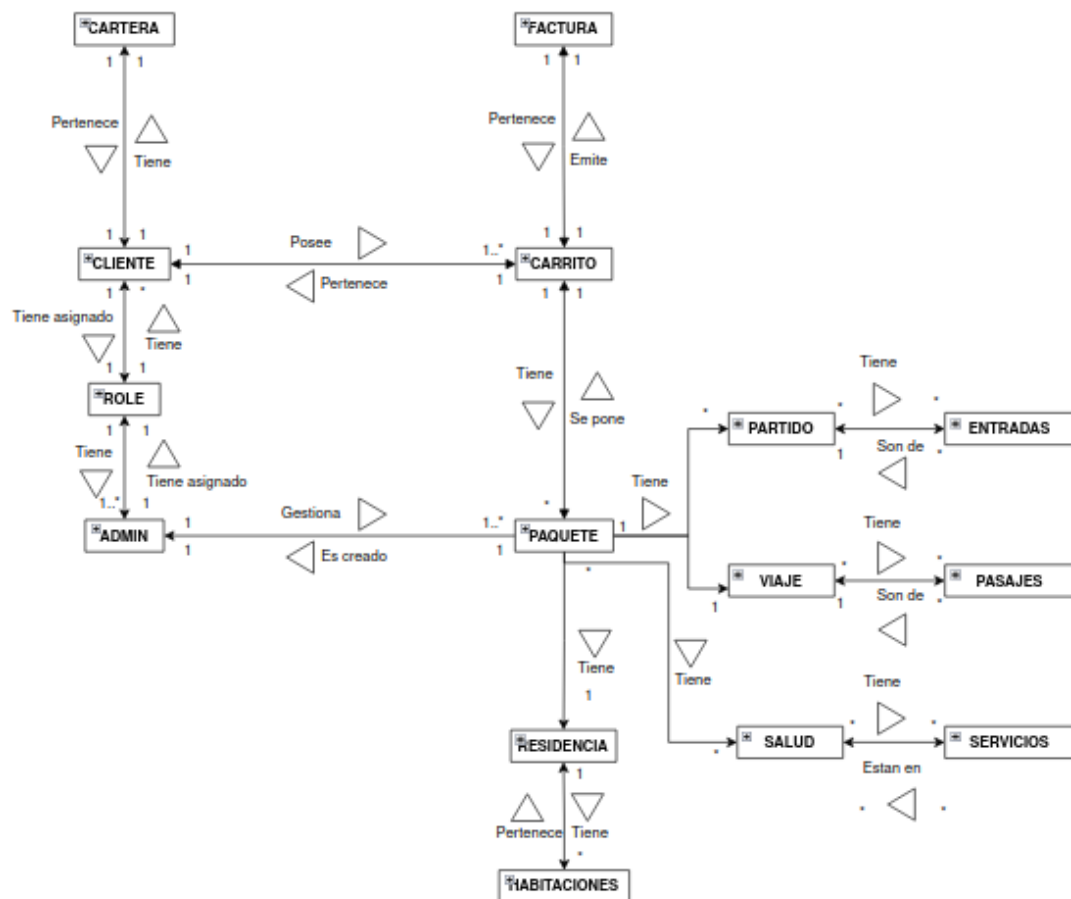
- Java 11 : Lenguaje de programación
- Spring 2.0.7 : Framework
- MYSQL 8.0 : Base de datos relacional
- DBeaver : Gestor de base de datos
- Maven : Gestor de proyectos
- Git : Sistema de control de versiones
- Github : Plataforma de control de versiones

## Tecnologías de desarrollo

- Lombok
- Spring JPA
- Spring cloud
- Eureka server y client
- Gateway cloud

## Diagramas

### Diagrama de negocio



# APIS

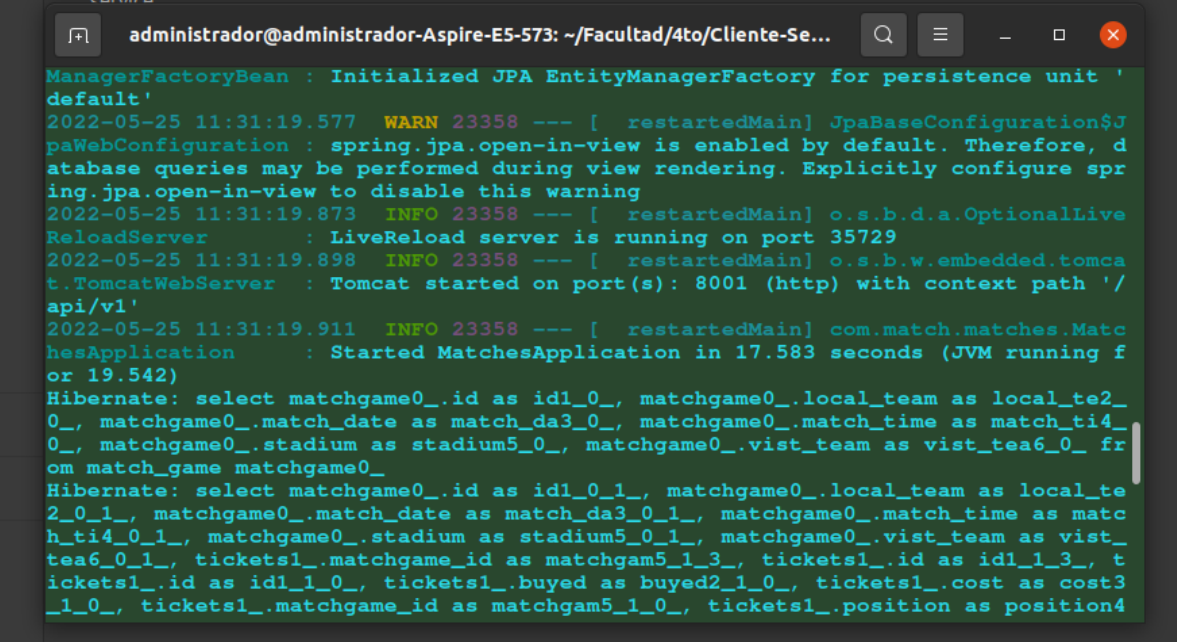
En la siguiente sección explicaremos las apis que desarrollamos.

## Servicio de partidos de fútbol

Este servicio está realizado con el fin de brindar la posibilidad de crear cada partido y cargar entradas. A su vez también se puede listar los partidos y consultar uno en particular. El administrador lo debe crear cuando quiere crear un paquete.

- Nombre : match-service
- Endpoint : “api/v1/match”

## Servicio en ejecución



```
administrador@administrador-Aspire-E5-573: ~/Facultad/4to/Cliente-Se...
ManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit '
default'
2022-05-25 11:31:19.577 WARN 23358 --- [ restartedMain] JpaBaseConfiguration$J
paWebConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, d
atabase queries may be performed during view rendering. Explicitly configure spr
ing.jpa.open-in-view to disable this warning
2022-05-25 11:31:19.873 INFO 23358 --- [ restartedMain] o.s.b.d.a.OptionalLive
ReloadServer : LiveReload server is running on port 35729
2022-05-25 11:31:19.898 INFO 23358 --- [ restartedMain] o.s.b.w.embedded.tomca
t.TomcatWebServer : Tomcat started on port(s): 8001 (http) with context path '/
api/v1'
2022-05-25 11:31:19.911 INFO 23358 --- [ restartedMain] com.match.matches.Matc
hesApplication : Started MatchesApplication in 17.583 seconds (JVM running f
or 19.542)
Hibernate: select matchgame0_.id as id1_0_, matchgame0_.local_team as local_te2_
0_, matchgame0_.match_date as match_da3_0_, matchgame0_.match_time as match_ti4_
0_, matchgame0_.stadium as stadium5_0_, matchgame0_.vist_team as vist_tea6_0_ fr
om match_game matchgame0_
Hibernate: select matchgame0_.id as id1_0_1_, matchgame0_.local_team as local_te
2_0_1_, matchgame0_.match_date as match_da3_0_1_, matchgame0_.match_time as matc
h_ti4_0_1_, matchgame0_.stadium as stadium5_0_1_, matchgame0_.vist_team as vist_
tea6_0_1_, tickets1_.matchgame_id as matchgam5_1_3_, tickets1_.id as id1_1_3_, t
ickets1_.id as id1_1_0_, tickets1_.buyed as buyed2_1_0_, tickets1_.cost as cost3
_1_0_, tickets1_.matchgame_id as matchgam5_1_0_, tickets1_.position as position4
```

# Dominio

## Partido de fútbol

```
public class MatchGame {
    @Id
    @GeneratedValue(strategy=GenerationType.IDENTITY)
    private Long id;

    @Column(name = "vistTeam", nullable = false, updatable = true)
    private String vistTeam;

    @Column(name = "localTeam", nullable = false, updatable = true)
    private String localTeam;

    @NotNull
    @Column(name = "matchDate", nullable = false, updatable = true)
    private LocalDate matchDate;

    @NotNull
    @Column(name = "matchTime", nullable = false, updatable = true)
    private String matchTime;

    @Column(name = "stadium", nullable = false, updatable = true)
    private String stadium;

    @OneToMany(mappedBy = "matchgame", cascade = CascadeType.ALL, orphanRemoval = true)
    private List<MatchTicket> tickets = new ArrayList<MatchTicket>();
}
```

## Entradas de fútbol

```
public class MatchTicket {
    @Id
    @GeneratedValue(strategy=GenerationType.IDENTITY)
    private Long id;

    @OneToOne
    private MatchGame matchgame;

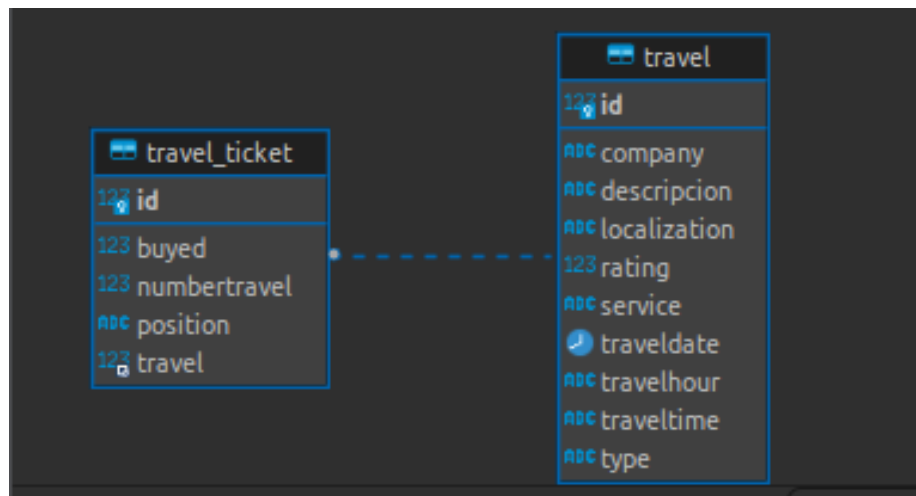
    @Column(name = "position", nullable = false, updatable = true)
    private String position;

    @Column(name = "buyed", nullable = false, updatable = true)
    private Boolean buyed = false;

    @NotNull
    @Column(name = "cost", nullable = false, updatable = true)
    private Double cost;
}
```



## Persistencia en Base de datos

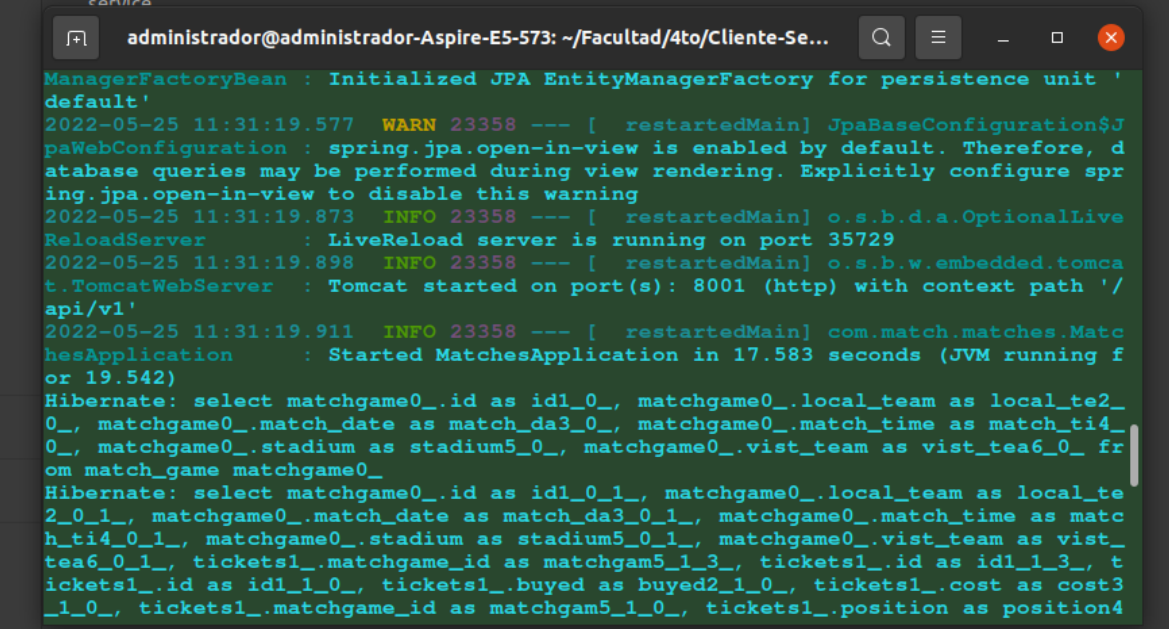


# Servicio de cobertura médica

Este servicio está realizado con el fin de permitir crear y consultar los servicios de cobertura médica para los paquetes.

- Nombre : health-insurance-service
- Endpoint : “api/v1/health”

## Servicio en ejecución



The screenshot shows a terminal window with the following content:

```
administrador@administrador-Aspire-E5-573: ~/Facultad/4to/Cliente-Se...
ManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit '
default'
2022-05-25 11:31:19.577 WARN 23358 --- [ restartedMain] JpaBaseConfiguration$J
paWebConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, d
atabase queries may be performed during view rendering. Explicitly configure spr
ing.jpa.open-in-view to disable this warning
2022-05-25 11:31:19.873 INFO 23358 --- [ restartedMain] o.s.b.d.a.OptionalLive
ReloadServer : LiveReload server is running on port 35729
2022-05-25 11:31:19.898 INFO 23358 --- [ restartedMain] o.s.b.w.embedded.tomca
t.TomcatWebServer : Tomcat started on port(s): 8001 (http) with context path '/
api/v1'
2022-05-25 11:31:19.911 INFO 23358 --- [ restartedMain] com.match.matches.Matc
hesApplication : Started MatchesApplication in 17.583 seconds (JVM running f
or 19.542)
Hibernate: select matchgame0_.id as id1_0_, matchgame0_.local_team as local_te2_
0_, matchgame0_.match_date as match_da3_0_, matchgame0_.match_time as match_ti4_
0_, matchgame0_.stadium as stadium5_0_, matchgame0_.vist_team as vist_tea6_0_ fr
om match_game matchgame0_
Hibernate: select matchgame0_.id as id1_0_1_, matchgame0_.local_team as local_te
2_0_1_, matchgame0_.match_date as match_da3_0_1_, matchgame0_.match_time as matc
h_ti4_0_1_, matchgame0_.stadium as stadium5_0_1_, matchgame0_.vist_team as vist_
tea6_0_1_, tickets1_.matchgame_id as matchgam5_1_3_, tickets1_.id as id1_1_3_, t
ickets1_.id as id1_1_0_, tickets1_.buyed as buyed2_1_0_, tickets1_.cost as cost3
_1_0_, tickets1_.matchgame_id as matchgam5_1_0_, tickets1_.position as position4
```

## Dominio

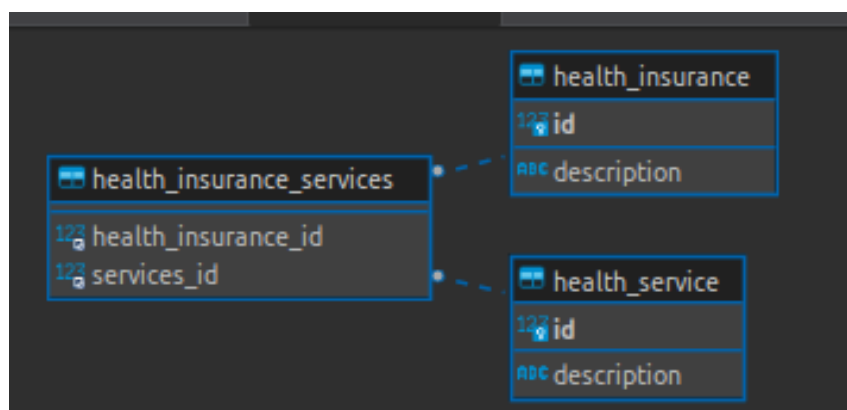
### Seguro médico

```
public class HealthInsurance {  
    @Id  
    @GeneratedValue(strategy=GenerationType.IDENTITY)  
    private Long id;  
  
    @Column(name = "description", nullable = false, updatable = true)  
    private String description;  
  
    @ManyToMany(cascade = {})  
    private List<HealthService> services = new ArrayList<HealthService>();  
}
```

### Servicios de salud

```
public class HealthService {  
    @Id  
    @GeneratedValue(strategy=GenerationType.IDENTITY)  
    private Long id;  
  
    @Column(name = "description", nullable = false, updatable = true)  
    private String description;  
}
```

## Persistencia en Base de datos

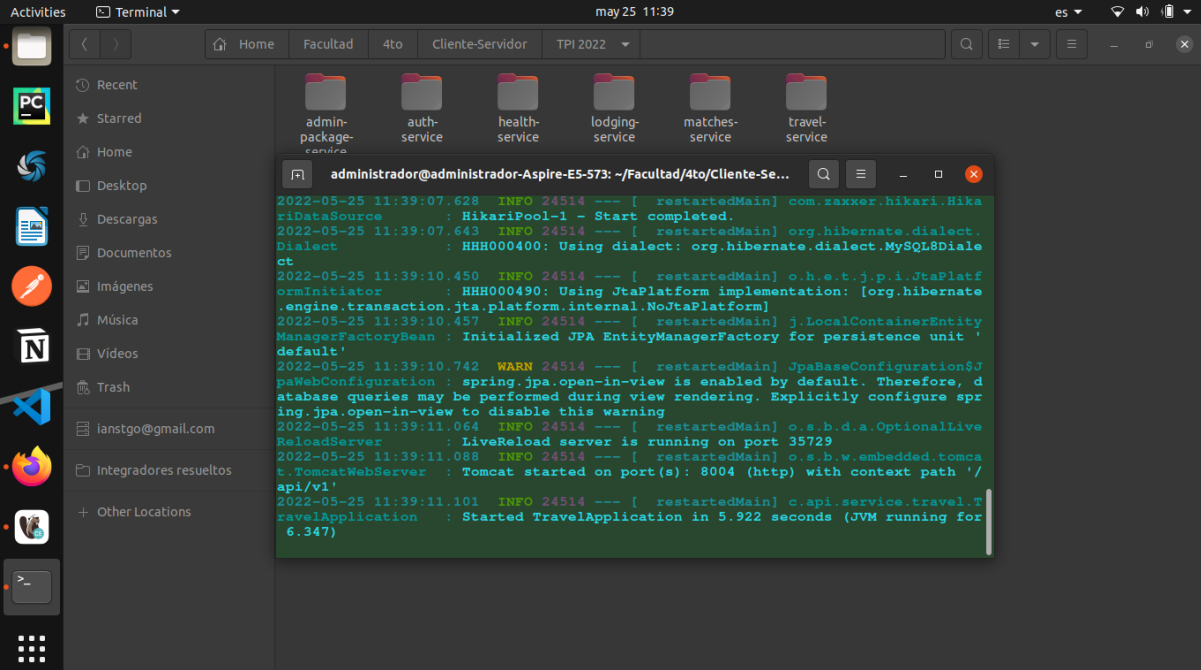


# Servicio de viajes

Este servicio está realizado con el fin de brindar la posibilidad de crear viajes y sus pasajes. Los viajes pueden ser de vuelo o por otro medio. Estos luego pueden ser consultados para poder añadirlos a un paquete de viaje.

- Nombre : travel-service
- Endpoint : “api/v1/travel”

## Servicio en ejecución



```
2022-05-25 11:39:07.628 INFO 24514 --- [ restartedMain] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Start completed.
2022-05-25 11:39:07.643 INFO 24514 --- [ restartedMain] org.hibernate.dialect.Dialect : HHH000400: Using dialect: org.hibernate.dialect.MySQL8Dialect
2022-05-25 11:39:10.450 INFO 24514 --- [ restartedMain] o.h.e.t.j.p.i.JtaPlatformInitiator : HHH000490: Using JtaPlatform implementation: [org.hibernate.engine.transaction.jta.platform.internal.NoJtaPlatform]
2022-05-25 11:39:10.457 INFO 24514 --- [ restartedMain] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'
2022-05-25 11:39:10.742 WARN 24514 --- [ restartedMain] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, database queries may be performed during view rendering. Explicitly configure spring.jpa.open-in-view to disable this warning
2022-05-25 11:39:11.064 INFO 24514 --- [ restartedMain] o.s.b.d.a.OptionalLiveReloadServer : LiveReload server is running on port 35729
2022-05-25 11:39:11.088 INFO 24514 --- [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8004 (http) with context path '/api/v1'
2022-05-25 11:39:11.101 INFO 24514 --- [ restartedMain] c.api.service.travel.TravelApplication : Started TravelApplication in 5.922 seconds (JVM running for 6.347)
```

## Dominio

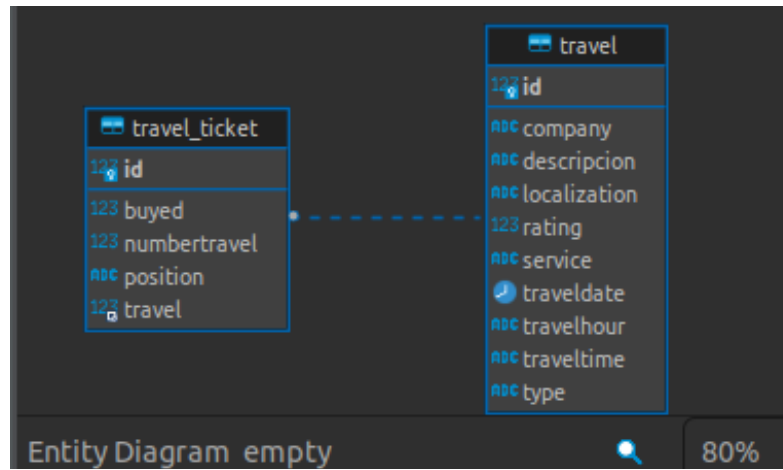
### Viaje

```
public class Travel {  
    @Id  
    @GeneratedValue(strategy=GenerationType.IDENTITY)  
    private Long id;  
  
    @Column(name = "rating", nullable = false, updatable = true)  
    private Double rating = 0.0;  
  
    @Column(name = "descripcion", nullable = false, updatable = true)  
    private String descripcion;  
|  
    @Column(name = "traveltime", nullable = false, updatable = true)  
    private String travelTime;  
  
    @Column(name = "traveldate", nullable = false, updatable = true)  
    private LocalDate travelDate;  
  
    @Column(name = "travelhour", nullable = false, updatable = true)  
    private String travelHour;  
  
    @Column(name = "localization", nullable = false, updatable = true)  
    private String localization;  
  
    @Column(name = "type", nullable = false, updatable = true)  
    private String type;  
  
    @Column(name = "service", nullable = false, updatable = true)  
    private String service;  
  
    @Column(name = "company", nullable = false, updatable = true)  
    private String company;  
  
    @OneToMany(mappedBy = "travel", cascade = {CascadeType.PERSIST, CascadeType.REMOVE, CascadeType.REFRESH}, orph  
    private List<TravelTicket> tickets = new ArrayList<TravelTicket>();  
}
```

### Pasajes

```
public class TravelTicket {  
    @Id  
    @GeneratedValue(strategy=GenerationType.IDENTITY)  
    private Long id;  
  
    @Column(name = "numbertravel", nullable = false, updatable = true)  
    private Long numberTravel;  
  
    @Column(name = "bued", nullable = false, updatable = true)  
    private Boolean bued;  
  
    @Column(name = "position", nullable = false, updatable = true)  
    private String position;  
  
    @OneToOne(cascade = {})  
    @JoinColumn(name="travel", nullable = false, updatable = true)  
    private Travel travel;  
}
```

## Persistencia en Base de datos

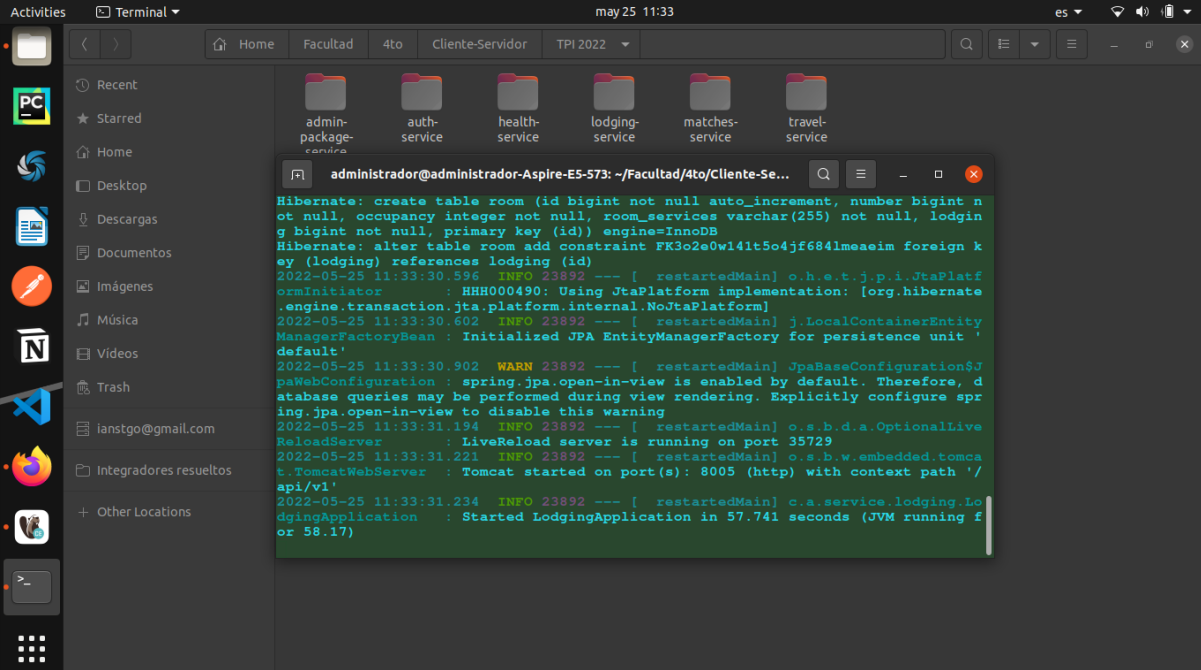


# Servicio de alojamiento

Este servicio está realizado con el fin de brindar la posibilidad de crear alojamientos y también las habitaciones. Estos luego podrán ser consultados y añadidos a cualquier paquete.

- nombre : lodging-service
- Endpoint : “api/v1/lodging”

## Servicio en ejecución



The screenshot shows a Linux desktop environment. In the background, a file manager window displays a directory containing several service folders: admin-package-service, auth-service, health-service, lodging-service, matches-service, and travel-service. In the foreground, a terminal window titled 'administrador@administrador-Aspire-E5-573: ~/Facultad/4to/Cliente-Se...' shows the output of a Java application. The terminal output includes Hibernate SQL commands for creating and altering tables, followed by various log messages from the application, including warnings about JPA configuration and information about the Tomcat web server and the LodgingApplication starting.

```
admin-  
package-  
service  
auth-  
service  
health-  
service  
lodging-  
service  
matches-  
service  
travel-  
service  
  
administrador@administrador-Aspire-E5-573: ~/Facultad/4to/Cliente-Se...  
Hibernate: create table room (id bigint not null auto_increment, number bigint n  
ot null, occupancy integer not null, room_services varchar(255) not null, lodgin  
g bigint not null, primary key (id)) engine=InnoDB  
Hibernate: alter table room add constraint FK3o2e0w141t5o4jf684lmeaeim foreign k  
ey (lodging) references lodging (id)  
2022-05-25 11:33:30.596 INFO 23892 --- [ restartedMain] o.h.e.t.j.p.i.JtaPlatf  
ormInitiator : HHH000490: Using JtaPlatform implementation: [org.hibernate  
.engine.transaction.jta.platform.internal.NoJtaPlatform]  
2022-05-25 11:33:30.602 INFO 23892 --- [ restartedMain] j.LocalContainerEntity  
ManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit '  
default'  
2022-05-25 11:33:30.902 WARN 23892 --- [ restartedMain] JpaBaseConfiguration$J  
paWebConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, d  
atabase queries may be performed during view rendering. Explicitly configure spr  
ing.jpa.open-in-view to disable this warning  
2022-05-25 11:33:31.194 INFO 23892 --- [ restartedMain] o.s.b.d.a.OptionalLive  
ReloadServer : LiveReload server is running on port 35729  
2022-05-25 11:33:31.221 INFO 23892 --- [ restartedMain] o.s.b.w.embedded.tomca  
t.TomcatWebServer : Tomcat started on port(s): 8005 (http) with context path '/  
api/v1'  
2022-05-25 11:33:31.234 INFO 23892 --- [ restartedMain] c.a.service.lodging.Lo  
dgingApplication : Started LodgingApplication in 57.741 seconds (JVM running f  
or 58.17)
```

## Dominio

### Alojamiento

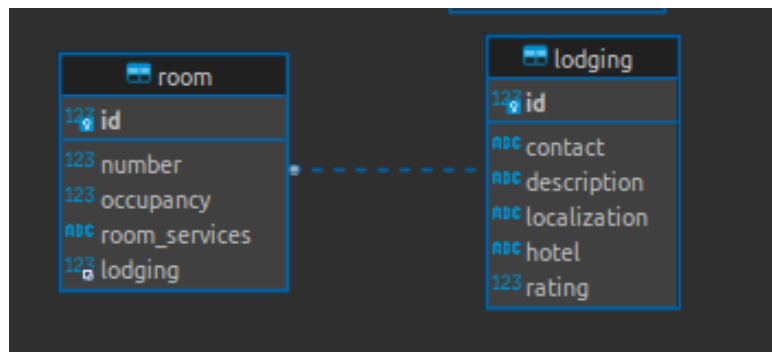
```
public class Lodging {  
    @Id  
    @GeneratedValue(strategy=GenerationType.IDENTITY)  
    private Long id;  
  
    @Column(name = "hotel", nullable = false, updatable = true)  
    private String nameHotel;  
  
    @Column(name = "localization", nullable = false, updatable = true)  
    private String localization;  
  
    @Column(name = "contact", nullable = false, updatable = true)  
    private String contact;  
  
    @NotNull  
    @Column(name = "rating", nullable = false, updatable = true)  
    private Double rating = 0.0;  
  
    @Column(name = "description", nullable = false, updatable = true)  
    private String description;  
  
    @OneToMany(mappedBy = "lodging", cascade = {CascadeType.PERSIST, CascadeType.REMOVE, Ca  
    private List<Room> rooms = new ArrayList<Room>();  
}
```

### Habitaciones

```
public class Room {  
    @Id  
    @GeneratedValue(strategy=GenerationType.IDENTITY)  
    private Long id;  
  
    @NotNull  
    @Column(name = "number", nullable = false, updatable = true)  
    private Long numberRoom;  
  
    @NotNull  
    @Column(name = "occupancy", nullable = false, updatable = true)  
    private int occupancy;  
  
    @OneToOne(cascade=CascadeType.ALL)  
    @JoinColumn(name="lodging", nullable = false, updatable = true)  
    private Lodging lodging;  
  
    @Column(name = "roomServices", nullable = false, updatable = true)  
    private String service;  
}
```



## Persistencia en Base de datos

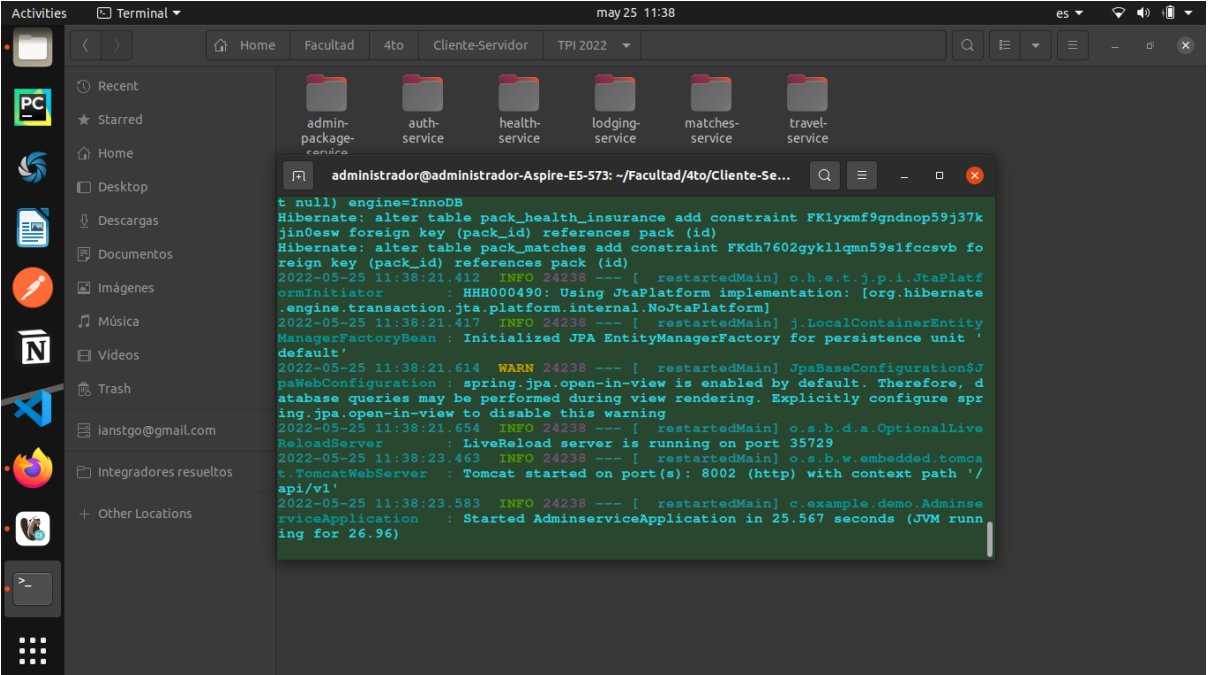


# Servicio de administración de paquetes

Este servicio está realizado con el fin de brindar la posibilidad de crear paquetes, para esto es necesario tener en ejecución los demás servicios y consultarlos para añadirlos al paquete.

- Nombre : package-admin-service
- Endpoint : “api/v1/admin/package”

## Servicio en ejecución



```
Activities Terminal may 25 11:38 es
Home Facultad 4to Cliente-Servidor TPI 2022
admin-package-auth-service health-service lodging-service matches-service travel-service

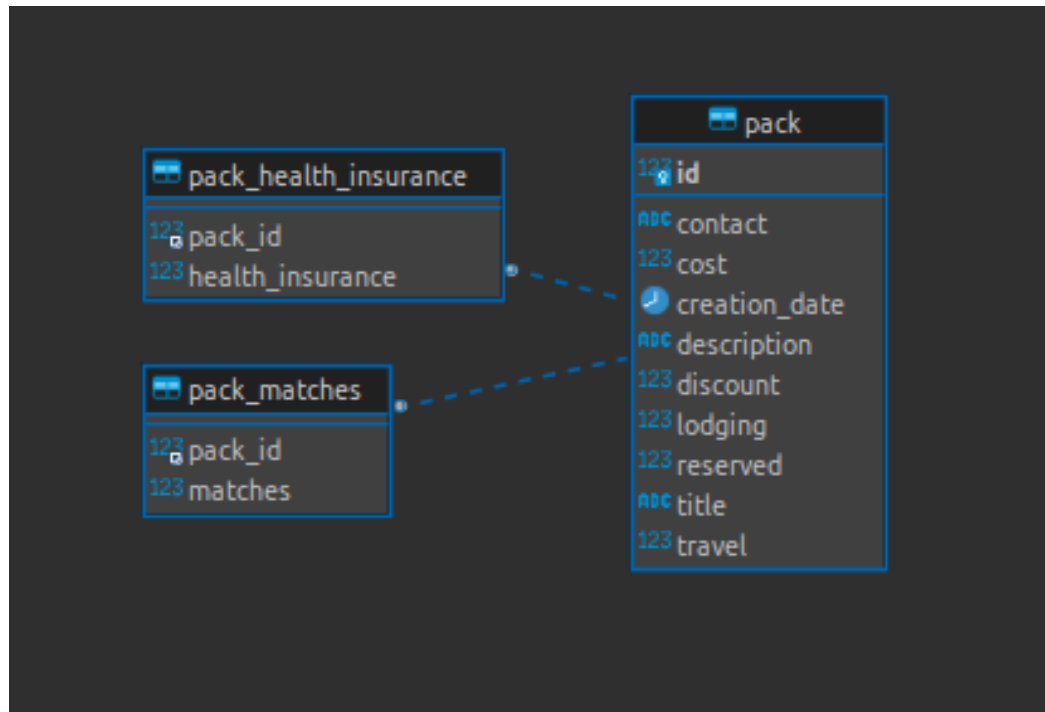
administrador@administrador-Aspire-E5-573: ~/Facultad/4to/Cliente-Se...
t null) engine=InnoDB
Hibernate: alter table pack_health_insurance add constraint FK1yxmf9gndnop59j37k
jin0esw foreign key (pack_id) references pack (id)
Hibernate: alter table pack_matches add constraint FKdh7602gykllqmn59sifccsvb fo
reign key (pack_id) references pack (id)
2022-05-25 11:38:21.412 INFO 24238 --- [ restartedMain] o.h.e.t.j.p.i.JtaPlatf
ormInitiator : HHH000490: Using JtaPlatform implementation: [org.hibernate
.engine.transaction.jta.platform.internal.NoJtaPlatform]
2022-05-25 11:38:21.417 INFO 24238 --- [ restartedMain] j.LocalContainerEntity
ManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit '
default'
2022-05-25 11:38:21.614 WARN 24238 --- [ restartedMain] JpaBaseConfiguration$J
paWebConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, d
atabase queries may be performed during view rendering. Explicitly configure spr
ing.jpa.open-in-view to disable this warning
2022-05-25 11:38:21.654 INFO 24238 --- [ restartedMain] o.s.b.d.a.OptionalLive
ReloadServer : LiveReload server is running on port 35729
2022-05-25 11:38:23.463 INFO 24238 --- [ restartedMain] o.s.b.w.embedded.tomcat
TomcatWebServer : Tomcat started on port(s): 8002 (http) with context path '/'
api/v1'
2022-05-25 11:38:23.583 INFO 24238 --- [ restartedMain] c.example.demo.Adminse
rvicApplication : Started AdminserviceApplication in 25.567 seconds (JVM runn
ing for 26.96)
```

Dominio

Paquete

```
public class Pack {  
    @Id  
    @GeneratedValue(strategy=GenerationType.IDENTITY)  
    private Long id;  
  
    @Column(name = "title", nullable = false, updatable = true)  
    private String title;  
  
    @Column(name = "creationDate", nullable = false, updatable = true)  
    private LocalDate creationDate = LocalDate.now();  
  
    @Column(name = "cost", nullable = false, updatable = true)  
    private Double cost;  
  
    @Column(name = "description", nullable = false, updatable = true)  
    private String description;  
  
    @Column(name = "contact", nullable = false, updatable = true)  
    private String contact;  
  
    @Column(name = "discount", nullable = true, updatable = true)  
    private int discount;  
  
    @Column(name = "reserved", nullable = false, updatable = true)  
    private Boolean reserved;  
  
    /**RELATIONSHIP */  
    @Column(name = "healthInsurance", nullable = false, updatable = true)  
    @ElementCollection(targetClass=Long.class)  
    private List<Long> healthInsurance = new ArrayList<>();  
  
    @Column(name = "lodging", nullable = false, updatable = true)  
    private Long lodging;  
  
    @Column(name = "matches", nullable = false, updatable = true)  
    @ElementCollection(targetClass=Long.class)  
    private List<Long> matches = new ArrayList<>();  
  
    @Column(name = "travel", nullable = false, updatable = true)  
    private Long travel;  
}
```

## Persistencia en Base de datos



# Formas de ejecución

Para la ejecución de la aplicación deben de ejecutarse primeramente Eureka server, luego todos los servicios que se necesiten, en caso de querer ejecutarse el servicio de administración de paquetes, deben ejecutarse todos los servicios. Por ultimo debe ejecutarse Gateway cloud

## ¿Qué es Eureka server?

### ¿Qué es?

Eureka es un servidor para el registro y localización de microservicios, balanceo de carga y tolerancia a fallos. **La función de Eureka es registrar las diferentes instancias de microservicios existentes, su localización, estado, metadatos...**

### ¿Cómo funciona?

Cada microservicio, durante su arranque, se comunicará con el servidor Eureka para notificar que está disponible, dónde está situado, sus metadatos... De esta forma **Eureka mantendrá en su registro la información de todos los microservicios del ecosistema.**

El nuevo microservicio continuará notificando a Eureka su estado cada 30 segundos, lo que denominan 'heartbeats'. Si después de tres periodos Eureka no recibe notificación de dicho microservicio, lo eliminará del registro. De la misma forma una vez vueltos a recibir tres notificaciones considerará el servicio disponible de nuevo.

Cada cliente de Eureka podrá también recuperar el registro para localizar otros microservicios con los que necesite comunicarse. Dicha información de registro se cachea en el cliente.

Eureka **se puede configurar para funcionar en modo cluster donde varias instancias "peer" intercambiarán su información.** Esto, junto al cacheo de la información de registro en los clientes da a Eureka una alta tolerancia a fallos.

The screenshot shows the Eureka UI in a browser window. The address bar shows 'localhost:8761'. The page has a top navigation bar with 'Renews (last min)' and '0'. Below this, there's a section 'DS Replicas' with a search bar containing 'localhost'. The main section is 'Instances currently registered with Eureka', which contains a table with the following data:

Application	AMIs	Availability Zones	Status
PACKAGE-ADMIN-SERVICE	n/a (1)	(1)	UP (1) - <a href="http://192.168.100.8:package-admin-service:8002">192.168.100.8:package-admin-service:8002</a>

Below the table is a 'General Info' section with a table of system metrics:

Name	Value
total-avail-memory	155mb
num-of-cpus	4
current-memory-usage	82mb (52%)
server-uptime	00:01
registered-replicas	<a href="http://localhost:8761/eureka/">http://localhost:8761/eureka/</a>
unavailable-replicas	<a href="http://localhost:8761/eureka/">http://localhost:8761/eureka/</a> ,

The bottom of the page shows the URL 'localhost:8002/api/v1/actuator/info'.

*En este ejemplo vemos como levantamos el servidor de Eureka y luego levantamos el servicio de Package-Admin-Service, el cual fue registrado por Eureka. Ahora el servicio de administración de paquetes ya es reconocido a partir de un nombre y a partir de esta interfaz obtenemos su estado. A partir de registrar todos estos servicios con eureka, decimos que estos son clientes del servicio Eureka.*

## ¿Qué es Gateway cloud?

**Spring Cloud Gateway** se encargan de proporcionar un punto de entrada a nuestro **ecosistema de microservicios**, proporcionando capacidades de enrutamiento dinámico, seguridad y monitorización de las llamadas que se realicen.

## ¿Qué beneficios me aporta?

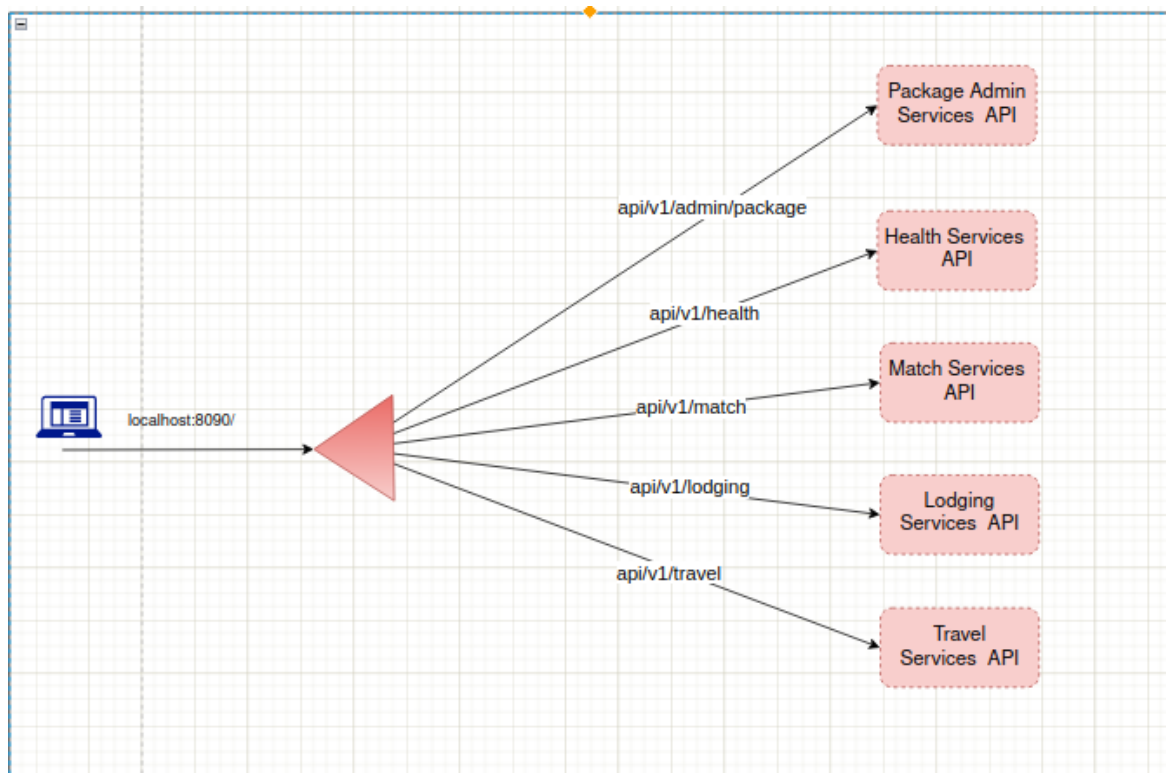
Entre tantas cosas, se dice que es la llave de entrada a las distintas APIs. Como vimos cada api tiene su puerto al cual accedemos cuando el servicio se está ejecutando para utilizarlo. Con Gateway podemos hacer uso de un solo puerto y partir de indicar a que ruta queremos acceder, este servicio nos redirecciona de un servicio a otro haciendo creer al cliente que realmente esta consumiendo una sola aplicación. Para esto es necesario, primeramente registrar mi aplicaciones con eureka server, luego configurar las rutas con el nombre o identificador de cada aplicacion.

```

spring:
  cloud:
    gateway:
      routes:
        - id: package-admin-service
          uri: lb://package-admin-service
          predicates:
            - Path=/api/v1/admin/package/**
          filters:
            - StripPrefix=4
        - id: health-insurance-service
          uri: lb://health-insurance-service
          predicates:
            - Path=/api/v1/health/**
          filters:
            - StripPrefix=3
        - id: lodging-service
          uri: lb://lodging-service
          predicates:
            - Path=/api/v1/lodging/**
          filters:
            - StripPrefix=3
        - id: match-service
          uri: lb://match-service
          predicates:
            - Path=/api/v1/match/**
          filters:
            - StripPrefix=3
        - id: travel-service
          uri: lb://travel-service
          predicates:
            - Path=/api/v1/travel/**
          filters:
            - StripPrefix=3

```

*Configuración de las rutas de las apis enlazadas con cada servicio.*



## Conclusión

Gracias a este trabajo logramos interiorizarnos en nuestra nueva arquitectura de desarrollo, pudiendo comprender sus ventajas y desventajas, aprendiendo a implementar nuevas tecnologías de desarrollo. Todavía quedan pendientes algunos servicios y llevar este proyecto backend al siguiente nivel que es implementar seguridad en las aplicaciones, persistencia en la nube y virtualizar a partir de docker las aplicaciones desarrolladas.



## Fuentes:

- <https://www.pragma.com.co/academia/lecciones/como-configurar-un-proyecto-con-spring-cloud-gateway>
- <https://www.paradigmadigital.com/dev/quien-es-quien-en-la-arquitectura-de-microservicios-spring-cloud-12/>
- <https://aws.amazon.com/es/microservices/>
- <https://www.javiergarzas.com/2015/06/microservicios.html>