Parking La Laguna:

Administración y Diseño de Base de Datos

Álvaro Rodríguez Gómez || alu0101362953@ull.edu.es
Paula Regalado de León || alu0101330174@ull.edu.es
Jonay Estévez Díaz || alu0101100586@ull.edu.es

Índice:

1. Introducción

El proyecto final de la asignatura de Administración y Diseño de Bases de Datos consiste en la creación, diseño y administración de una base de datos relacional de una propuesta original, la cual podía ser un caso real o un simple supuesto. En nuestro caso, optamos por enfocarnos en un caso real, dado que nuestra base de datos se sumerge en la temática de los aparcamientos de coches, teniendo en cuenta las plazas, los garajes, los vehículos, así como los empleados, las reservas, pagos y descuentos entre otros elementos que afectan directamente.

2. Objetivos

Los objetivos de nuestro proyecto se enfocan en los siguientes puntos:

- Representar los requisitos de datos y las relaciones para administrar las operaciones de un aparcamiento de coches, incluído las propias plazas de aparcamiento, la reserva de las mismas, el pago, tarifas y gestión de empleados.
- Identificar las entidades y los atributos que deben incluirse en la base de datos como la marca y el modelo de los vehículos, el tamaño y la disponibilidad del espacio del estacionamiento, así como la información de contacto del cliente y las funciones de los empleados
- Definición de las relaciones entre entidades, como la relación entre un coche y una plaza de aparcamiento, o la relación entre una reserva y un pago
- Garantizar que la base de datos esté correctamente normalizada para minimizar la redundancia y la dependencia, y maximizar la integridad y flexibilidad de los datos.

3. Entidades y relaciones

Las entidades que podemos distinguir en nuestro modelo son las siguientes:

❖ **Employee**. Representa a los empleados que trabajan en alguno de los parkings. Podemos distinguirlos por un id, nombre, compuesto por su nombre de pila y sus

apellidos. Según su rol, el empleado se especializa en un trabajo en específico, los cuales pueden ser: Manager, cajero y segurata.

- **Car park**. Corresponde a la red de parkings registrados los cuales podemos distinguirlos por un id. Sus demás atributos son el nombre, la localización, el número total de plazas, horario y facturas.

- **Parking Space**. Representa las plazas que constituyen un parking. Está constituida por el número de filas y número de columnas representado en una misma cadena. Otros atributos a tener en cuenta son las dimensiones, constituidas por el ancho y el largo y un booleano en el que verifique que esta plaza está disponible o no.

- **Customer**. Representa a los clientes que hacen uso de los servicios del parking. Su clave primaria es su DNI. Sus atributos son el nombre, constituido por el nombre y apellidos, email y número de teléfono.

- **Car**. Almacena todos los coches que conduce un cliente, además de la vinculación con el propio cliente. Los datos que se guardan del coche son la matrícula, la marca y el modelo.

- **Reservation**. Corresponde a cada una de las reservas que han hecho a cualquiera de los parkings registrados. Además en esta se guarda el registro de entrada y salida del vehículo y el tiempo.

- **Complaint**. Entidad débil que corresponde con la descripción de un problema asociado a una reserva ya establecida. Se guardan los datos que lo vincula con la reserva, así como el problema del cliente y la solución llevada a cabo.

- **Discount.** Almacena todos los descuentos que se le pueden aplicar a una reserva, guardando el nombre del descuento y el porcentaje de descuento que aplica.

- **Payment**. Forma de pago que usamos para pagar la reserva. Está es una entidad débil ya que si no hay reserva, no habría que realizar un pago. Los principales atributos que tiene el el valor monetario del pago a realizar, así como el tipo de pago, es decir, si se pagó utilizando dinero en efectivo, tarjeta de crédito o de débito.

4. Especificaciones de Requisitos

Representaremos que cada parking tenga un conjunto de empleados asignados. Un empleado solo puede estar asociado a un parking. Cada empleado tiene un rol asignado (manager, cajero y segurata), no puede ejercer más de uno a la vez.

Un parking está constituido por un número de plazas. Estas pueden ser de dos tipos: Normal y para gente con movilidad reducida, las cuales pueden ser, personas con discapacidades o embarazadas. Cada plaza, al ser un espacio físico, con dimensiones de largo y ancho, está asignada a un solo parking. Las plazas de aparcamiento tienen asignadas un nombre alfanumérico, compuesto por dos letras o una letra y un número.

Cualquier cliente puede hacer una o más reservas y este, estará asignado a un coche, del cual guardamos los datos, en el momento de una reserva. El cliente puede aparcar en un parking. Cada reserva tiene una forma de pago asignada. Puede tener diferentes formas de pago: Efectivo, tarjeta de crédito y tarjetas de débito. En las reservas, deben constar las los tiempos de entrada y salida. Se tiene en cuenta que puede que la reserva todavía está vigente, es por ello que la fecha de salida puede no estar especificada, en caso de que uno de los disparadores creará una excepción si la fecha de salida es menor que la de entrada. Cuando se crea una reserva y no se especifica un tiempo de entrada, significa que la reserva es para el momento en que se realiza, y gracias a otro disparador, podemos actualizar la tupla de la reserva indicando el momento actuar en el que se ingresó la tupla.

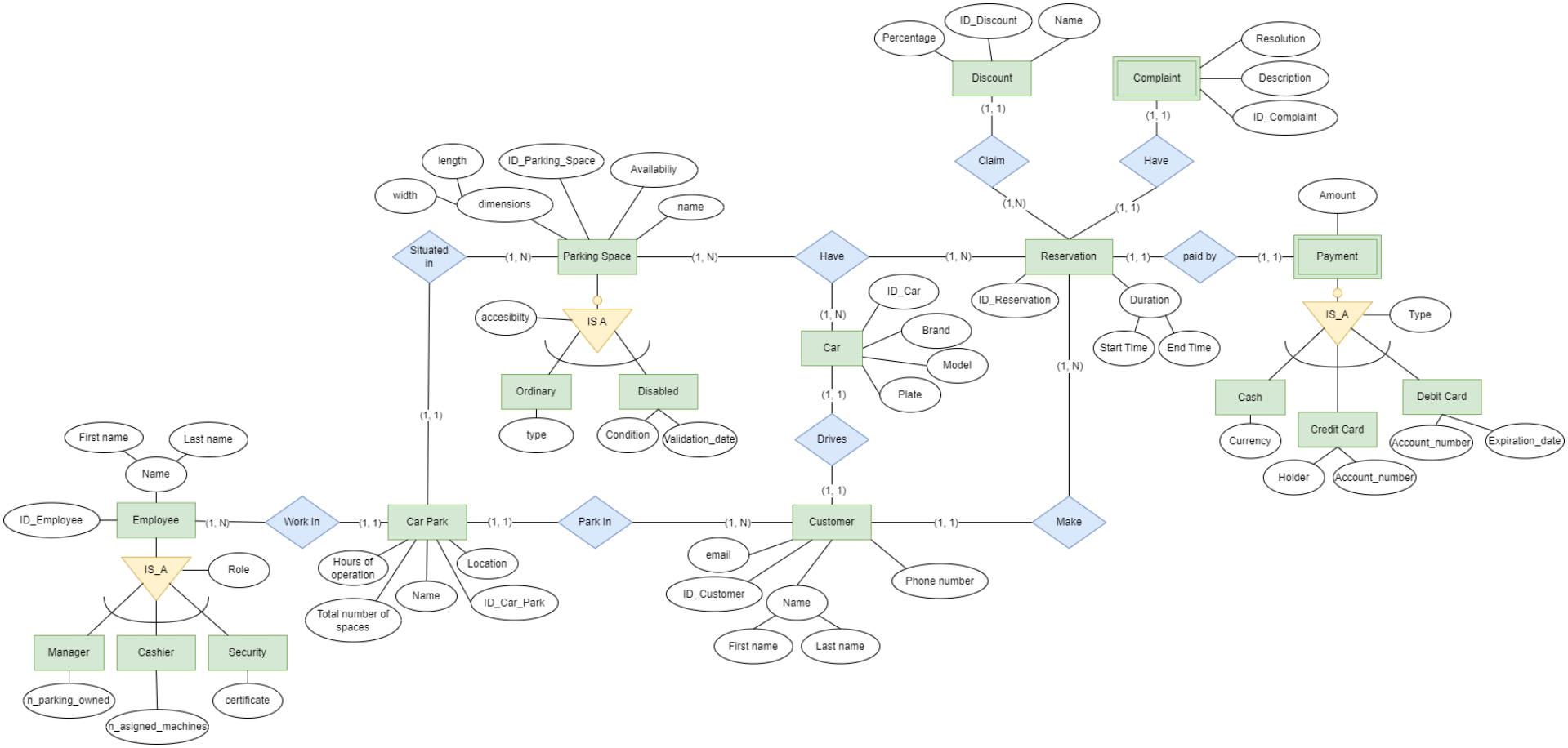
Los coches se guardan en nuestra base de datos dependiendo del cliente que lo conduce, así mismo, almacenamos la información de la matrícula, el modelo y marca del mismo.

Cuando se aplica una reserva, se guarda la información de reserva, la plaza de garaje a la que aplica y el coche que se va a situar en dicho espacio. Además de actualiza automáticamente mediante el uso de disparadores algunos aspectos de la base de datos, como, en éste caso, la actualización de la disponibilidad de la plaza, pues si está reservada, no de puede ofrecer a otro cliente distinto

Las reservas también son afectadas por descuentos, éstos son descuentos que tienen un rango del 10 al 100 por ciento de descuento. Por otro lado, tenemos las quejas de las reservas existentes, por las que un cliente puede reclamar cualquier problema que tenga con su reserva, el aparcamiento asignado, etc.

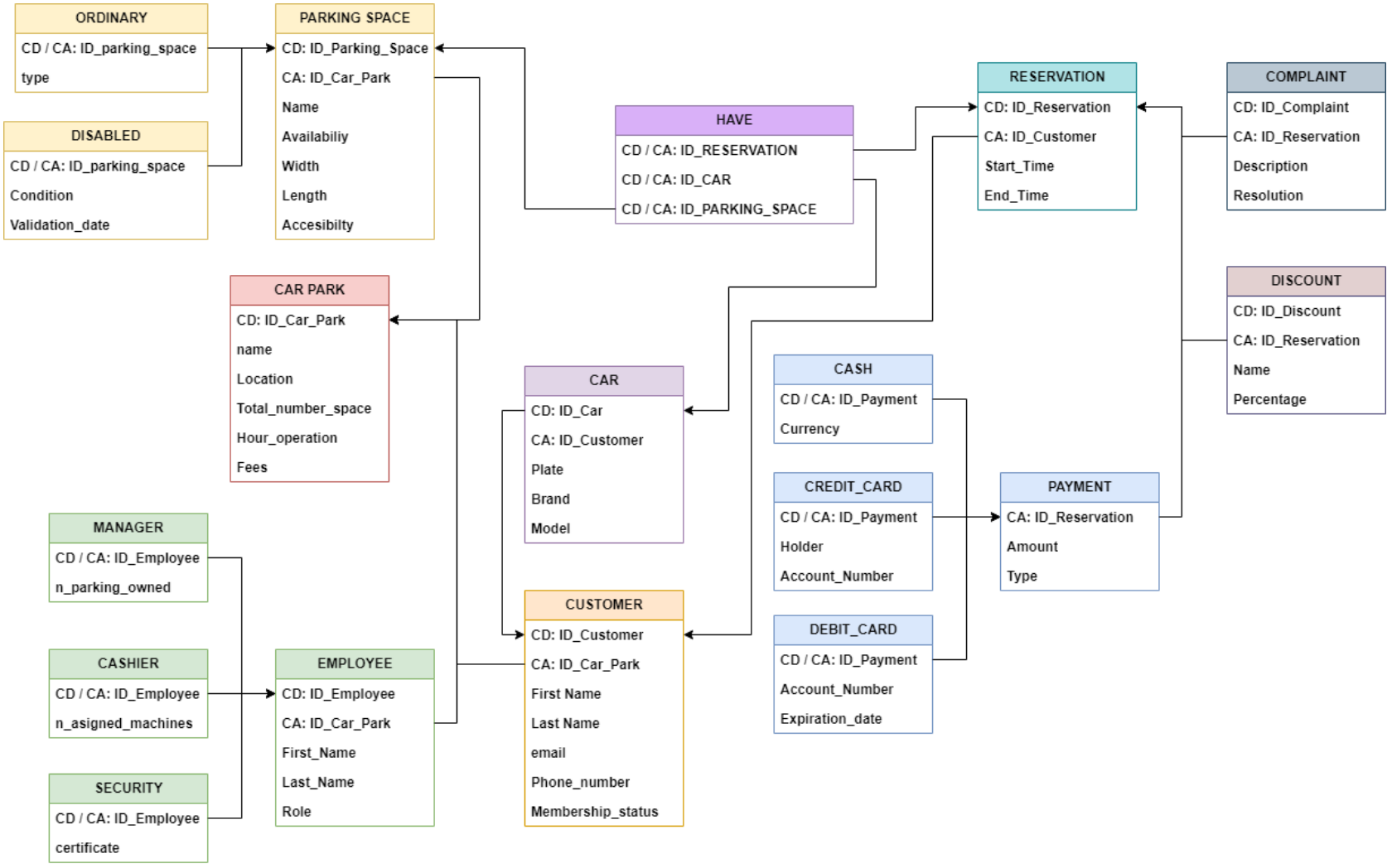
5. Modelo Entidad - Relación

El modelo Entidad - Relación desarrollado se puede consultar en la siguiente imagen:



6. Modelo Relacional

El modelo relacional obtenido, tras la transformación del modelo entidad-relación, es el siguiente:



7. Script SQL

Para la implementación de la base de datos, utilizamos un script sql, que conecta con el sistema gestor de bases de datos relacional Postgresql. Dicho fichero se puede consultar en el siguiente enlace [Link_script_Parking].

En primer lugar, se realiza la creación de la base de datos sobre la que se montan las tablas que la compondrán y se realiza la conexión a la base de datos recién creada.

Para continuar, lo que se realiza es la creación de las tablas, además como se puede observar en el modelo relacional, se crearon tablas para las relaciones triples y tablas para especificar el tipo de empleado, plaza de aparcamiento y pago. Como se puede observar en el script, se ingresaron algunas condiciones en la inserción de los datos de las tablas, como por ejemplo el uso de valores que no pueden estar vacíos, lo cual se realiza con el uso de “NOT NULL”. Por otro lado se asignaron las restricciones que aplican a las claves ajenas de otras tablas y se usó, para todas las tablas le eliminación en cascada.

Tras la creación de las tablas se realizaron las verificaciones o “checks” necesarias para las tablas oportunas (Líneas 283-333). Las verificaciones e implementadas fueron las siguientes:

- Los números de teléfono de los clientes deben ser únicos.
- El precio a pagar (payment.amount) debe ser mayor que 0.
- Los tipos de pago están restringidos a tres posibles valores: `'cash', 'credit card', 'debit card'`.
- Los coches deben tener una matrícula única.
- El nombre de los aparcamientos debe ser único.
- Los aparcamientos deben tener un mínimo de 10 plazas
- Los aparcamientos deben tener un mínimo de 8 horas abiertos.
- Los administradores de los aparcamientos deben de tener en propiedad como mínimo un aparcamiento, ya que si no no serían administradores.

- La accesibilidad de los aparcamientos está restringido a dos casos: `'ordinary'`, `'disabled'`
- Los aparcamientos ordinarios pueden ser del tipo `'touring car'` o `'boxcar'`
- Solo hay dos tipos de usuarios, dado el estatus de subscripción `'none'`, `'vip'`
- Los descuentos no pueden pasar de un 90 por ciento.

En cuanto a los disparadores, se diseñaron seis disparadores diferentes. el primer disparador de nombre **check_expiration_date** se aplica antes de insertar o actualizar la tabla debit card, y para cada fila, ejecuta la función **check_expiration_date** que comprueba si la nueva fecha de expiración introducida es menor que la fecha actual, se lanza una excepción que manda el mensaje de que la tarjeta ha expirado

Para el segundo disparador, se aplica antes de cualquier inserción en la tabla de **reservation**, donde llamamos a la función **check_reservation_start_end_time** que primero comprueba si existe una fecha de salida y si es así, comprueba que el tiempo de entrada es mayor que el de salida, creando así una excepción de que no tiene sentido la definición de los tiempos de la reserva.

Los dos siguientes disparadores hacen uso de la misma función, ya que lo que se comprueba en un atributo similar en ambas tablas. La función verifica si el número de cuenta es único, es decir, que el número de cuenta debe ser único tanto para las tarjetas de crédito como para las de débito. Si el select de la función retorna un 0 es que el número de cuenta es único. Ésto se realiza antes de la inserción o actualización de las tablas credit card y debit card para cada tupla.

El disparador **update_reservation_start_time** de ejecuta antes de la inserción en la tabla reservation, y ejecuta la función **update_start_time_to_current** que si la fecha de inicio de la reserva está vacío, cambia su valor a la fecha y hora actuales.

El último disparador es el encargado de actualizar la disponibilidad de un aparcamiento de la base de datos, cambiando su valor por false, ya que es una atributo booleano, donde la plaza tenga un coche y una reserva. Se ejecuta después de cualquier inserción en la tabla have

Como parte final del script se realizó la carga de datos, para ello se realizó para cada tabla una inserción múltiple (Líneas 432-674).

8. API REST con Flask (CRUD)

Las operaciones realizadas en la API REST con Flask se pueden consultar en el fichero app.py [Link_app_Flask] y en las templates creadas [Link_Templates].

8.1. Crear

Realizamos un formulario, **reservate.html**, que pide al usuario diferentes datos, como el coche, el aparcamiento, el parking, sus datos, etc. Con ellos, luego en la ruta de la app, comprobamos que los datos ingresados, no están vacíos, y si el parking seleccionado existe dentro de la base de datos. Tras ésto, se realizan con el uso del **cur** las inserciones de toda la información en las tablas de la base de datos.

8.2. Leer

Para llevar a cabo la lectura, primero se estableció una nueva ruta de nuestra app, en este caso **/read**, si el método no es post, se renderiza **read.html**, donde encontraremos un formulario para seleccionar la tabla a consultar, por otro lado, si el método requerido es post, lo que se hace es obtener el nombre de la tabla, y se realiza la conexión con la base de datos y se obtienen los atributos de la tabla, ya que usamos una consulta que nos permite obtener el schema de la tabla seleccionada, luego se almacenan todas las tuplas de la tabla y se renderiza el html que muestra las tablas de nombre **table.html**.
Para la sección principal, la plantilla usa un bucle for para iterar sobre la tupla que tiene los nombres de los atributos, que representa los nombres de las columnas de la tabla. Luego, muestra el nombre de cada columna en un elemento th.
Para la sección del tbody, la plantilla usa otro bucle for para iterar sobre cada fila de datos, luego usa otro bucle for anidado para acceder a los valores de cada fila y los muestra en elementos td.

8.3. Modificar

En cuanto a la modificación, tenemos dos rutas, la primera **/update** nos permite pedir al usuario que tabla desea actualizar, luego se msn redirigirá a **/updatefromtable,** donde se nos pedirán los datos a ingresar para actualizar la tupla o tuplas afectadas por el cambio.

8.4. Eliminar

Para eliminar, lo que se realiza es la selección de la tabla donde se necesita eliminar información, tras ésto, ingresa otro formulario, donde debemos escoger cual es

la columna y el valor para realizar la eliminación. Se crearon dos rutas, la primera en **/delete**, que es donde seleccionaremos la tabla y luego es en **/deletefromtable** donde se selecciona y se ingresa el valor.

9. Consultas y Operaciones sobre base de datos

9.1. Consultas

Las consultas que se realizaron están descritas a continuación, junto con la salida por terminal de las mismas, además pueden ser consultadas en el enlace [Link_Consultas].

- Matrícula y nombre del cliente aparcado en plazas para embarazadas:

```
-- 1) Matricula y nombre del cliente aparcado en plazas para embarazadas
SELECT car.plate AS Matricula, car.brand AS Marca, car.model AS Modelo,
       customer.first_name as Nombre_Cliente, customer.last_name AS Apellido_Cliente
FROM car
JOIN customer ON car.id_customer = customer.id_customer
JOIN have ON car.id_car = have.id_car
JOIN parking_space ON have.id_parking_space = parking_space.id_parking_space
JOIN disabled ON parking_space.id_parking_space = disabled.id_parking_space
WHERE disabled.condition = 'disabled';
```

```
 matricula |  marca   |   modelo   | nombre_cliente | apellido_cliente
-----------+----------+------------+----------------+------------------
 0030HTH   | Opel     | Corsa      | Marcos         | Reyes Contreras
 7878DDV   | Mercedes | Marco Polo | Cristo         | Socas Gutierrez
(2 rows)
```

- Número de plazas disponibles para cada Parking:

```
-- 2) Numero de plazas disponibles para cada Parking
SELECT car_park.park_name AS Nombre_Aparcamiento,
       COUNT(parking_space.id_parking_space) as Plazas_disponibles_para_reserva
FROM car_park
INNER JOIN parking_space ON car_park.id_car_park = parking_space.id_car_park
WHERE parking_space.availability = 'true'
GROUP BY car_park.park_name;
```

```
 nombre_aparcamiento  | plazas_disponibles_para_reserva
----------------------+---------------------------------
 Parking Santa Cruz   |                               3
 Parking La Laguna    |                               3
 Parking Concepcion   |                               2
 Parking Los Realejos |                               2
(4 rows)
```

- Todos los parking y para cada uno los trabajadores del mismo:

```sql
-- 3) Todos los parking y para cada uno los trabajadores del mismo
SELECT car_park.park_name AS Nombre_Aparcamiento, employee.first_name AS Nombre_Empleado,
       employee.last_name AS Apellido_Empleado, employee.role_name AS Ocupacion,
       manager.n_parking_owned AS Aparcamientos_en_Propiedad,
       cashier.n_asigned_machines AS Maquinas_asignadas,
       security.certificate AS Certificado_Seguridad
FROM car_park
LEFT JOIN employee ON car_park.id_car_park = employee.id_car_park
LEFT JOIN manager ON employee.id_employee = manager.id_employee
LEFT JOIN cashier ON employee.id_employee = cashier.id_employee
LEFT JOIN security ON employee.id_employee = security.id_employee;
```

```
 nombre_aparcamiento  | nombre_empleado | apellido_empleado  | ocupacion | aparcamientos_en_propiedad | maquinas_asignadas |       certificado_seguridad
----------------------+-----------------+--------------------+-----------+----------------------------+--------------------+-----------------------------------
 Parking La Laguna    | Antonio         | Guijarro Rodríguez | Cashier   |                            |                  1 |
 Parking Santa Cruz   | María           | Hernández Bello    | Cashier   |                            |                  1 |
 Parking Concepcion   | Nadia           | Esquivel Cruz      | Cashier   |                            |                  1 |
 Parking Los Realejos | Cristian        | Redondo Rojas      | Cashier   |                            |                  1 |
 Parking Los Realejos | Manolo          | Díaz Díaz          | Manager   |                          1 |                    |
 Parking Santa Cruz   | Bruno           | Estévez Estévez    | Manager   |                          1 |                    |
 Parking La Laguna    | Manuel          | Hernandez Díaz     | Manager   |                          2 |                    |
 Parking La Laguna    | Juliana         | Pérez Laya         | Security  |                            |                    | Certificado en Seguridad Privada
 Parking Concepcion   | Lazaro          | Martin Esquivel    | Security  |                            |                    | Certificado de Segurita de Prosegur
(9 rows)
```

- Todos los clientes que han reservado en enero del 2023 y el número de reserva:

```sql
-- 4) Todos los clientes que han reservado en enero del 2023 y el numero de reserva
SELECT customer.first_name AS Nombre_Cliente, customer.last_name AS Apellidos_Cliente,
       reservation.id_reservation AS Numero_reserva
FROM customer
INNER JOIN reservation ON customer.id_customer = reservation.id_customer
WHERE EXTRACT(YEAR FROM reservation.start_time) = 2023 AND
      EXTRACT(MONTH FROM reservation.start_time) = 1;
```

nombre_cliente	apellidos_cliente	numero_reserva
Marcos	Reyes Contreras	3
Violeta	Rodriguez Hernández	4
Pablo	Gómez Gómez	5
Jonay	Estévez Díaz	6
Álvaro	Rodríguez Gómez	7
Paula	Regalado de León	8
Marcos	Reyes Contreras	9
Maria	Díaz Díaz	10
Pedro	García Martin	11
Amanda	Garcia González	12
Juan	Laya, Hernández	13
Cristo	Socas Gutierrez	14

(12 rows)

- Consultar el nombre, matricula, fecha de inicio, fecha de salida, nombre de la plaza y el nombre del parking:

```sql
-- 5) Consultar el nombre, matricula, fecha de inicio, fecha de salida,
--    nombre de la plaza y el nombre del parking
SELECT customer.first_name AS Nombre_cliente, customer.last_name AS Apellido_Cliente,
       car.plate AS Matricula, reservation.start_time AS Fecha_inicio,
       reservation.end_time AS Fecha_salida, parking_space.name AS Nombre_Plaza,
       car_park.park_name AS Aparcamiento
FROM customer
INNER JOIN car ON customer.id_customer = car.id_customer
INNER JOIN have ON car.id_car = have.id_car
INNER JOIN reservation ON have.id_reservation = reservation.id_reservation
INNER JOIN parking_space ON have.id_parking_space = parking_space.id_parking_space
INNER JOIN car_park ON parking_space.id_car_park = car_park.id_car_park;
```

nombre_cliente	apellido_cliente	matricula	fecha_inicio	fecha_salida	nombre_plaza	aparcamiento
Pablo	Gómez Gómez	0331BCD	2022-12-31 10:20:00	2022-12-31 15:28:49	A2	Parking La Laguna
Maria	Perez Gutierrez	0768DFG	2022-12-30 19:10:00	2022-12-30 21:45:50	AA	Parking Concepcion
Marcos	Reyes Contreras	0030HTH	2023-01-01 18:30:00	2023-01-02 20:30:00	A2	Parking La Laguna
Violeta	Rodriguez Hernández	0404LMN	2023-01-02 09:00:00	2023-01-02 09:30:00	B2	Parking Los Realejos
Pablo	Gómez Gómez	0331BCD	2023-01-03 11:15:00	2023-01-02 14:15:00	AB	Parking Concepcion
Jonay	Estévez Díaz	9843FYR	2023-01-04 15:30:00	2023-01-02 16:00:00	C5	Parking Santa Cruz
Álvaro	Rodríguez Gómez	5619DRT	2023-01-05 20:00:00	2023-01-02 23:59:00	AA	Parking Concepcion
Paula	Regalado de León	4897DQR	2023-01-06 17:45:00		A2	Parking La Laguna
Marcos	Reyes Contreras	0030HTH	2023-01-07 13:00:00		A3	Parking La Laguna
Maria	Díaz Díaz	7865JJL	2023-01-08 08:10:00		C7	Parking Santa Cruz
Pedro	García Martin	7653HTL	2023-01-09 10:05:00		AC	Parking Concepcion
Amanda	Garcia González	1205CHJ	2023-01-10 07:35:00		AD	Parking Concepcion
Juan	Laya, Hernández	3827GYR	2023-01-10 12:20:00		B1	Parking Los Realejos
Cristo	Socas Gutierrez	7878DDV	2023-01-10 06:55:00		B5	Parking Los Realejos

(14 rows)

- Clientes a los que se les ha aplicado descuento y el porcentaje de descuento:

```sql
-- 6) Clientes a los que se les ha aplicado descuento y el porcentaje de descuento
SELECT customer.first_name AS Nombre_Cliente, customer.last_name AS Apellido_Cliente,
       reservation.id_reservation AS Numero_reserva, discount.percentage AS Porcentaje_descuento
FROM customer
INNER JOIN reservation ON customer.id_customer = reservation.id_customer
INNER JOIN discount ON reservation.id_reservation = discount.id_reservation;
```

```
 nombre_cliente | apellido_cliente | numero_reserva | porcentaje_descuento
----------------+------------------+----------------+----------------------
 Jonay          | Estévez Díaz     |              6 |                   25
 Maria          | Díaz Díaz        |             10 |                   50
 Amanda         | Garcia González  |             12 |                   75
 Juan           | Laya, Hernández  |             13 |                   80
(4 rows)
```

- Clientes que han reservado más de una vez y donde:

```sql
-- 7) Clientes que han reservado más de una vez y donde
SELECT customer.first_name AS Nombre_Cliente, customer.last_name AS Apellido_Cliente,
       car_park.park_name AS Aparcamiento
FROM customer
INNER JOIN reservation ON customer.id_customer = reservation.id_customer
INNER JOIN have ON reservation.id_reservation = have.id_reservation
INNER JOIN parking_space ON have.id_parking_space = parking_space.id_parking_space
INNER JOIN car_park ON parking_space.id_car_park = car_park.id_car_park
GROUP BY customer.id_customer, car_park.id_car_park
HAVING COUNT(reservation.id_reservation) > 1;
```

```
 nombre_cliente | apellido_cliente |   aparcamiento
----------------+------------------+------------------
 Marcos         | Reyes Contreras  | Parking La Laguna
(1 row)
```

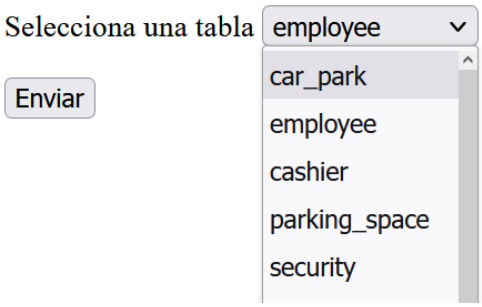
- Parking que ha tenido más reservas:

```sql
-- 8) Parking que ha tenido más reservas
SELECT car_park.park_name AS Aparcamiento, car_park.fees AS Tarifa,
       COUNT(reservation.id_reservation) as Numero_Total_Reservas
FROM car_park
LEFT JOIN parking_space ON car_park.id_car_park = parking_space.id_car_park
LEFT JOIN have ON parking_space.id_parking_space = have.id_parking_space
LEFT JOIN reservation ON have.id_reservation = reservation.id_reservation
GROUP BY car_park.id_car_park
ORDER BY COUNT(reservation.id_reservation) DESC
LIMIT 1;
```

```
    aparcamiento     | tarifa | numero_total_reservas
---------------------+--------+-----------------------
 Parking Concepcion |   1.05 |                     5
(1 row)
```

9.2. Operaciones

- Lectura de una tabla: en estos casos no pueden existir errores dado que el usuario no puede introducir otro valor que no esté dispuesto por el programa.



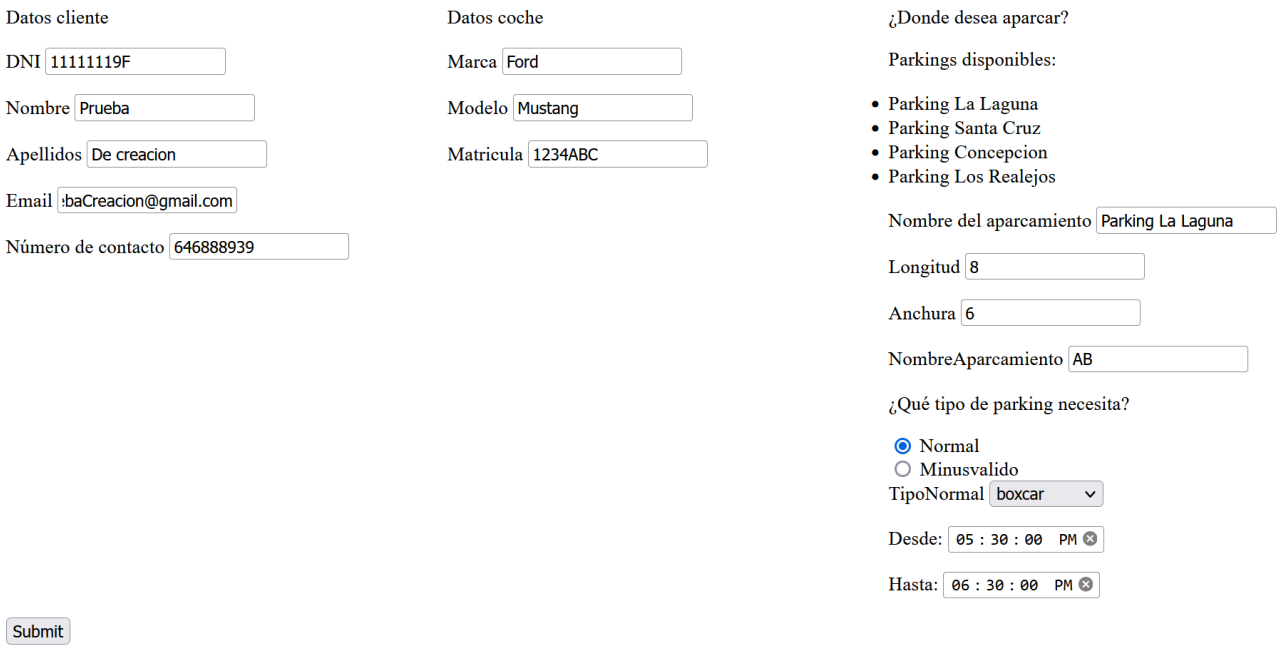
⇒

id_car	id_customer	plate	brand	model
1	00000001A	0331BCD	Seat	Panda
2	00000002B	0768DFG	Opel	Astra
3	00000003C	0030HTH	Opel	Corsa
4	00000004D	0404LMN	Ford	Focus
5	00000005E	1205CHJ	Ford	Fiesta
6	00000006F	9843FYR	Porshe	Panamera
7	00000007G	5619DRT	Seat	Ibiza
8	00000008H	4897DQR	Dacia	Dokker
9	00000009I	7865JJL	Citroen	Van
10	00000010J	7653HTL	Fiat	Ducato
11	00000011L	3827GYR	Ford	Transit
12	00000012M	7878DDV	Mercedes	Marco Polo

⇓

id_customer	id_car_park	first_name	last_name	email	phone_number
00000001A	1	Pablo	Gómez Gómez	pabgomez@email.com	665786534
00000002B	1	Maria	Perez Gutierrez	marperez@email.com	647653894
00000003C	1	Marcos	Reyes Contreras	marcreyes@email.com	678906565
00000004D	2	Violeta	Rodriguez Hernández	viorguez@email.com	637452893
00000005E	2	Amanda	Garcia González	amandagar@email.com	665545780
00000006F	2	Jonay	Estévez Díaz	jonayest@gmail.com	678456345
00000007G	3	Álvaro	Rodríguez Gómez	alvaroguez@gmail.com	723567765
00000008H	3	Paula	Regalado de León	paularegal@gmail.com	668554338
00000009I	3	Maria	Díaz Díaz	mauxidiaz@gmail.com	654546654
00000010J	4	Pedro	García Martin	pedrogar@gmail.com	788543342
00000011L	4	Juan	Laya, Hernández	juanlaya@gmail.com	778112345
00000012M	4	Cristo	Socas Gutierrez	crissocas@gmail.com	657345678
11111119F	1	Prueba	De creacion	pruebaCreacion@gmail.com	646888939

- Crear una reserva: esta operación involucra varias tablas de la base de datos a la vez.



Casos de error:

- Faltan campos:

Error

Debe completar todos los campos del formulario

- Campos repetidos en algunas tablas: muestran los errores controlados por Postgres con los checks que hemos creado.

duplicate key value violates unique constraint "customer_pkey"

duplicate key value violates unique constraint "check_car_plate"

9.2. Rutas de la aplicación

- Index: obtiene datos interesantes sobre el estado de los parkings.

Parking Status

Parking Concepcion ubicado en: La Laguna

Capacidad: 3 / 26

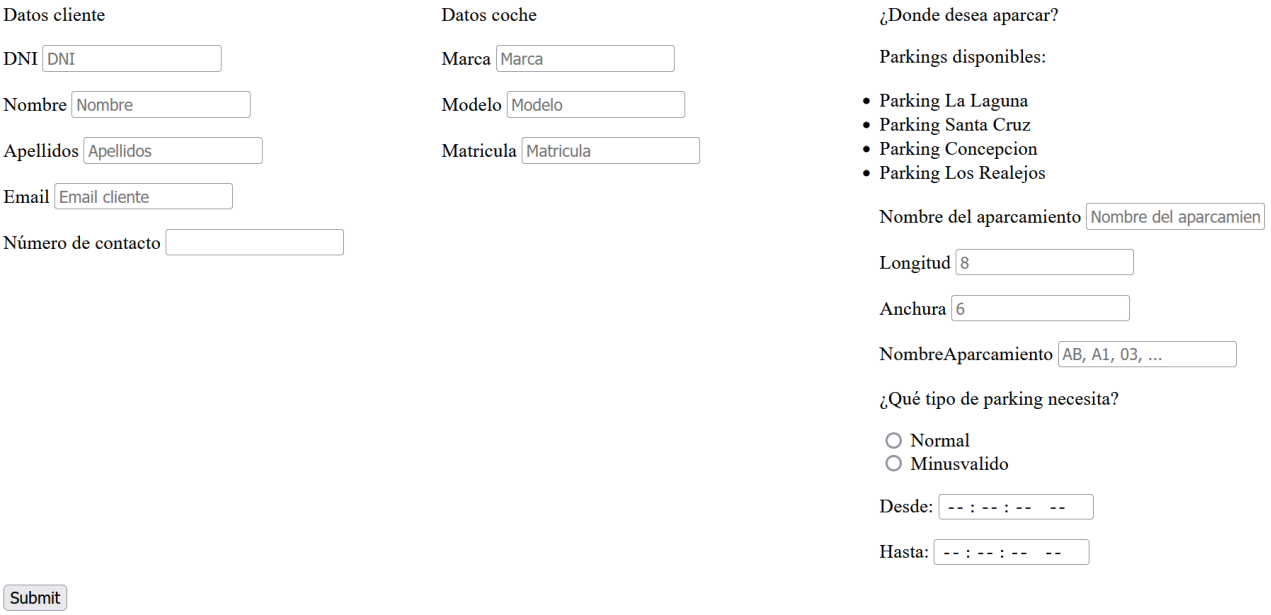
Parking Los Realejos ubicado en: Los Realejos

Capacidad: 3 / 50

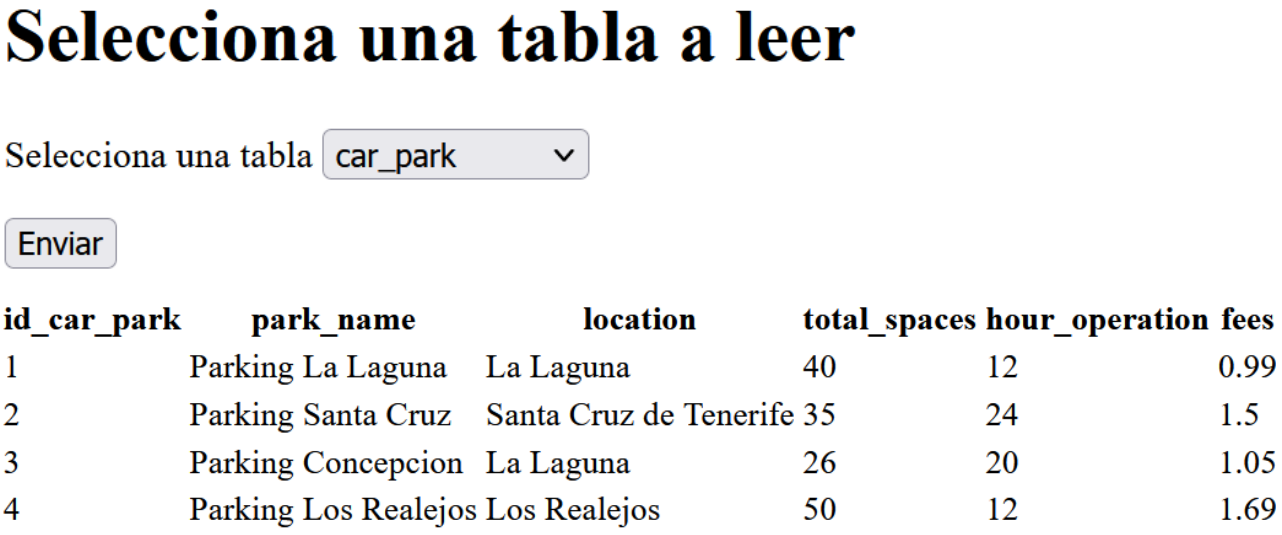
Parking Santa Cruz ubicado en: Santa Cruz de Tenerife

Capacidad: 3 / 35

- create: añade una nueva reserva a la base de datos con todos los datos necesarios para las tablas involucradas.



- read: permite seleccionar las tablas sobre la que leer todos sus datos.



- delete: solicita la tabla de la cual queremos borrar, y en la siguiente pantalla nos indica mediante qué campo queremos hacer la operación y qué valor usar.

Selecciona una tupla id_car_park

Valor 3

Enviar

id_car_park	park_name	location	total_spaces	hour_operation	fees
1	Parking La Laguna	La Laguna	40	12	0.99
2	Parking Santa Cruz	Santa Cruz de Tenerife	35	24	1.5
4	Parking Los Realejos	Los Realejos	50	12	1.69

- update: solicita la tabla de la cual queremos actualizar, y en la siguiente pantalla nos indica mediante qué campo queremos hacer la operación y qué valor usar.

Selecciona una tabla para actualizar

Selecciona una actualizar car_park

Enviar

Selecciona una tupla location

Valor La Laguna

Valor Santa Ursula

Enviar

id_car_park	park_name	location	total_spaces	hour_operation	fees
2	Parking Santa Cruz	Santa Cruz de Tenerife	35	24	1.5
4	Parking Los Realejos	Los Realejos	50	12	1.69
1	Parking La Laguna	Santa Ursula	40	12	0.99