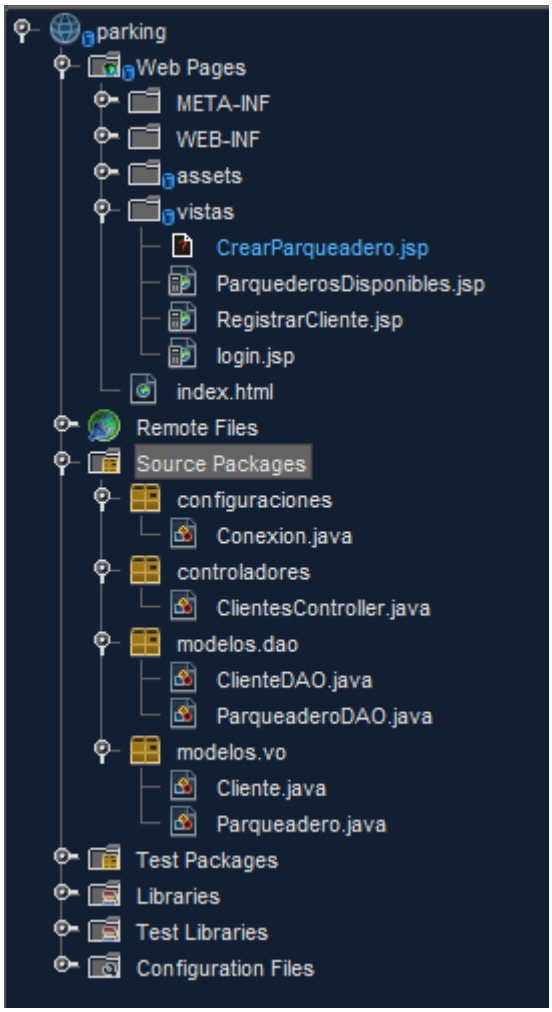
Sprint 2

1. Presentación MVC
   Estructura de archivos requerida para el modelo MVC implementado en este caso



a. Implementación de la lógica de negocio

Funcion borrarParqueadero con la cual se borraran todos los parqueaderos existentes para posteriormente crear uno nuevo con cantidades de puestos diferentes y sin clientes asociados.

```
public boolean borrarParqueadero(){
    PreparedStatement ps;
        try {
            ps = conexion.prepareStatement("DELETE FROM parqueadero");
            ps.execute();
            return true;
        } catch (SQLException e) {
            System.out.println(e.toString());
            return false;
        }
}
```

Funcion crea parqueadero se ejecutara tantas veces como parqueaderos se quieran.

```
public boolean CrearParqueadero(Parqueadero parqueadero){
    PreparedStatement ps;
        try {
            ps = conexion.prepareStatement("INSERT INTO parqueadero(idp, disponible, relacion)VALUES(?,?,?)");
            ps.setInt(1, parqueadero.getIdp());
            ps.setBoolean(2, parqueadero.isDisponible());
            ps.setInt(3, parqueadero.getRelacion());
            return true;
        } catch (SQLException e) {
            System.out.println(e.toString());
            return false;
        }
}
```

Funcion InsertarCliente agrega un cliente a el sistema asignandole un id y credenciales de uso del parqueadero.

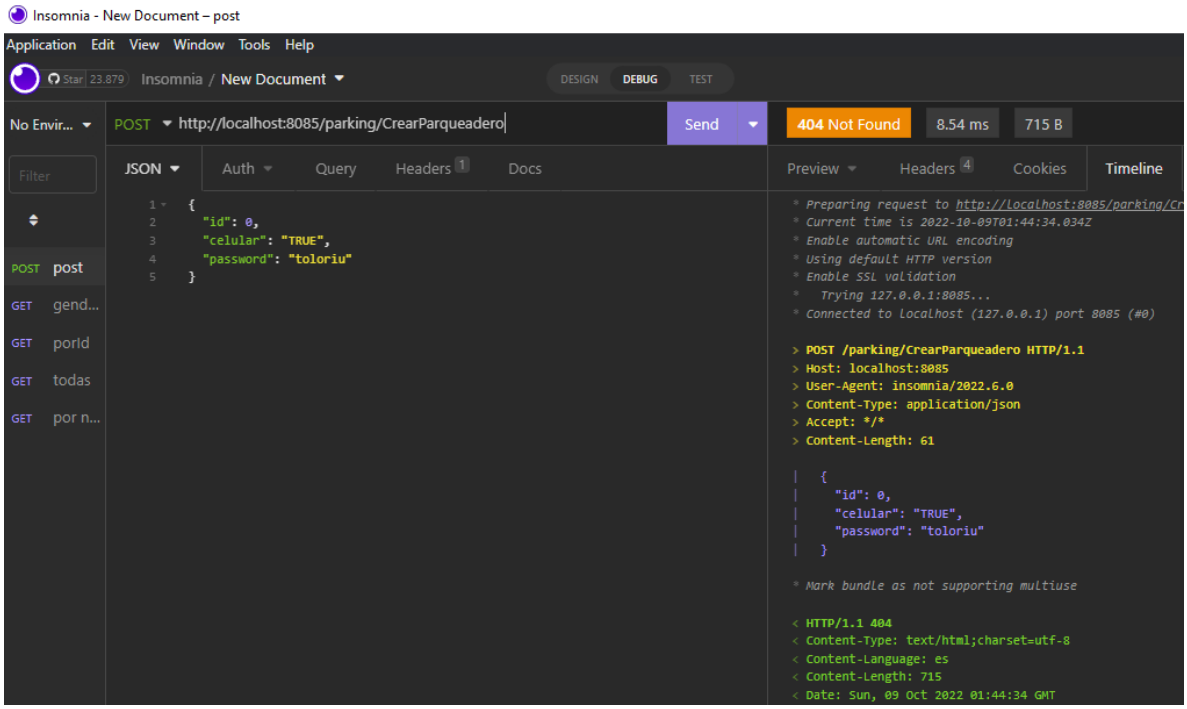
```
public boolean insertCliente(Cliente cliente){
    PreparedStatement ps;
        try {
            ps = conexion.prepareStatement("INSERT INTO cliente(nombre, celular, password)VALUES(?,?,?)");
            ps.setString(1, cliente.getNombre());
            ps.setInt(2, cliente.getCelular());
            ps.setString(3, cliente.getPassword());
            return true;
        } catch (SQLException e) {
            System.out.println(e.toString());
            return false;
        }
}
```

Funcion ModificarParqueadero se ejecutara cuando ingrese un cliente y tome el puesto se modificaara laa relacion entre parquadero y cliente agregandole informacion de el cliente

```java
public boolean modificarParqueadero(Parqueadero parqueadero){
    PreparedStatement ps;
        try {
            ps = conexion.prepareStatement("UPDATE parqueadero SET disponible WHERE:relacion=NOT NULL");
            ps.setBoolean(1, parqueadero.isDisponible());
            ps.setInt(2, parqueadero.getRelacion());
            ps.execute();
            return true;
        } catch (SQLException e) {
            System.out.println(e.toString());
            return false;
        }
}
```

b. Pruebas unitarias de la lógica desarrollada

Las primeras pruebas Arrojan errores los cuales en el desarrollo no se evidenciaros



2. Informe de retrospectiva
   Para lograr poner a punto una aplicación desacoplada en componentes MVC es muy necesario tener en cuenta la funcionalidad de los equipos usados para crear el proyecto dada que en casos puede hacer que no función en cosas las cuales funcionan y perder mucho tiempo intentando arreglar el código sin tener éxito.

3. Historias de usuario a desarrollar en el sprint 3. (Trello)
[historia de usuario]