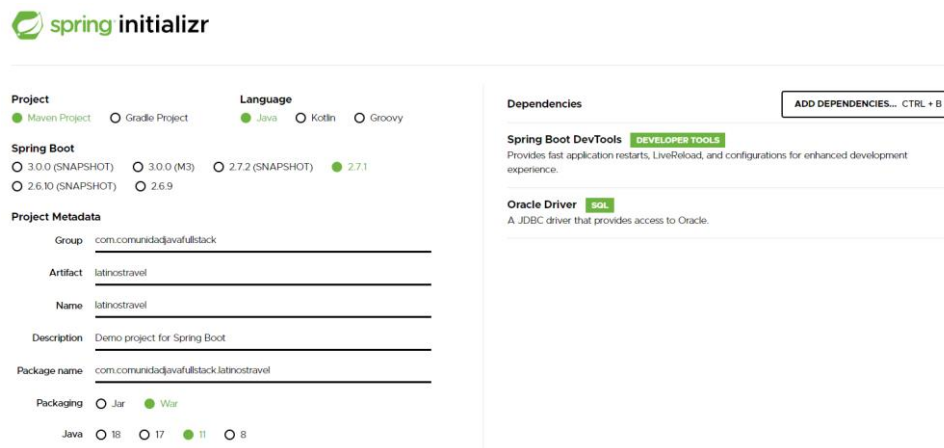


## WORKSHOP: IMPLEMENTACION DE UNA API REST CON JDBC

### Pasos

1. Crear el proyecto inicial desde el sitio web: <https://start.spring.io/>
2. Configurar el proyecto con la siguiente información:



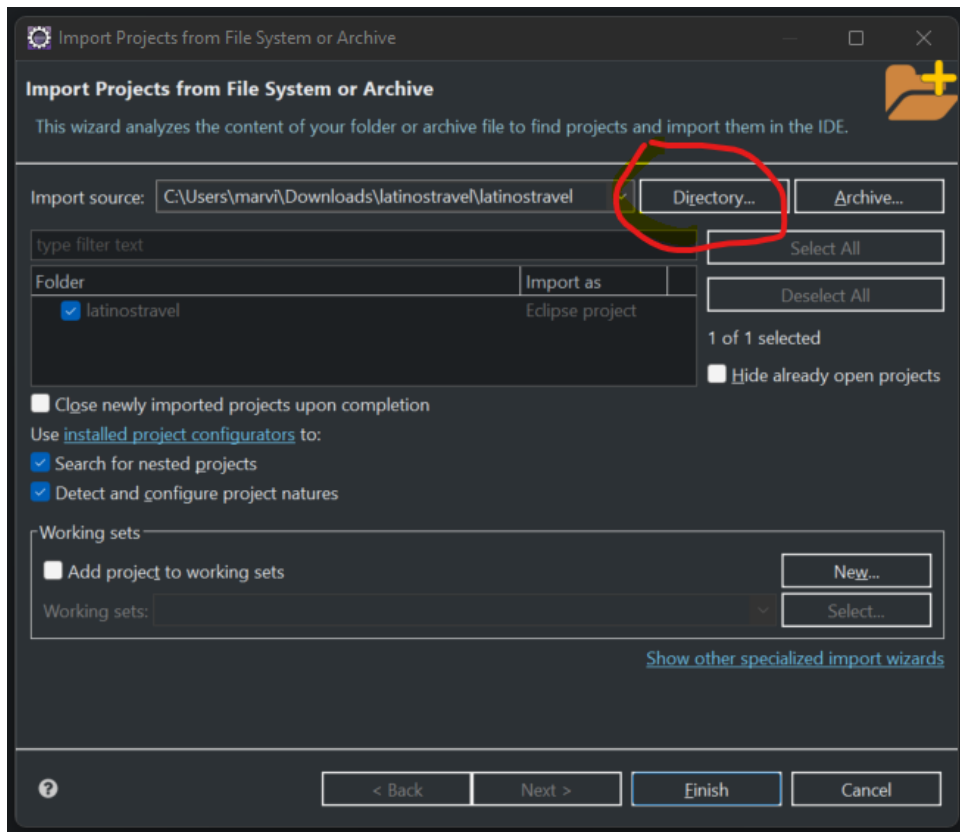
The screenshot shows the Spring Initializr web form with the following configuration:

- Project:** ☒ Maven Project, ☐ Gradle Project
- Language:** ☒ Java, ☐ Kotlin, ☐ Groovy
- Spring Boot:** ☐ 3.0.0 (SNAPSHOT), ☐ 3.0.0 (M3), ☐ 2.7.2 (SNAPSHOT), ☒ 2.7.1, ☐ 2.6.10 (SNAPSHOT), ☐ 2.6.9
- Project Metadata:**
  - Group:** com.comunidadjavafullstack
  - Artifact:** latinostravel
  - Name:** latinostravel
  - Description:** Demo project for Spring Boot
  - Package name:** com.comunidadjavafullstack.latinostravel
- Packaging:** ☐ Jar, ☒ War
- Java:** ☐ 18, ☐ 17, ☒ 11, ☐ 8
- Dependencies:** 
  - Spring Boot DevTools:** ☒ **DEVELOPER TOOLS**  
Provides fast application restarts, LiveReload, and configurations for enhanced development experience.
  - Oracle Driver:** ☒ **SQL**  
A JDBC driver that provides access to Oracle.

3. De clic en el botón Generate.
4. Ir a Descargas y descomprimir el archivo .zip



5. Abrir el IDE Eclipse
6. Ir a File --> Open Projects from File System



7. Desde la ventana modal de clic en el botón Directory y seleccione la carpeta descomprimida.
8. De clic en Finalizar.

A continuación, Maven intentará realizar la descarga y actualización de las dependencias.

9. Abrir el archivo pom.xml
10. Asegúrese de que se encuentren las siguientes dependencias, en caso de que no existan agregarlas:

```
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-jdbc</artifactId>
</dependency>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-web</artifactId>
</dependency>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-web-services</artifactId>
</dependency>
<dependency>
  <groupId>com.oracle.jdbc</groupId>
  <artifactId>ojdbc8</artifactId>
  <version>12.2.0.1</version>
</dependency>
<dependency>
<groupId>org.projectlombok</groupId>
<artifactId>lombok</artifactId>
<optional>true</optional>
</dependency>
<dependency>
<groupId>org.projectlombok</groupId>
<artifactId>lombok</artifactId>
```

```

<scope>provided</scope>
</dependency>
<dependency>
<groupId>io.springfox</groupId>
<artifactId>springfox-swagger2</artifactId>
<version>2.9.2</version>
</dependency>
<dependency>
<groupId>io.springfox</groupId>
<artifactId>springfox-swagger-ui</artifactId>
<version>2.9.2</version>
</dependency>

```

11. Abrir el archivo `application.properties` y añadir la siguiente información:

```

server.port=8001
server.servlet.context-path=/wsLatinosTravel

# API Configuration
api.version=v1
api.base=ESQUEMA

spring.datasource.url=jdbc:oracle:thin:@localhost:1521/xe
spring.datasource.driver-class-name=oracle.jdbc.driver.OracleDriver
spring.datasource.username=USUARIO
spring.datasource.password=clave

# Connection Pool Configuration
spring.datasource.tomcat.max-wait=20000
spring.datasource.tomcat.max-active=50
spring.datasource.tomcat.max-idle=20
spring.datasource.tomcat.min-idle=5

spring.mvc.pathmatch.matching-strategy = ANT_PATH_MATCHER

```

12. Actualice la información del archivo `.properties` según la configuración de su instancia de Oracle Server en el equipo local.
13. Agregue un nuevo paquete con el nombre `api` dentro del paquete: `com.comunidadjavafullstack.latinotravel`
14. Añada un archivo de tipo Clase con el nombre `ResponseError` y copie el siguiente contenido:

```

import java.util.Map;

public class ResponseError extends Exception {
    private static final long serialVersionUID = 1L;
    private String messageError;
    private String codigoError;
    private Map<String, Object> mapa;
    private Object lista;

    public ResponseError()
    {
        super();
    }
    public ResponseError(String messageError) {
        super();
        this.messageError = messageError;
    }

    public ResponseError(String messageError, String codigoError) {
        super();
    }

```

```

this.messageError = messageError;
this.codigoError = codigoError;
}

public ResponseError(String messageError, String codigoError, Map<String, Object> mapa) {
    super();
    this.messageError = messageError;
    this.codigoError = codigoError;
    this.mapa = mapa;
}

public ResponseError(String messageError, String codigoError, Object lista) {
    super();
    this.messageError = messageError;
    this.codigoError = codigoError;
    this.lista = lista;
}

public String getMessageError() {
    return messageError;
}

public void setMessageError(String messageError) {
    this.messageError = messageError;
}

public String getCodigoError() {
    return codigoError;
}

public void setCodigoError(String codigoError) {
    this.codigoError = codigoError;
}

public Object getLista() {
    return lista;
}

public void setLista(Object lista) {
    this.lista = lista;
}

public Map<String, Object> getMapa() {
    return mapa;
}

public void setMapa(Map<String, Object> mapa) {
    this.mapa = mapa;
}
}

```

15. Añada un archivo de tipo Clase con el nombre ResponseMessage

```

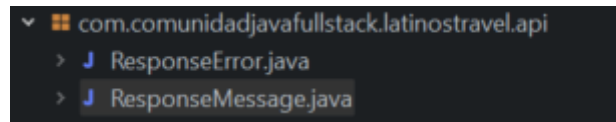
@Data
public class ResponseMessage {

    private int status;
    private String error;
    private String message;
    private Object data;

}

```

El nuevo paquete debe visualizarse de la siguiente manera.



16. Agregue un nuevo paquete con el nombre util dentro del paquete:

com.comunidadjavafullstack.latinotravel

17. Añada un archivo de tipo Clase con el nombre OracleJdbcCall y copie el siguiente contenido:

```
import java.util.HashMap;
import java.util.Map;

import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.jdbc.core.RowMapper;
import org.springframework.jdbc.core.SqlOutParameter;
import org.springframework.jdbc.core.SqlParameter;
import org.springframework.jdbc.core.simple.SimpleJdbcCall;

import oracle.jdbc.OracleTypes;

public class OracleJdbcCall<T> {
    private JdbcTemplate jdbcTemplate;
    private SimpleJdbcCall simpleJdbcCall;

    private String schemaName;
    private String catalogName;

    private String[] declareParametersIn;
    private Map<String, Object> mapParametersInValue;

    public OracleJdbcCall<T> crearInstancia(){
        var jdbcCall = new OracleJdbcCall<T>(new JdbcTemplate(JdbcTemplate.getDataSource()), schemaName,
        catalogName);
        return jdbcCall;
    }

    public OracleJdbcCall(JdbcTemplate jdbcTemplate, String schemaName, String catalogName) {
        this.jdbcTemplate = jdbcTemplate;
        if (schemaName != null && !schemaName.trim().isEmpty() && catalogName != null
        && !catalogName.trim().isEmpty()) {
            this.schemaName = schemaName;
            this.catalogName = catalogName;
            this.mapParametersInValue = new HashMap<>();
        } else {
            throw new NullPointerException("Los parámetros schemaName y catalogName no pueden ser null o vacío.");
        }
    }

    public JdbcTemplate getJdbcTemplate() {
        return jdbcTemplate;
    }

    public OracleJdbcCall<T> newCall(String procedureName) {
        if (procedureName != null && !procedureName.trim().isEmpty()) {
            simpleJdbcCall = new
            SimpleJdbcCall(jdbcTemplate).withSchemaName(schemaName).withCatalogName(catalogName)
```

```

.withProcedureName(procedureName);
} else {
throw new NullPointerException("El parámetro procedureName no puede ser null o vacío.");
}
return this;
}

public OracleJdbcCall<T> addDeclareParametersIn(int[] paramOracleTypes, String... declareParametersIn) {
if (simpleJdbcCall != null) {
if (paramOracleTypes != null && declareParametersIn != null) {
if (paramOracleTypes.length > 0 && declareParametersIn.length > 0) {
if (paramOracleTypes.length == declareParametersIn.length) {
this.declareParametersIn = declareParametersIn;
for (int i = 0; i < declareParametersIn.length; i++) {
if (declareParametersIn[i] != null && !declareParametersIn[i].trim().isEmpty()) {
simpleJdbcCall.addDeclaredParameter(
new SqlParameter(declareParametersIn[i], paramOracleTypes[i]));
} else {
throw new NullPointerException(
"El parámetro declarado del Procedimiento no puede ser null o estar vacío.");
}
}
} else {
throw new ArrayIndexOutOfBoundsException(
"La cantidad de argumentos en la variable paramOracleTypes difiere de la cantidad de "
+ "argumentos definidos en la variable declareParametersIn, deben tener la misma cantidad "
+ "de argumentos.");
}
} else {
throw new NullPointerException(
"Los parámetros paramOracleTypes y declareParametersIn deben contener al menos un valor ambos.");
}
} else {
throw new NullPointerException(
"Los parámetros paramOracleTypes y declareParametersIn no pueden ser null.");
}
} else {
throw new NullPointerException(
"Asegurese de invocar al metodo newCall antes de realizar cualquier operación, para poder crear una "
+ "instancia del Objeto SimpleJdbcCall.");
}
return this;
}
}

public OracleJdbcCall<T> addParametersInValue(Object... parametersInValue) {
if (parametersInValue != null) {
if (parametersInValue.length > 0) {
if (declareParametersIn != null && declareParametersIn.length > 0) {
if (declareParametersIn.length == parametersInValue.length) {
for (int i = 0; i < parametersInValue.length; i++) {
mapParametersInValue.put(declareParametersIn[i], parametersInValue[i]);
}
}
} else {
throw new ArrayIndexOutOfBoundsException(
"La cantidad de argumentos en la variable declareParametersIn difiere de la cantidad de "
+ "argumentos definidos en la variable parametersInValue, deben tener la misma cantida de argumentos.");
}
}
}
}
}

```

```

} else {
throw new NullPointerException(
"La variable declareParametersIn no puede ser null o estar vacío, antes de establecer los valores a "
+ "los parámetros hay que declarar los parámetros.");
}
} else {
throw new NullPointerException("La variable parametersInValue no puede estar vacío.");
}
} else {
throw new NullPointerException("La variable parametersInValue no puede ser null.");
}
return this;
}

```

```

public OracleJdbcCall<T> addDeclareParametersOut(int[] paramOracleTypes, RowMapper<T> rowMapper,
String... declareParametersOut) {
if (simpleJdbcCall != null) {
if (paramOracleTypes != null && declareParametersOut != null) {
if (paramOracleTypes.length > 0 && declareParametersOut.length > 0) {
if (paramOracleTypes.length == declareParametersOut.length) {
for (int i = 0; i < declareParametersOut.length; i++) {
if (declareParametersOut[i] != null && !declareParametersOut[i].trim().isEmpty()) {
if (paramOracleTypes[i] == OracleTypes.CURSOR) {
if (rowMapper != null) {
simpleJdbcCall.addDeclaredParameter(
new SqlOutParameter(declareParametersOut[i], paramOracleTypes[i], rowMapper));
} else {
throw new NullPointerException("La implementación de RowMapper no puede ser null.");
}
} else {
simpleJdbcCall.addDeclaredParameter(
new SqlOutParameter(declareParametersOut[i], paramOracleTypes[i]));
}
} else {
throw new NullPointerException(
"El parámetro declarado del Procedimiento no puede ser null o estar vacío.");
}
}
} else {
throw new ArrayIndexOutOfBoundsException(
"La cantidad de argumentos en la variable paramOracleTypes difiere de la cantidad de "
+ "argumentos definidos en la variable declareParametersOut, deben tener la misma cantida "
+ "de argumentos.");
}
} else {
throw new NullPointerException("Los parámetros paramOracleTypes y declareParametersOut deben contener
al menos un valor ambos.");
}
} else {
throw new NullPointerException("Los parámetros paramOracleTypes y declareParametersOut no pueden ser
null.");
}
} else {
throw new NullPointerException(
"No se ha invocado al metodo newCall para crear" + " una instancia del Objeto SimpleJdbcCall.");
}
}

```

```

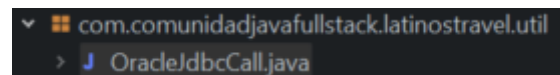
return this;
}

public Map<String, Object> execute(Map<String, Object> mapParametersInValue) {
return simpleJdbcCall.execute(mapParametersInValue);
}

public Map<String, Object> execute() {
return simpleJdbcCall.execute(mapParametersInValue);
}
}

```

El nuevo paquete debe visualizarse de la siguiente manera.



18. Añada un archivo de tipo Clase con el nombre SwaggerConfig en el paquete com.comunidadjavafullstack.latinotravel y copie el siguiente contenido:

```

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

import springfox.documentation.builders.PathSelectors;
import springfox.documentation.builders.RequestHandlerSelectors;
import springfox.documentation.spi.DocumentationType;
import springfox.documentation.spring.web.plugins.Docket;
import springfox.documentation.swagger2.annotations.EnableSwagger2;

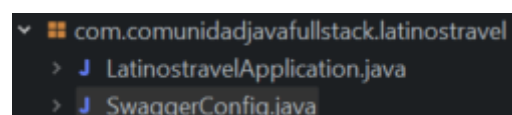
@Configuration
@EnableSwagger2
public class SwaggerConfig {

    @Bean
    public Docket api() {
return new Docket(DocumentationType.SWAGGER_2)
.select()
.apis(RequestHandlerSelectors.any())
.paths(PathSelectors.any())
.build();
}

}

```

El nuevo archivo debe visualizarse de la siguiente manera.



19. Agregue un nuevo paquete con el nombre domain dentro del paquete: com.comunidadjavafullstack.latinotravel
20. Añada un archivo de tipo Clase con el nombre Pasajero y defina la estructura de la clase Pasajero. El contenido debería observarse como en el siguiente código.



```

import java.time.LocalDate;

import com.fasterxml.jackson.annotation.JsonInclude;
import com.fasterxml.jackson.annotation.JsonInclude.Include;

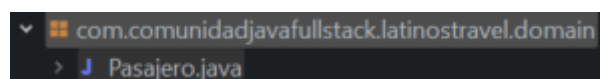
import io.swagger.annotations.ApiModelProperty;
import lombok.Data;

@Data
@JsonInclude(value = Include.NON_NULL)
public class Pasajero {

    @ApiModelProperty(position = 0)
    private Long id;
    @ApiModelProperty(position = 1)
    private String primerNombre;
    @ApiModelProperty(position = 2)
    private String segundoNombre;
    @ApiModelProperty(position = 3)
    private String primerApellido;
    @ApiModelProperty(position = 4)
    private String segundoApellido;
    @ApiModelProperty(position = 5)
    private String tipoDocumento;
    @ApiModelProperty(position = 6)
    private String numeroDocumento;
    @ApiModelProperty(position = 7)
    private LocalDate fechaNacimiento;
    @ApiModelProperty(position = 8)
    private String pais;
    @ApiModelProperty(position = 9)
    private String telefono;
    @ApiModelProperty(position = 10)
    private String email;
    @ApiModelProperty(position = 11)
    private String clave;
    @ApiModelProperty(position = 12)
    private int activo;
}

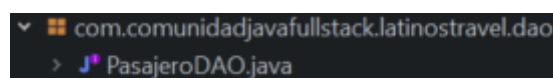
```

El nuevo paquete debe visualizarse de la siguiente manera:

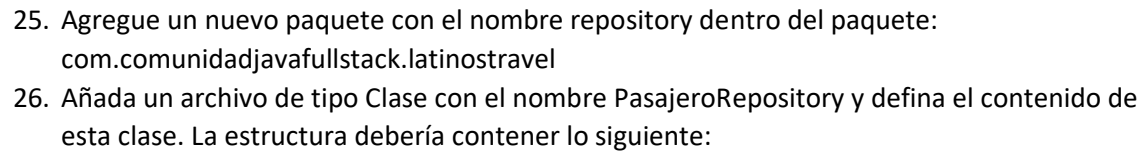


21. Agregue un nuevo paquete con el nombre dao dentro del paquete:  
com.comunidadjavafullstack.latinotravel
22. Añada un archivo de tipo Interfaz con el nombre PasajeroDAO y realice una copia del contenido del recurso con nombre: Recurso 1.txt

El nuevo paquete debe visualizarse de la siguiente manera:



- El nuevo paquete debe visualizarse de la siguiente manera:



```
@Override  
public Map<String, Object> guardarPasajero(Pasajero pasajero) {  
  
    return oracleJdbcCall.crearInstancia().newCall(CRUD_PASAJERO)  
        .addDeclareParametersIn(new int[] {  
            OracleTypes.NUMBER,  
            OracleTypes.VARCHAR,  
            OracleTypes.VARCHAR,  
            OracleTypes.VARCHAR,  
            OracleTypes.VARCHAR,  
            OracleTypes.VARCHAR,  
            OracleTypes.VARCHAR,  
            OracleTypes.DATE,  
            OracleTypes.VARCHAR,  
            OracleTypes.VARCHAR,  
            OracleTypes.VARCHAR,  
            OracleTypes.VARCHAR,  
            OracleTypes.VARCHAR  
        })
```

```

},
pPasajeroId,
pPrimerNombre,
                pSegundoNombre,
                pPrimerApellido,
                pSegundoApellido,
                pTipoDocumento,
                pNumDocumento,
                pFechaNacimiento,
                pPais,
                pTelefono,
                pEmail,
                pClave,
pTipoOperacion)
.addParametersInValue(
pasajero.getId(),
pasajero.getPrimerNombre(),
pasajero.getSegundoNombre(),
pasajero.getPrimerApellido(),
pasajero.getSegundoApellido(),
pasajero.getTipoDocumento(),
pasajero.getNumeroDocumento(),
pasajero.getFechaNacimiento(),
pasajero.getPais(),
pasajero.getTelefono(),
pasajero.getEmail(),
pasajero.getClave(),
"I")
.addDeclareParametersOut(new int[] { OracleTypes.CURSOR, OracleTypes.VARCHAR,
OracleTypes.VARCHAR}, new PasajeroMapper(), "pRegistro", "pResultado", "pMsgError")
.execute();

}

```

```

@Override
public Map<String, Object> actualizarPasajero(Pasajero pasajero) {
return oracleJdbcCall.crearInstancia().newCall(CRUD_PASAJERO)
.addDeclareParametersIn(new int[] {
OracleTypes.NUMBER,
OracleTypes.VARCHAR,
OracleTypes.VARCHAR,
OracleTypes.VARCHAR,
OracleTypes.VARCHAR,
OracleTypes.VARCHAR,
OracleTypes.VARCHAR,
OracleTypes.VARCHAR,
OracleTypes.VARCHAR,
OracleTypes.DATE,
OracleTypes.VARCHAR,
OracleTypes.VARCHAR,
OracleTypes.VARCHAR,
OracleTypes.VARCHAR,
OracleTypes.VARCHAR,
OracleTypes.VARCHAR
},
pPasajeroId,
pPrimerNombre,
                pSegundoNombre,
                pPrimerApellido,
                pSegundoApellido,
                pTipoDocumento,
                pNumDocumento,
                pFechaNacimiento,

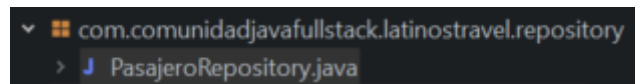
```

```

        pPais,
        pTelefono,
        pEmail,
        pClave,
        pTipoOperacion)
        .addParametersInValue(
        pasajero.getId(),
        pasajero.getPrimerNombre(),
        pasajero.getSegundoNombre(),
        pasajero.getPrimerApellido(),
        pasajero.getSegundoApellido(),
        pasajero.getTipoDocumento(),
        pasajero.getNumeroDocumento(),
        pasajero.getFechaNacimiento(),
        pasajero.getPais(),
        pasajero.getTelefono(),
        pasajero.getEmail(),
        pasajero.getClave(),
        "U")
        .addDeclareParametersOut(new int[] { OracleTypes.CURSOR, OracleTypes.CHAR,
        OracleTypes.CHAR}, new PasajeroMapper(), "pRegistro", "pResultado", "pMsgError")
        .execute();
    }
}

```

El nuevo paquete debe visualizarse de la siguiente manera:



27. Agregue un nuevo paquete con el nombre service dentro del paquete:  
com.comunidadjavafullstack.latinotravel
28. Dentro del nuevo paquete añada un archivo de tipo Interfaz con el nombre PasajeroIService y defina el contenido de esta clase. La estructura debería contener lo siguiente:

```

import java.util.Map;

public interface PasajeroIService {

    Map<String, Object> guardarPasajero(Pasajero objeto);

    Map<String, Object> actualizarPasajero(Pasajero objeto);
}

```

29. Dentro del nuevo paquete añada un archivo de tipo Clase con el nombre PasajeroImplService y defina el contenido de esta clase. La estructura debería contener lo siguiente:

```

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

@Transactional
@Service
public class PasajeroImplService implements PasajeroIService{

    private PasajeroDAO pasajeroDAO;
}

```

```

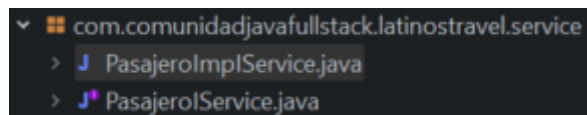
@Autowired
public PasajeroImplService(PasajeroDAO pasajeroDao) {
    this.pasajeroDAO = pasajeroDao;
}

@Transactional
@Override
public Map<String, Object> guardarPasajero(Pasajero objeto) {
    return pasajeroDAO.guardarPasajero(objeto);
}

@Transactional
@Override
public Map<String, Object> actualizarPasajero(Pasajero objeto) {
    return pasajeroDAO.actualizarPasajero(objeto);
}
}

```

El nuevo paquete debe visualizarse de la siguiente manera:



30. Agregue un nuevo paquete con el nombre controller dentro del paquete:  
com.comunidadjavafullstack.latinotravel
31. Dentro del nuevo paquete añada un archivo de tipo Interfaz con el nombre  
PasajeroController y defina el contenido de esta clase. La estructura debería contener lo  
siguiente:

```

import java.util.ArrayList;
import java.util.List;
import java.util.Map;
import java.util.Optional;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import com.comunidadjavafullstack.latinotravel.api.ResponseMessage;
import com.comunidadjavafullstack.latinotravel.domain.Pasajero;
import com.comunidadjavafullstack.latinotravel.service.PasajeroIService;

import io.swagger.annotations.Api;
import io.swagger.annotations.ApiOperation;

@CrossOrigin(origins = "**")
@Api(description = "Este controlador va a manejar la inserción y actualización de registros de pasajeros.")
@RestController
@RequestMapping("${api.version}/${api.base}/gestion")

```

```

public class PasajeroController {

    private PasajeroService pasajeroService;

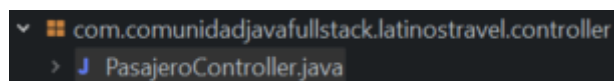
    @Autowired
    public PasajeroController(PasajeroService pasajeroService) {
        this.pasajeroService = pasajeroService;
    }

    @ApiOperation(value = "crearPasajero", notes = "Crear un registro de Pasajero")
    @PostMapping("pasajero")
    public ResponseEntity<ResponseMessage> guardarPasajero(@RequestBody Pasajero objeto){
        Map<String, Object> map = pasajeroService.guardarPasajero(objeto);
        var lista = Optional.<List<?>>ofNullable((ArrayList<?>) map.get("pRegistro"));
        ResponseMessage responseMessage = new ResponseMessage();
        responseMessage.setStatus(HttpStatus.OK.value());
        responseMessage.setError((map.get("pMsgError") != null) ? map.get("pMsgError").toString() :
null);
        responseMessage.setMessage(map.get("pResultado") != null ? map.get("pResultado").toString()
: null);
        responseMessage.setData(lista.orElse(null));
        return new ResponseEntity<>(responseMessage, HttpStatus.OK);
    }

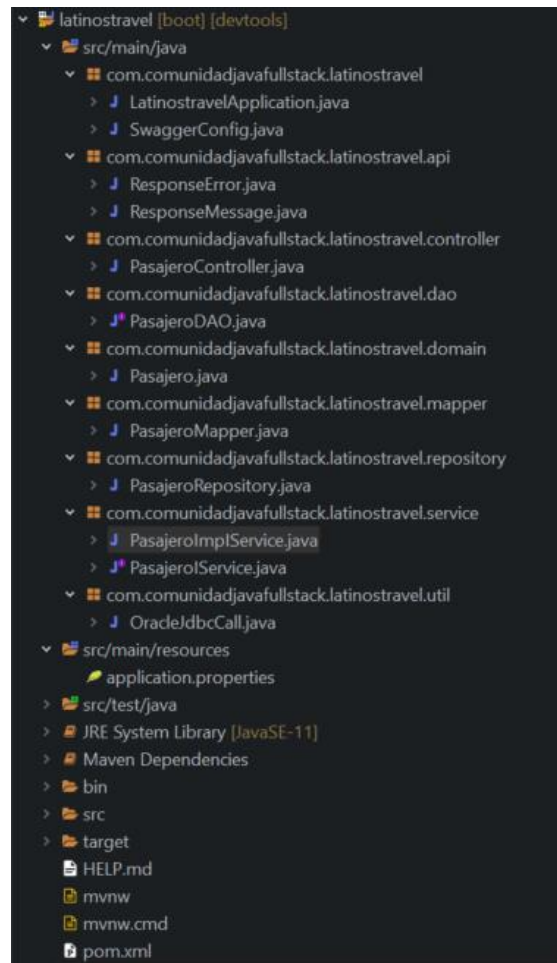
    @ApiOperation(value = "actualizarPasajero", notes = "Actualizar/Pasivar un registro de Pasajero")
    @PutMapping("pasajero")
    public ResponseEntity<ResponseMessage> actualizarMarca(@RequestBody Pasajero objeto) {
        Map<String, Object> map = pasajeroService.actualizarPasajero(objeto);
        var lista = Optional.<List<?>>ofNullable((ArrayList<?>) map.get("pRegistro"));
        ResponseMessage responseMessage = new ResponseMessage();
        responseMessage.setStatus(HttpStatus.OK.value());
        responseMessage.setError((map.get("pMsgError") != null) ? map.get("pMsgError").toString() :
null);
        responseMessage.setMessage(map.get("pResultado") != null ? map.get("pResultado").toString()
: null);
        responseMessage.setData(lista.orElse(null));
        return new ResponseEntity<>(responseMessage, HttpStatus.OK);
    }
}

```

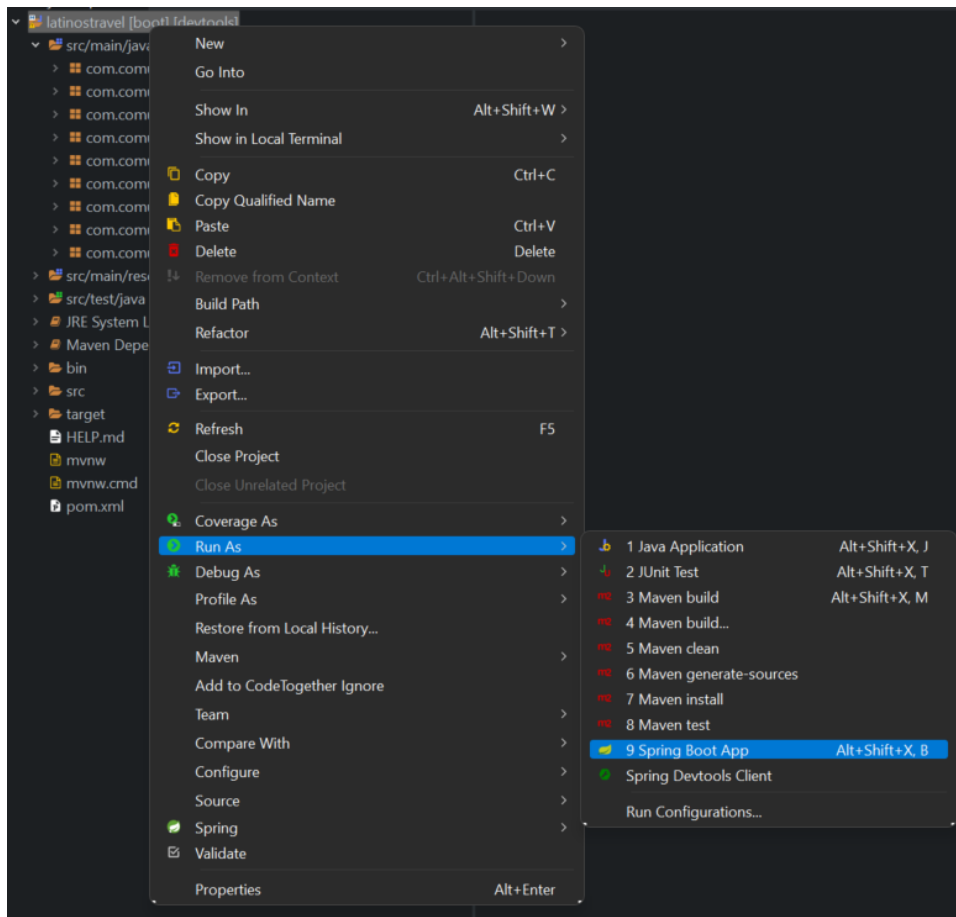
El nuevo paquete debe visualizarse de la siguiente manera:



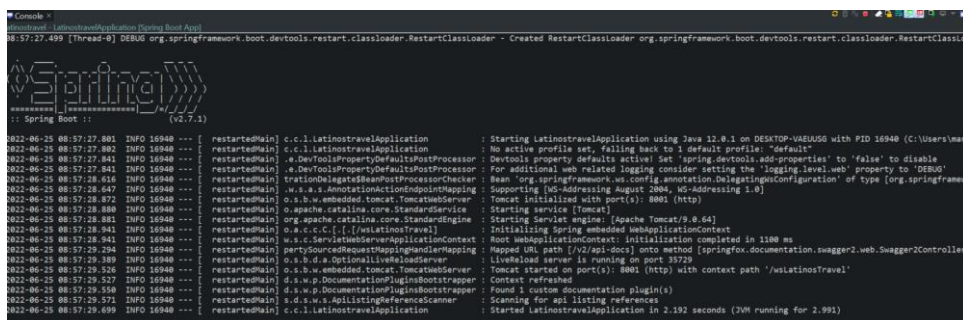
Finalmente el proyecto de la API debería visualizar la siguiente estructura:



32. Intente ejecutar la aplicación de Spring Boot. Siga los siguientes pasos: Desde la raíz principal del proyecto de clic derecho --> Run As --> Spring Boot App

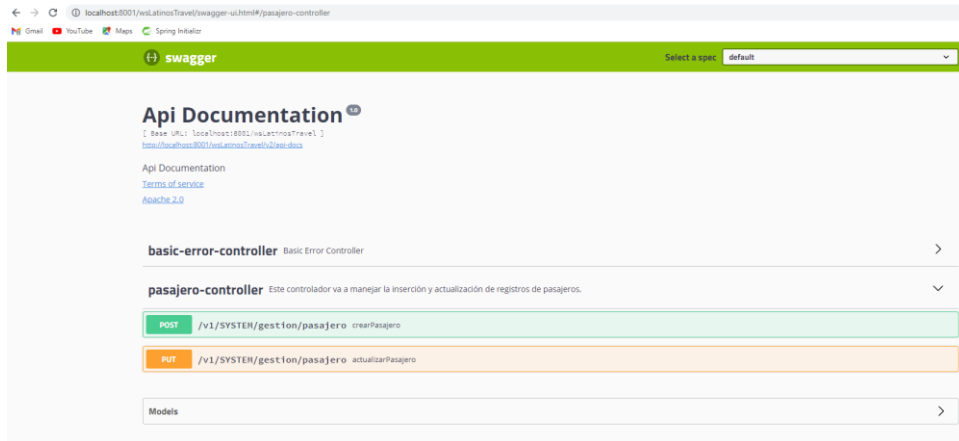


33. Si la aplicación no presentó ningún error de compilación deberá mostrar lo siguiente en la consola:

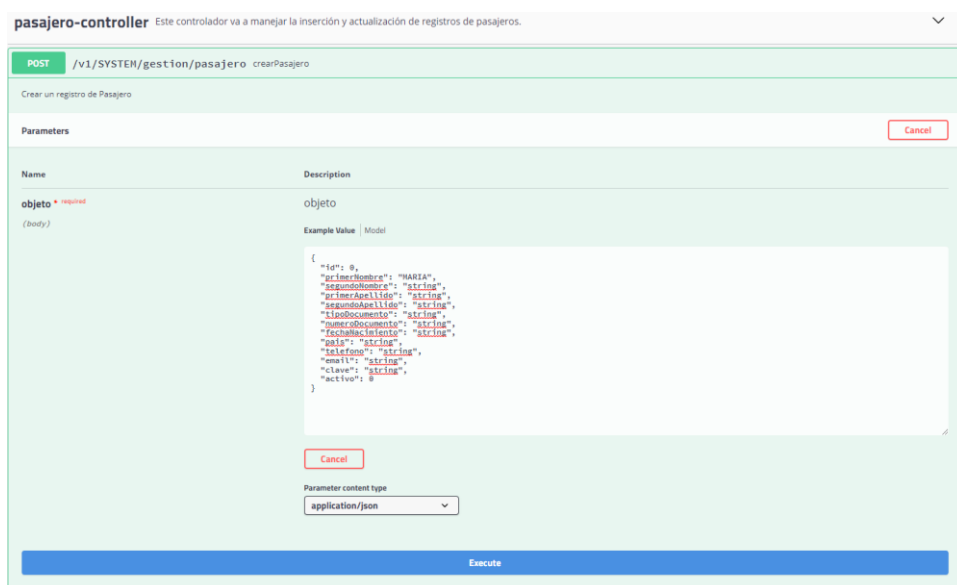


34. En un navegador web escriba la url <http://localhost:8001/wsLatinosTravel/swagger-ui.html> y realice pruebas sobre los servicios creados.





35. De clic sobre uno de los servicios y cuando se abra el panel de clic en el boton Try it out



36. Edite la información de los parametros.

37. De clic en el botón Execute.

38. Realice los pasos 35 y 36 para realizar pruebas en los otros servicios que contenga su API.