

Universidad Don Bosco.



Facultad de Ingeniería.

Ingeniería en Ciencias de la Computación.

Asignatura: Programación con Estructura de Datos.

Docente: Ing. Carmen Celia Morales Samayoa.

Ciclo y grupo de clase: Ciclo II-2020 / 01T Aula B-22.

Proyecto: Parking Storage.

Integrantes de grupo:

Beltrán García Mario Josué

García Aparicio Sara Daniela

Hernández Villanueva Edgar Ernesto

Mejía Gámez José Ricardo

Rosales Mendoza Patrick Ernesto

Carné:

BG171969

GA190843

HV191966

MG190361

RM181976

Soyapango, 06/05/2020.

Índice.

Introducción.....	4
Lógica a utilizar para la solución del problema.....	5
Problemática.....	5
Solución de la problemática.....	5
Herramientas a utilizar.....	6
Diagramas UML de la solución del Problema.....	8
Diagramas de Caso de Uso.....	8
Módulo de Entrada y Salida del Parqueo.....	8
Módulo de Administrador del Sistema.....	8
Módulo de Secretaria.....	9
Diagrama de Clases.....	9
Diagramas de Secuencia.....	10
Módulo de Administradores.....	10
Módulo de Secretaria.....	11
Módulo de Entrada y Salida del Parqueo.....	12
Diagrama Relacional de la Base de Datos.....	13
Innovación aplicada en la Problemática y ¿Porque es Importante?.....	13
Diseño del Prototipo	15
Login.....	15
Módulo de Administrador.....	15
Formulario Carnets.....	16
Formulario Usuarios.....	18
Formulario Espacios	18

Formulario Zonas.....	19
Formulario Tipos de Estacionamientos.....	20
Formulario Tickets.	20
Módulo de Entrada.....	21
Módulo de Salida.	22
Módulo Secretaria.....	23
Tipos de Alerta.....	24
Evidencia del Uso de TADS Listas.....	25
Bibliografía.	29

Introducción.

A lo largo de la historia, los estacionamientos surgieron de forma que algunas personas se adueñaban de propiedades ajena, como calles y avenidas, cobrándoles un precio que muchas veces no era lo justo, personas que manejaban su vehículo, personas que tenían la necesidad de dejar estacionado su vehículo en un lugar en el que hipotéticamente estuvieran seguros; pues no podían llevarlo hasta su lugar de trabajo.

Hoy en día los estacionamientos han tomado parte en los alrededores de nuestro país, ya que funcionan de la misma manera antes mencionada; pero con la excepción de que ahora si existen establecimientos legalizados para dicho fin, con el objetivo de dejar nuestros vehículos en un lugar seguro y cerca de nuestros lugares de trabajo o de los lugares en los que nos encontramos.

El objetivo de nuestro proyecto es el de brindar a dichos establecimientos, nombrados como parqueos o estacionamientos como ya lo mencionábamos, un sistema de control y ordenamiento que permita registrar las entradas y salidas del mismo, sus espacios disponibles, además de notificar cuando, ya no se encuentre ningún espacio disponible, es decir que ya este todo el lugar ocupado.

Para llevar a cabo la realización de nuestro proyecto, desarrollaremos con las siguientes herramientas que son: el lenguaje de programación C#, por medio del cual se crearan las listas para el funcionamiento del estacionamiento, además utilizaremos el gestor de base de datos SQL Server para almacenar la información de cada uno de los usuarios participantes en el sistema y las librerías para generar los códigos QR para los tickets y carnets.

Lógica a utilizar para la solución del problema.

Problemática.

Actualmente con el incremento exponencial de vehículos la mayoría de parqueos tienen un problema en común, el cual es el desorden de los espacios ocupados y disponibles; debido a esto generan un congestionamiento dentro del área y fuera de los lugares como, por ejemplo: un centro comercial, empresas privadas, hospitales, locales, organizaciones gubernamentales, universidades y entidades educativas.

Al momento de entrar a un parqueo, en la mayoría de los casos no se encuentran espacios disponibles, en consecuencia, las personas rodean la zona para encontrar alguno disponible, lo cual ocasiona pérdidas de tiempo y estrés a los conductores, por lo general los parqueos no poseen áreas con secciones específicas para empleados, personas particulares, etc.

Solución de la problemática.

Se implementará un sistema informático para la organización de los espacios del parqueo, además incluiremos secciones para distintas personas, como empleados, personas particulares (clientes), personas con una discapacidad y gerentes.

Dentro del sistema se llevará un control de los vehículos que entran y salen del parqueo para gestionar los espacios del mismo y dividirlos entre disponibles y ocupados, siendo así eficaz en no generar congestionamiento dentro del sitio, por consiguiente, esto evitaría pérdida de tiempo para los conductores en encontrar un parqueo disponible. También hará un conteo de los espacios disponibles donde se mostrará en una pantalla, al igual cuando el parqueo tenga pocos espacios se les advertirá a los conductores que lo tengan en cuenta, y por último se le avisará que el parqueo está lleno.

Por otra parte, tendremos un apartado para los empleados y gerentes donde ellos se puedan registrar para generarle un espacio fijo en el parqueo y que estos puedan accesar libremente al espacio asignado con su propio carnet.

Para las personas particulares y con una discapacidad seleccionaran el tipo de persona, donde se le generará un ticket de entrada con la sección asignada y con el número de espacio correspondiente.

Al salir del parqueo se leerá el código del ticket (posiblemente se agregará el método) y procederá a mostrarle un mensaje de salida y el espacio ocupado pasará a estar disponible.

Herramientas a utilizar.

Se utilizará el lenguaje de programación C# para la realización del sistema y los módulos de administrador, secretaria y visualización de los parqueos disponibles y ocupados (simulación).

Además, con un sistema gestor de base de datos (SGBD) SQL Server para la recolección de los datos de los empleados, gerentes, el ingreso de espacios del parqueo y los usuarios del sistema.

Utilizaremos SQL Server, debido a que es compatible con los entornos de Visual Studio, así mismo con el lenguaje de programación C# ya que nos permite una mayor confiabilidad, escalabilidad y seguridad.

Librerías a Utilizar con sus Funciones:

- ✓ **La librería System.Collection.Generic:** nos permitirá aplicar la función de la lista por defecto que incluye el lenguaje C#. Nos permitirá utilizar el manejo de los datos de los empleados y gerentes al momento de ingresar, mostrar, editar o eliminar dentro de las listas.
- ✓ **La librería Zxing:** se utilizará para generar los códigos QR para el carnet de los empleados y para los tickets del estacionamiento donde aplicaremos un código único para cada uno, sin embargo, esta librería no está incluida en el lenguaje de C#, por esto la agregaremos a la parte de referencias de nuestra solución.
- ✓ **La librería IO:** permite leer y escribir en distintos tipos de archivos, en este caso se utilizara para archivos PDF.

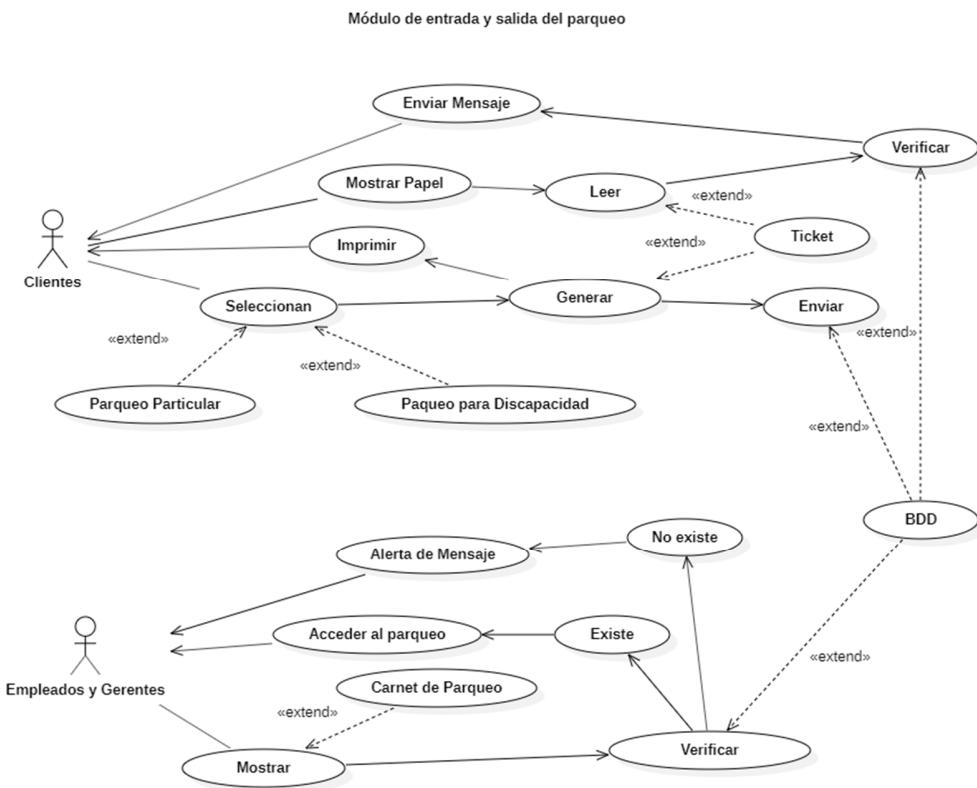
- ✓ **La librería Aforge.Video.DirectShow:** captura y guarda imágenes en webcam, en este sistema servirá para la lectura de los códigos QR de los tickets y carnets de los empleados.
- ✓ **La librería Drawing:** guarda las imágenes en forma de bytes para los códigos QR.
- ✓ **La librería CrystalDecisions.Shared:** se ocupa para generar reportes desde visual studio c#, y en este caso generara cada uno de los tickets en formato PDF y se guardara en la unidad C de nuestra PC.
- ✓ **La librería Security.Cryptography:** se utiliza para encriptar la contraseña en la parte de las validaciones de cada uno de los usuarios registrados en el sistema.
- ✓ **La librería Data.SqlClient:** nos permitirá hacer la conexión de C# con SQL Server para guardar cada uno de los registros en la base de datos.
- ✓ **La librería RegularExpressions:** se utiliza para hacer las validaciones de diferentes campos.

La simulación de parqueos disponibles y ocupados se realizará con TAD lista. En los espacios disponibles se utilizará una lista, igualmente con los espacios ocupados; donde un nodo de la lista disponible pasará a ser un nodo de la lista ocupado y se verá reflejado en la simulación.

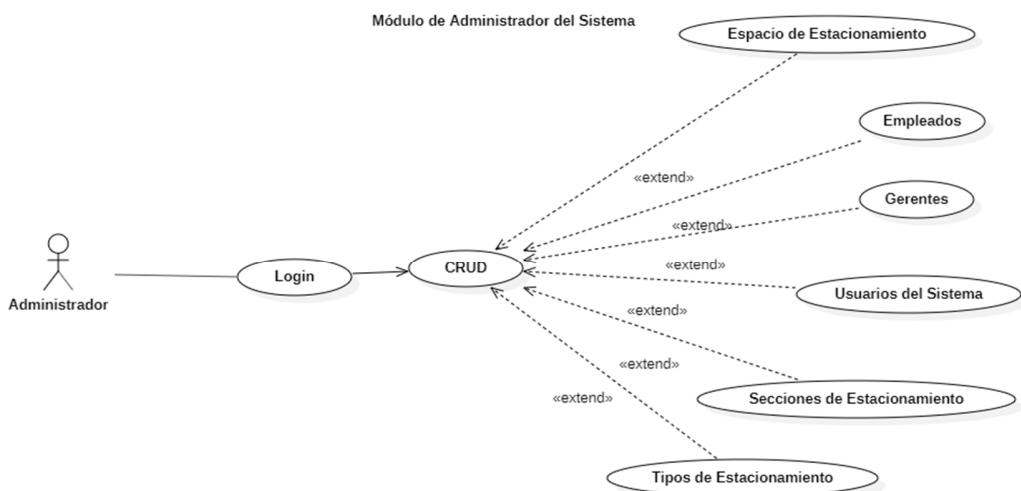
Diagramas UML de la solución del Problema.

Diagramas de Caso de Uso.

Módulo de Entrada y Salida del Parqueo.



Módulo de Administrador del Sistema.



Módulo de Secretaria.

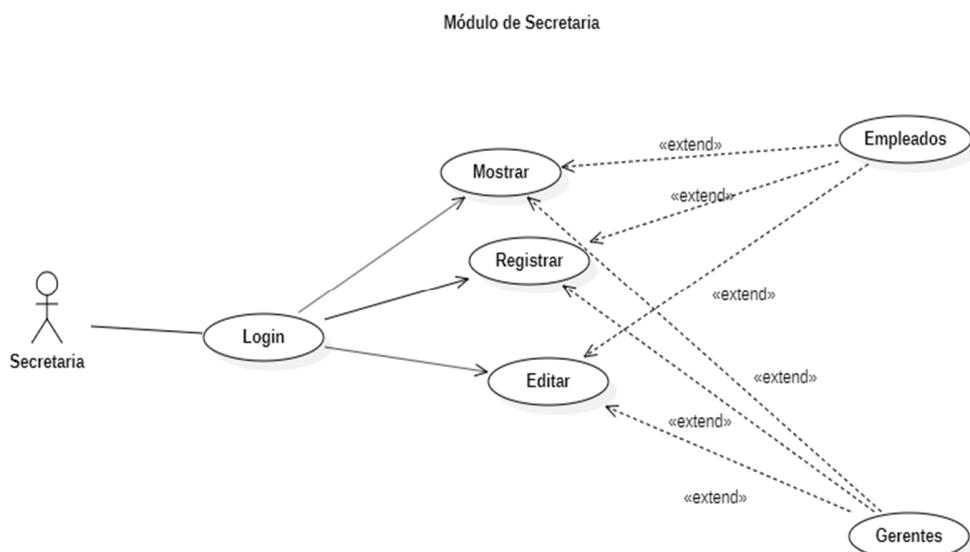
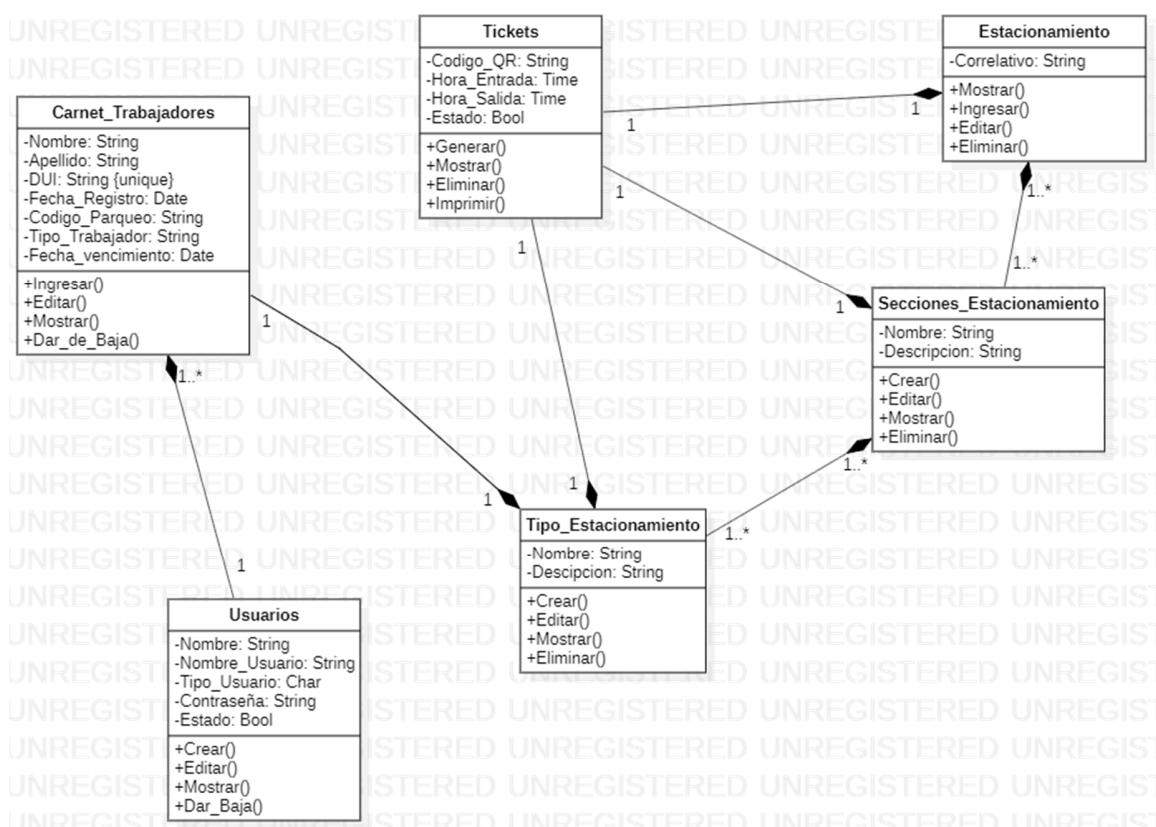
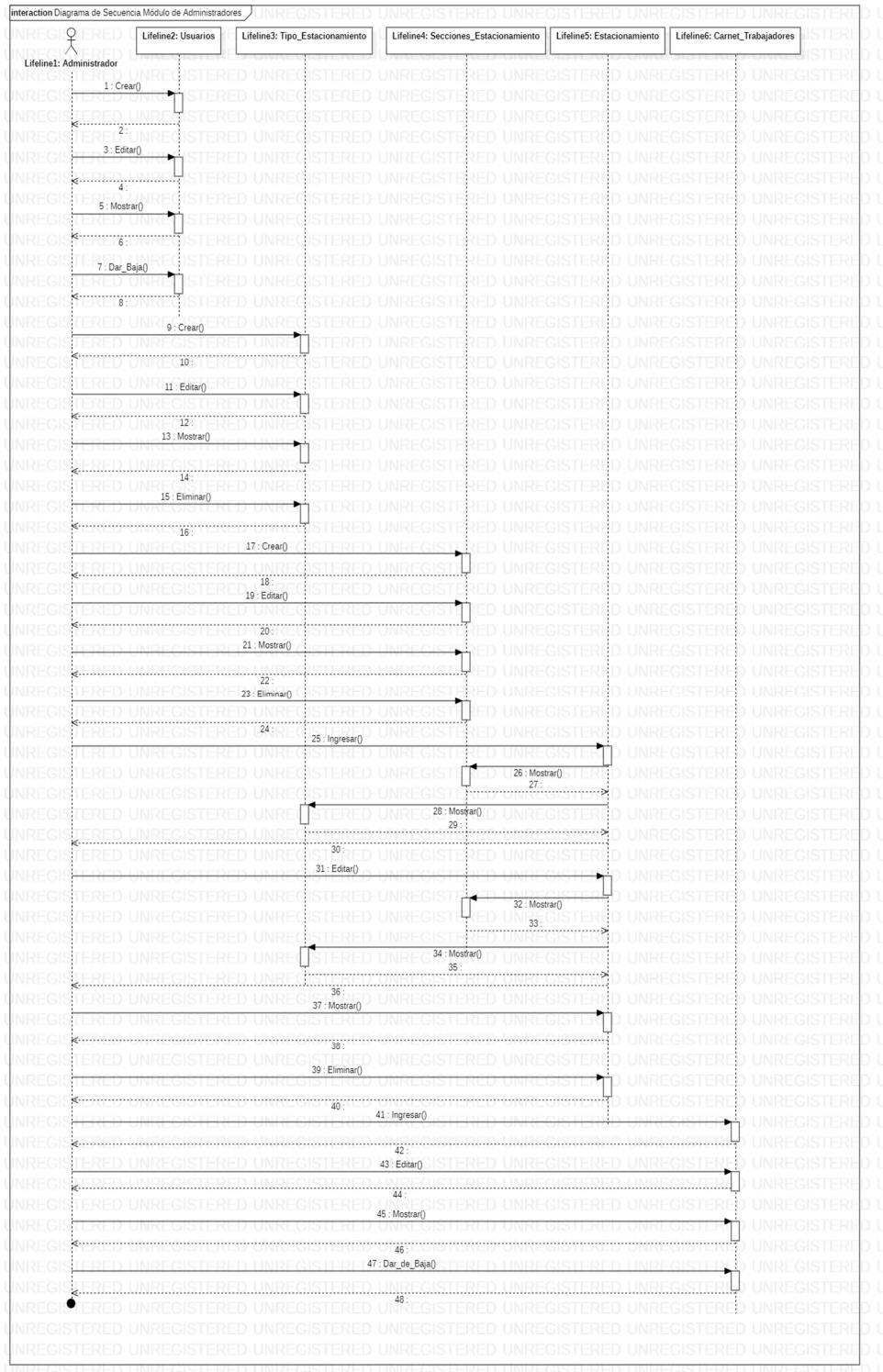


Diagrama de Clases.

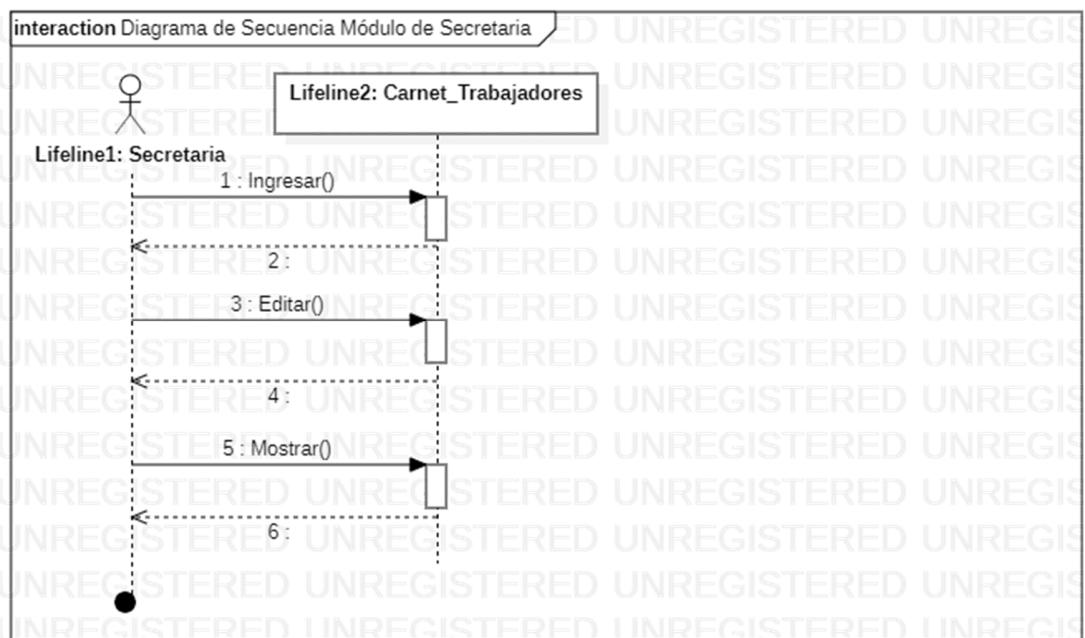


Diagramas de Secuencia

Módulo de Administradores.



Módulo de Secretaria.



Módulo de Entrada y Salida del Parqueo.

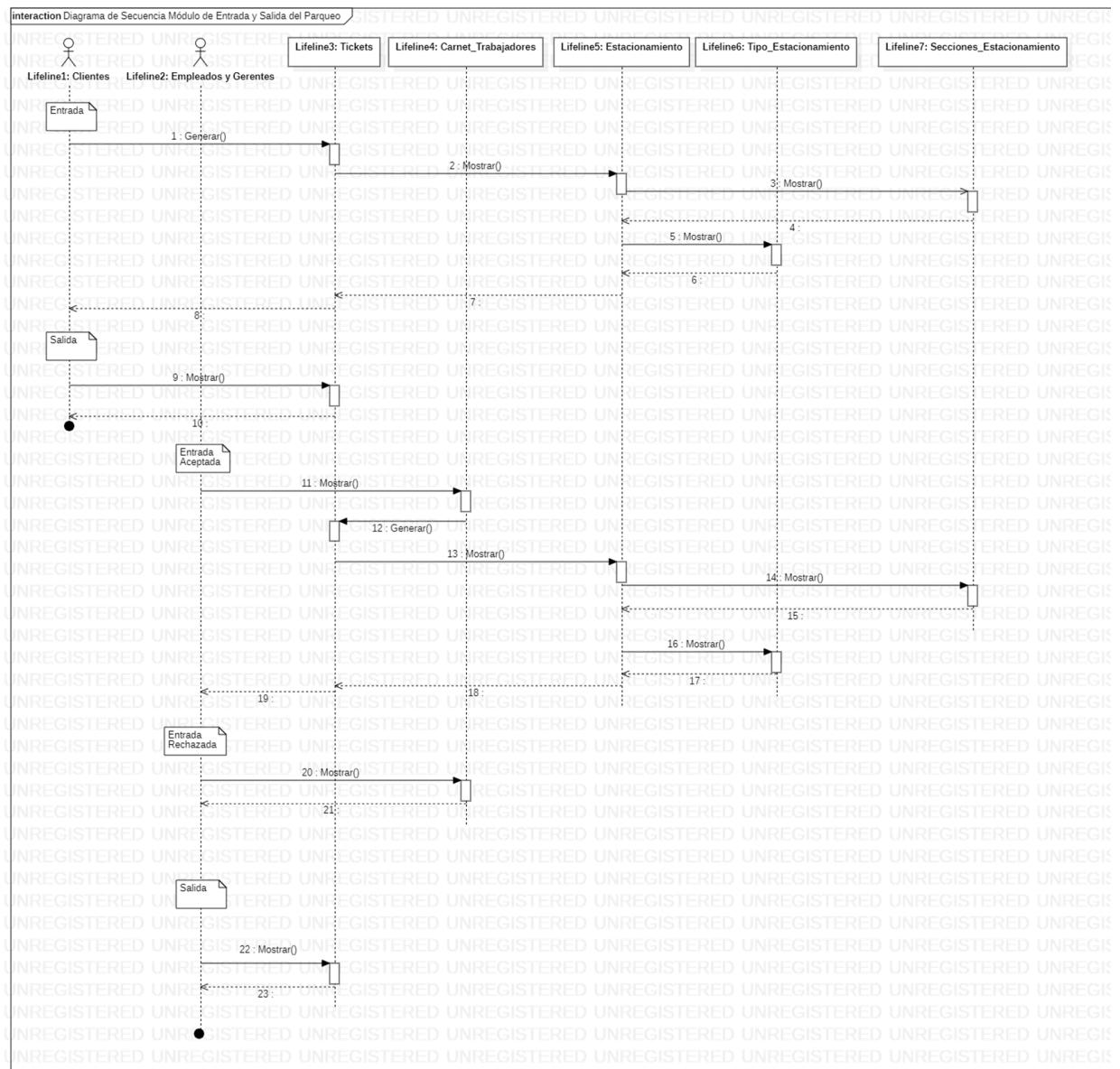
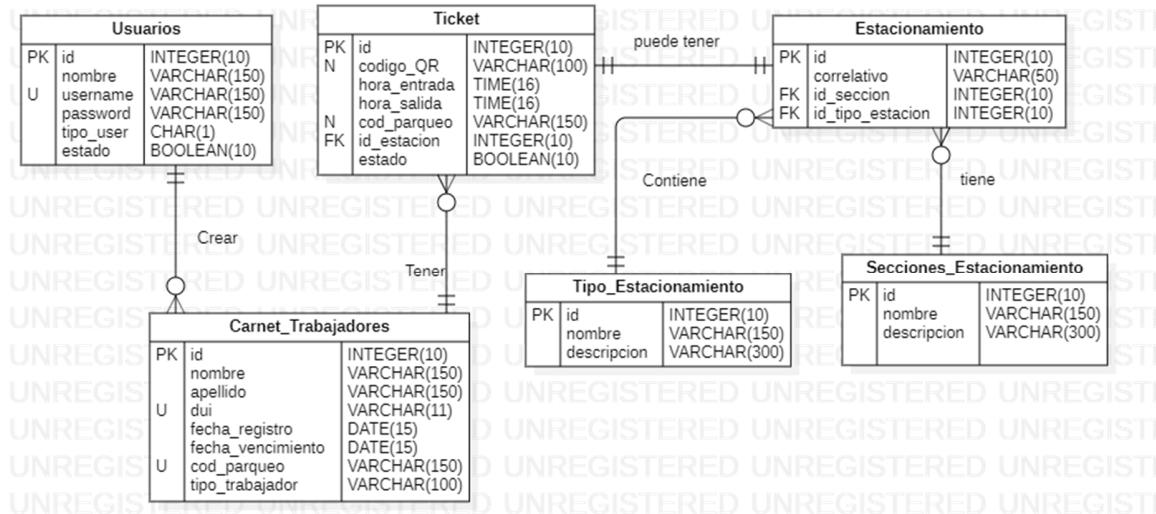


Diagrama Relacional de la Base de Datos.



Innovación aplicada en la Problemática y ¿Porque es Importante?

Los sistemas de estacionamiento actuales aplicadas en el país son útiles, pero tienen cierta desventaja como muchos productos, la cual es que no tienen un sistema de ordenamiento, y es a esta problemática que buscamos darle solución.

Los sistemas de estacionamientos que están implementados tienen un defecto, y es que no tienen un manejo de los estacionamientos disponibles.

Las innovaciones que aplicaremos en nuestro sistema, sería una forma de detectar los espacios libres mediante el uso de estructuras enlazadas, como listas, para que en ella se nos muestren los espacios disponibles, los espacios sobrantes y cuando el parqueo este lleno.

Con este sistema buscamos no solo una forma de un parqueo dinámico y organizado, sino que tambien proveer un espacio de estacionamiento para la necesidad de las personas, como por ejemplo, los trabajadores del lugar, la gente con discapacidades, mujeres embarazadas y demás.

Esto se organizaría por diferentes secciones, un espacio público para cualquiera que entre, un espacio para solo gente con discapacidades y así sucesivamente con cada caso que necesite un nuevo espacio y/o sección.

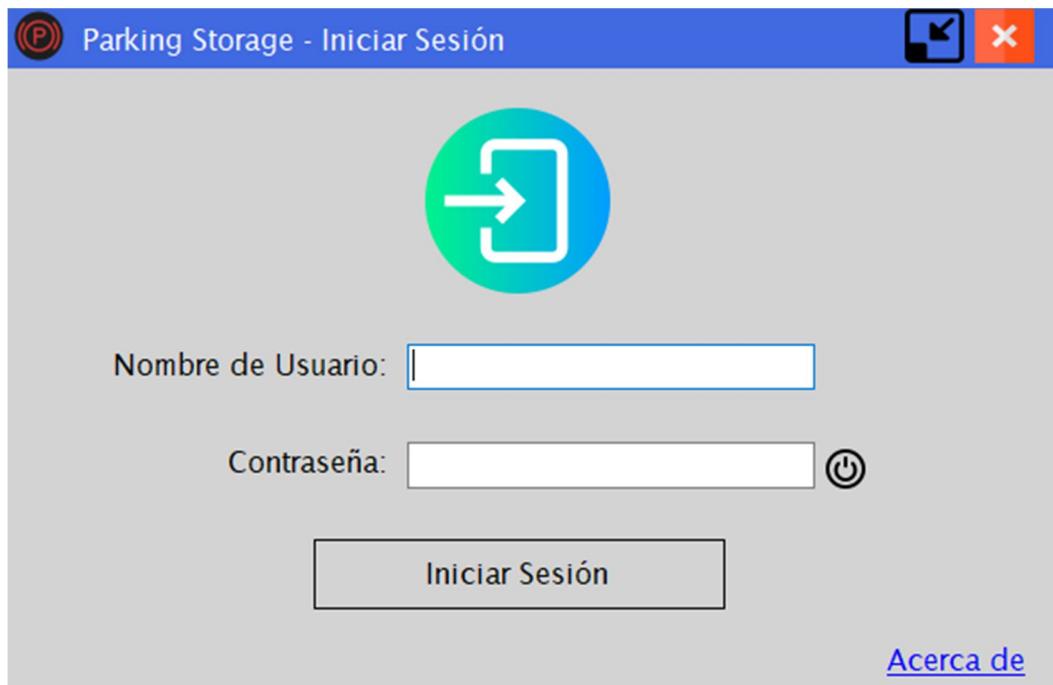
Una vez un ticket sea tomado para el estacionamiento, un número sería dado en este dónde el estacionamiento sería reservado para esta persona y se quitaría ese número al dar otro ticket, resolviendo dicha situación de que no haya problema con ir a buscar un parqueo, haciendo saber al usuario que ya tiene un lugar asignado y solo debe de ocuparlo.

Herramienta Git, este nos permitirá ejecutar comandos para controlar las versiones de nuestro proyecto, tanto en las diferentes ramas que se crearán para los desarrolladores y que puedan llevar un control de sus avances en el código del proyecto.

GitHub una plataforma para subir nuestro repositorio del proyecto que estamos desarrollando y controlar las versiones del mismo, así como el de cada rama.

Diseño del Prototipo

Login.



The login screen features a large teal circular icon with a white arrow pointing right, centered above the input fields. The title bar at the top reads "Parking Storage - Iniciar Sesión". Below the title bar are standard window control buttons (minimize, maximize, close). The main area contains two input fields: "Nombre de Usuario:" and "Contraseña:", each with a placeholder text and a clear button. A large "Iniciar Sesión" button is centered below the inputs. At the bottom right is a blue link labeled "Acerca de".

Módulo de Administrador.



The administrator panel has a dark blue sidebar on the left with icons and labels for "Master" and various management functions. The main area is titled "Sumario del Sistema" and displays five summary cards: Carnets (9), Usuarios (7), Zonas (4), Espacios (16), and Tickets (18). Below these is a card for "Tickets Expirados" (12). The top right shows the current time "10:53:28" and date "sábado, 21 de marzo de 2020". The title bar at the top reads "Parking Storage - Panel de Control".

Formulario Carnets.

Ingreso de Datos

Parking Storage - Panel de Control

Administrador

Master

- Inicio
- Carnets
- Usuarios
- Espacios
- Zonas
- Tipos de Espacios
- Tickets
- Ver Parqueo
- Cerrar Sesión

Ingresar los siguientes datos

Nombre:	<input type="text"/>
Apellido:	<input type="text"/>
Tipo de Trabajador:	Empleado
DUI:	<input type="text"/>

 Registrar

Búsqueda, Edición y Eliminación.

Parking Storage - Panel de Control

Administrador

Master

- Inicio
- Carnets
- Usuarios
- Espacios
- Zonas
- Tipos de Espacios
- Tickets
- Ver Parqueo
- Cerrar Sesión

Búsqueda por nombre, apellido o código de carnet

Nombre	Apellido	DUI	Fecha de Regis
Mario	Flores	84994989-8	15/03/2020
Joel	González	74599385-9	17/03/2020
Karen	Vásquez	69611959-9	17/03/2020
Javier	Nuñez	15493685-8	17/03/2020
David	Flores	49494948-5	18/03/2020
Daniel	Gómez	74589236-8	18/03/2020
Javier	Nuñez	13582874-8	18/03/2020
Gabriela	Humel	84691995-5	18/03/2020
Gabriel	López	49849859-8	19/03/2020

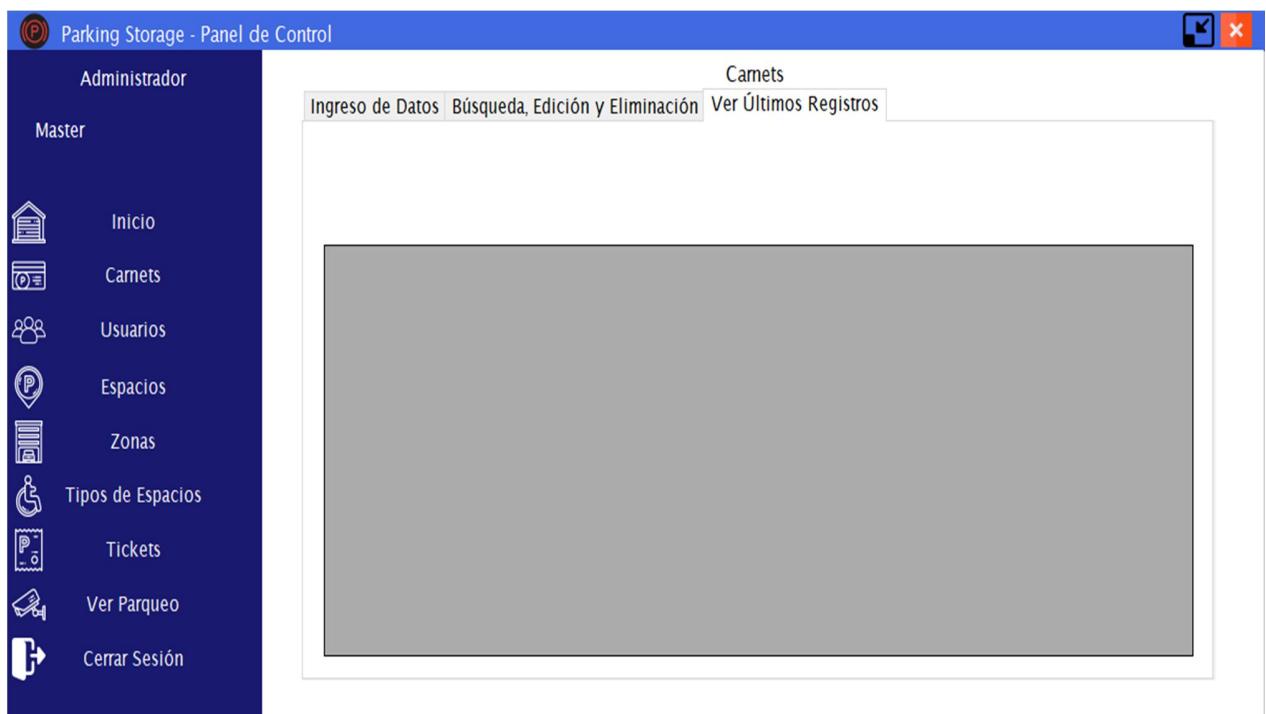
 Generar Carnet

 Buscar

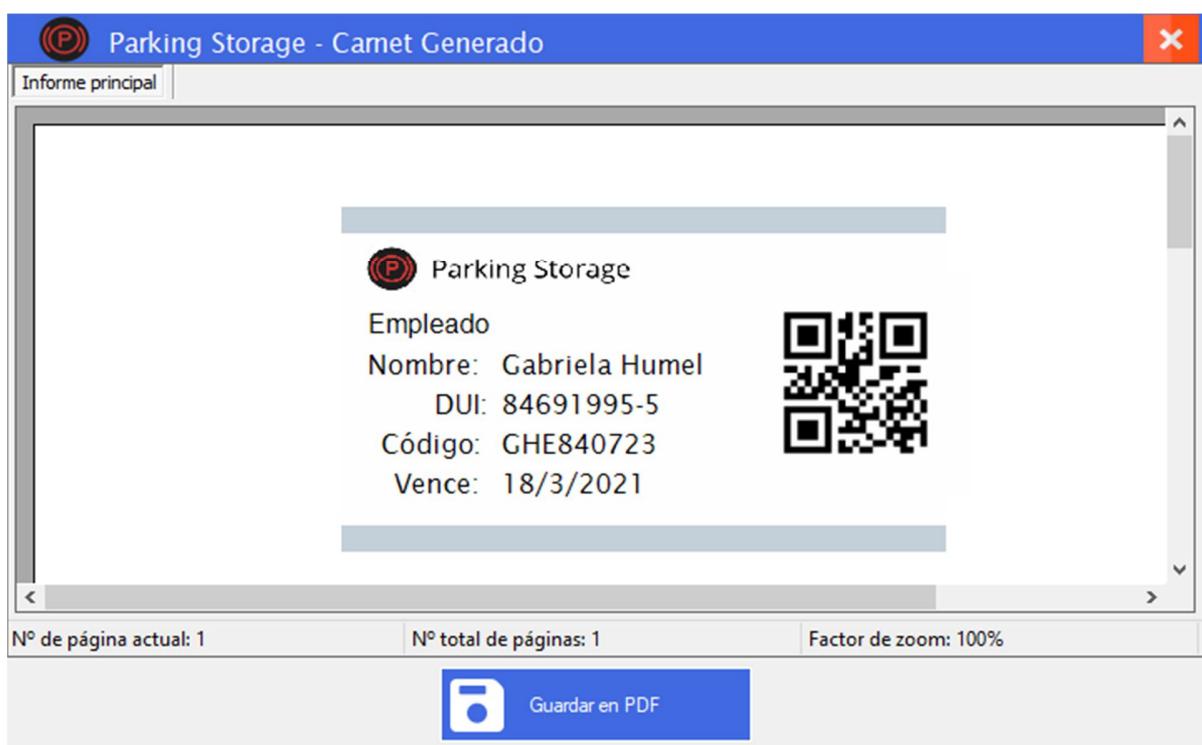
 Editar

 Estado

Ver Últimos Registros.



Carnet Generado.



Formulario Usuarios.

Parking Storage - Panel de Control

Administrador

Master

- Inicio
- Cams
- Usuarios
- Espacios
- Zonas
- Tipos de Espacios
- Tickets
- Ver Parqueo
- Cerrar Sesión

Usuarios

Ingreso de Datos Búsqueda, Edición y Eliminación Ver Últimos Registros

Ingrese los siguientes datos

Nombre:	<input type="text"/>
Nombre de Usuario:	<input type="text"/>
Tipo de Usuario:	<input type="text" value="Administrador"/>
Contraseña:	<input type="password"/>
Verificar Contraseña:	<input type="password"/>

Registrar

Formulario Espacios

Parking Storage - Panel de Control

Administrador

Master

- Inicio
- Cams
- Usuarios
- Espacios
- Zonas
- Tipos de Espacios
- Tickets
- Ver Parqueo
- Cerrar Sesión

Espacios de Estacionamiento

Ingreso de Datos Búsqueda, Edición y Eliminación Ver Últimos Registros

Ingrese los siguientes datos

Tipo de Estacionamiento:	<input type="text" value="CLIENTES"/>
Sección:	<input type="text" value="PLAZA ESTE"/>
Correlativo:	<input type="text"/>

Registrar

Parking Storage - Panel de Control

Administrador

Master

- Inicio
- Cams
- Usuarios
- Espacios
- Zonas
- Tipos de Espacios
- Tickets
- Ver Parqueo
- Cerrar Sesión

Espacios de Estacionamiento

Ingreso de Datos Búsqueda, Edición y Eliminación Ver Últimos Registros

Búsqueda por correlativo

<input type="text"/>	Buscar	Editar	Eliminar
----------------------	--------	--------	----------

	Correlativo	Sección de Estacionamiento	Tipo de Estacionamiento	Estado
>	0001P	PLAZA ESTE	CLIENTES	Ocupado
	0002P	PLAZA ESTE	CLIENTES	Ocupado
	0003P	PLAZA ESTE	CLIENTES	Ocupado
	0004P	PLAZA ESTE	CLIENTES	Ocupado
	0005P	AVENIDA OESTE	ESPECIALES	Disponible
	0006P	AVENIDA OESTE	ESPECIALES	Disponible
	0009P	PABELLÓN CENTRAL	EMPLEADOS	Disponible
	0007P	PLAZA ESTE	CLIENTES	Ocupado
	0008P	PLAZA ESTE	CLIENTES	Disponible
	0010P	PLAZA ESTE	CLIENTES	Disponible
	0012P	PLAZA ESTE	CLIENTES	Ocupado

Formulario Zonas.

Parking Storage - Panel de Control

Zonas de Estacionamiento

Ingreso de Datos Búsqueda, Edición y Eliminación Ver Últimos Registros

Ingrese los siguientes datos

Nombre:

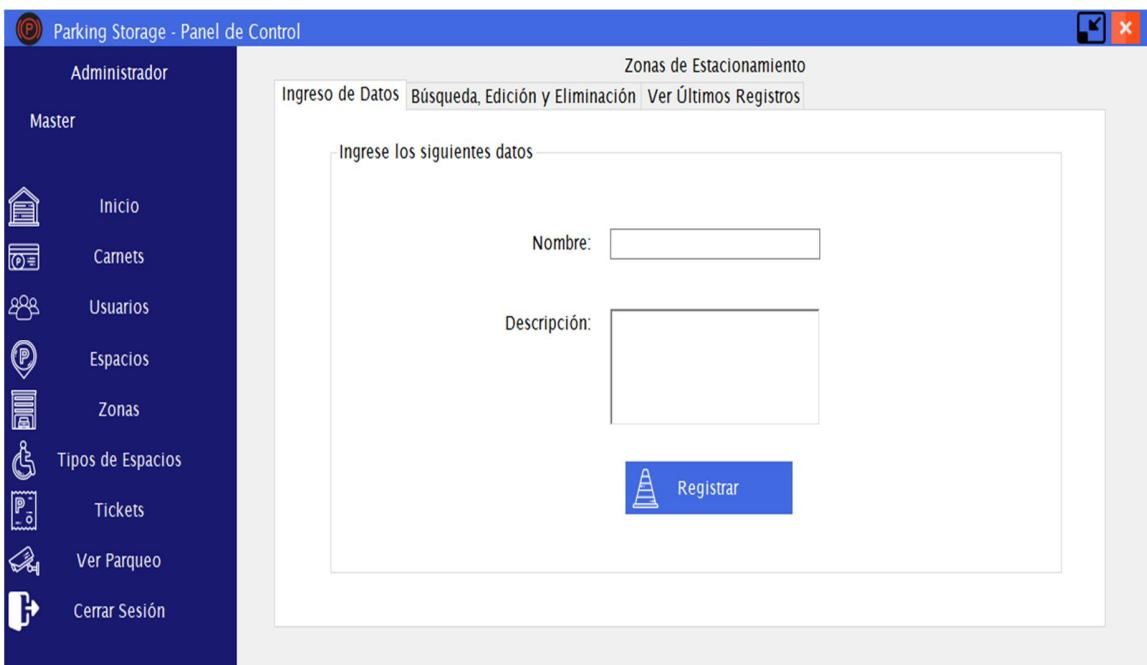
Descripción:

 Registrar

Administrador

Master

Inicio Carnets Usuarios Espacios Zonas Tipos de Espacios Tickets Ver Parqueo Cerrar Sesión



Parking Storage - Panel de Control

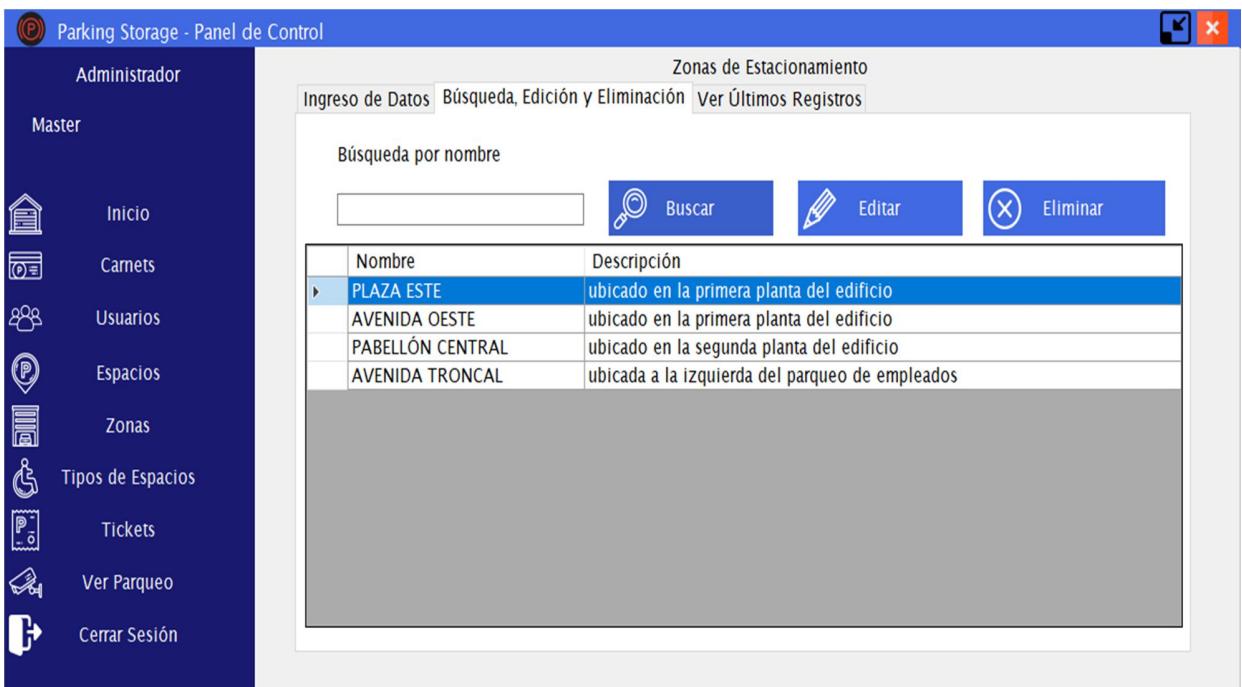
Zonas de Estacionamiento

Ingreso de Datos Búsqueda, Edición y Eliminación Ver Últimos Registros

Búsqueda por nombre

 Buscar  Editar  Eliminar

	Nombre	Descripción
▶	PLAZA ESTE	ubicado en la primera planta del edificio
	AVENIDA OESTE	ubicado en la primera planta del edificio
	PABELLÓN CENTRAL	ubicado en la segunda planta del edificio
	AVENIDA TRONCAL	ubicada a la izquierda del parqueo de empleados



Formulario Tipos de Estacionamientos.

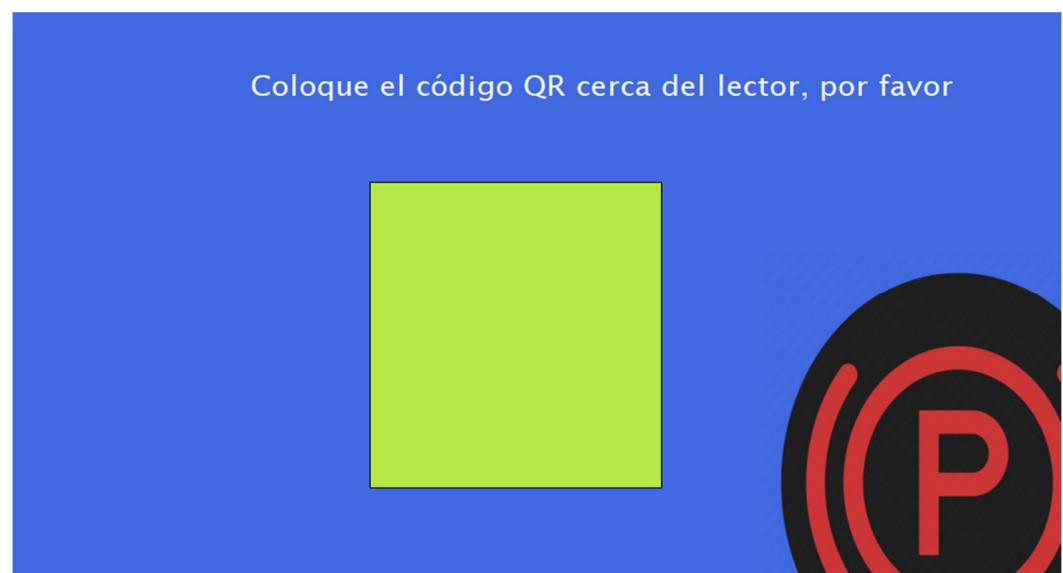
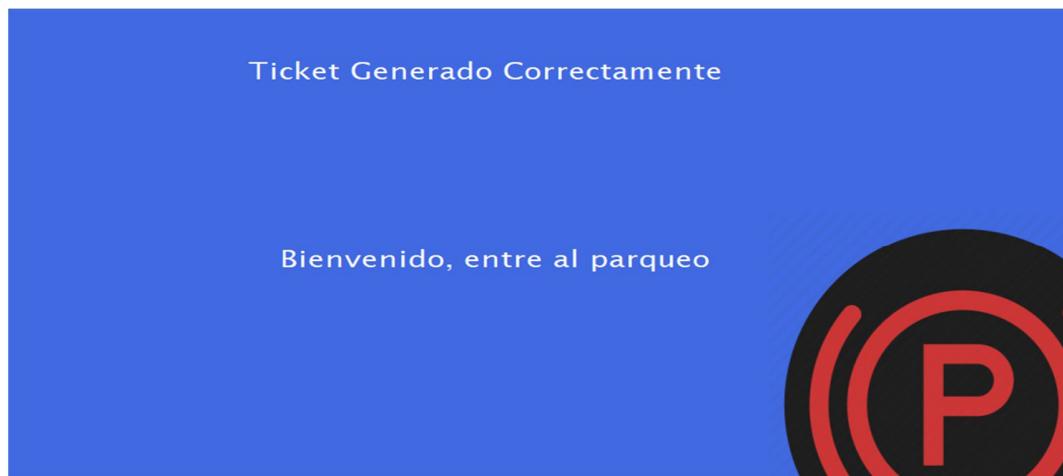
The screenshot shows a Windows application window titled "Parking Storage - Panel de Control". On the left is a dark sidebar menu with icons and labels: "Administrador", "Master", "Inicio", "Carnets", "Usuarios", "Espacios", "Zonas", "Tipos de Espacios", "Tickets", "Ver Parqueo", and "Cerrar Sesión". The main panel has a title bar "Tipos de Estacionamiento" with tabs: "Ingreso de Datos" (selected), "Búsqueda, Edición y Eliminación", and "Ver Últimos Registros". Below the tabs is a text input field "Ingrese los siguientes datos". Inside this field are two text boxes: "Nombre:" followed by a text input box, and "Descripción:" followed by a larger text input box. At the bottom right of the input field is a blue button with a person icon labeled "Registrar".

Formulario Tickets.

The screenshot shows a Windows application window titled "Parking Storage - Panel de Control". The sidebar menu is identical to the previous screenshot. The main panel has a title bar "Tickets" with a "Filtrado" section containing a dropdown menu set to "Todos" and a search bar with a magnifying glass icon labeled "Buscar". Below this is a table titled "Búsqueda por sección, tipo de estacionamiento, código de ticket y fecha". The table has columns: "Código de Ticket", "Fecha", "Hora de Entrada", and "Hora de Salida". It lists 15 rows of ticket data. The first row is highlighted in blue. The data is as follows:

Código de Ticket	Fecha	Hora de Entrada	Hora de Salida
CP46493824	15/03/2020	16:05:01	N/A
CP37121247	15/03/2020	16:24:43	N/A
CP49520143	15/03/2020	16:27:41	N/A
CP45614088	15/03/2020	17:02:20	N/A
EP86698184	15/03/2020	17:41:12	02:04:50
EA98153246	15/03/2020	17:47:14	02:32:23
CP28256606	17/03/2020	18:00:08	17:21:57
EA43647602	17/03/2020	18:02:27	02:36:17
EA62561365	17/03/2020	18:14:49	02:33:58
EP43407498	17/03/2020	18:23:29	02:31:11
CP61479232	17/03/2020	18:28:46	02:36:48
CP53369307	17/03/2020	19:46:47	02:38:23

Módulo de Entrada.



Módulo de Salida.

Coloque el código QR en el lector, por favor



Ticket Generado.

Parking Storage



Ticket
Estacionamiento

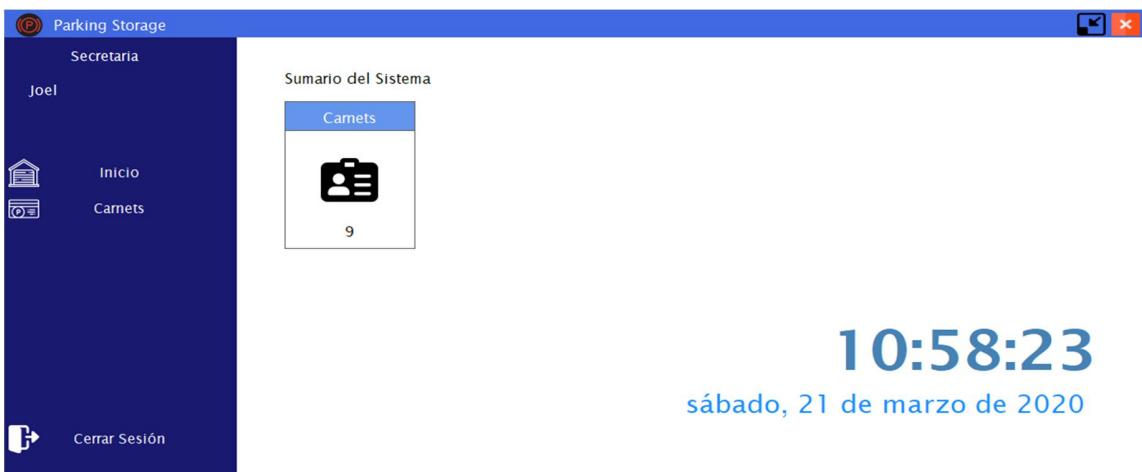
CLIENTES

Zona: PLAZA ESTE

Parqueo: 0008P



Módulo Secretaria.



Tipos de Alerta.

Advertencia.



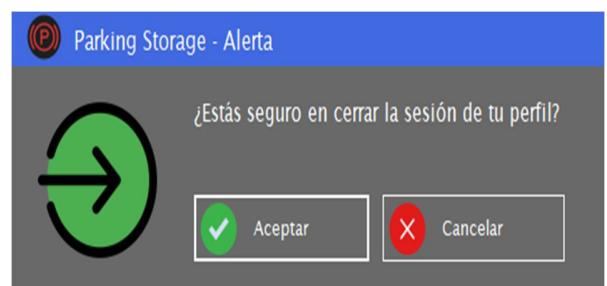
Información.



Error.

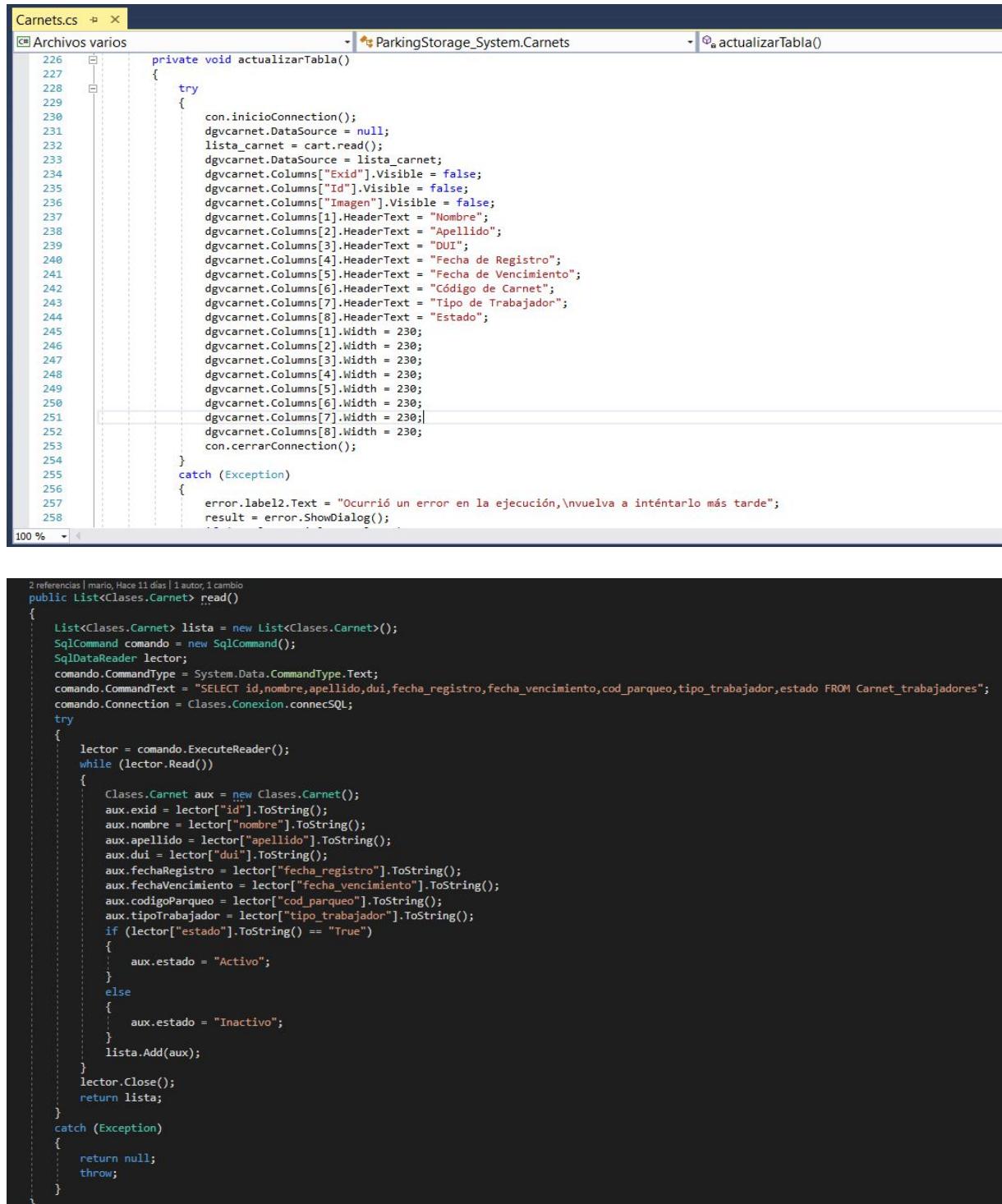


Cerrar Sesión.



Evidencia del Uso de TADS Listas.

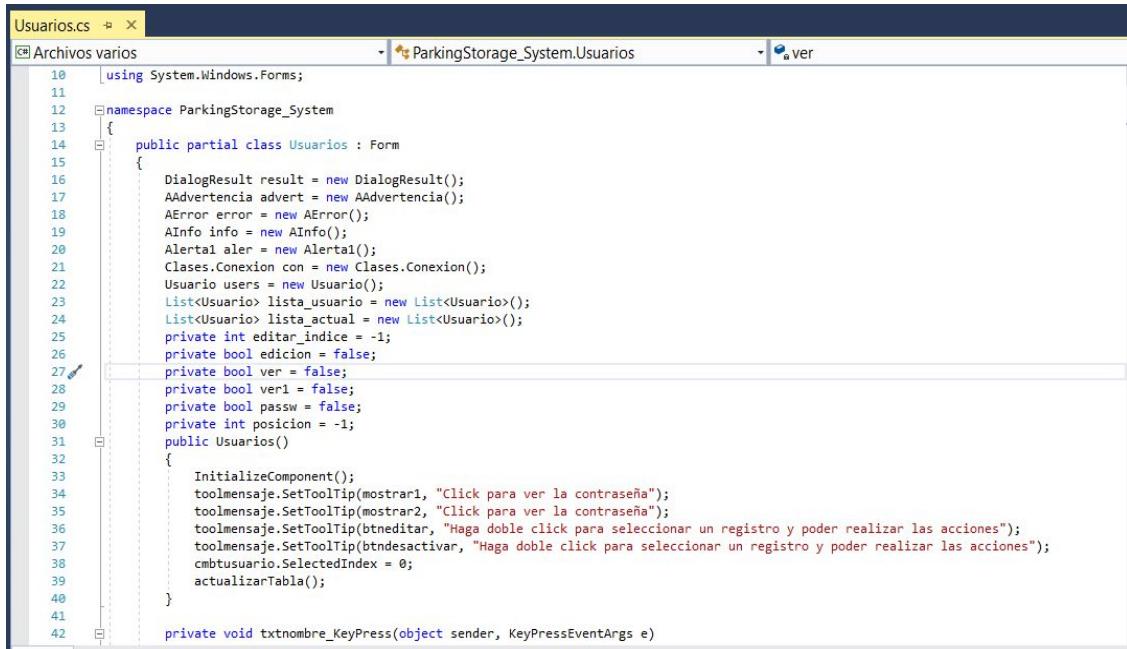
Formulario Carnets.



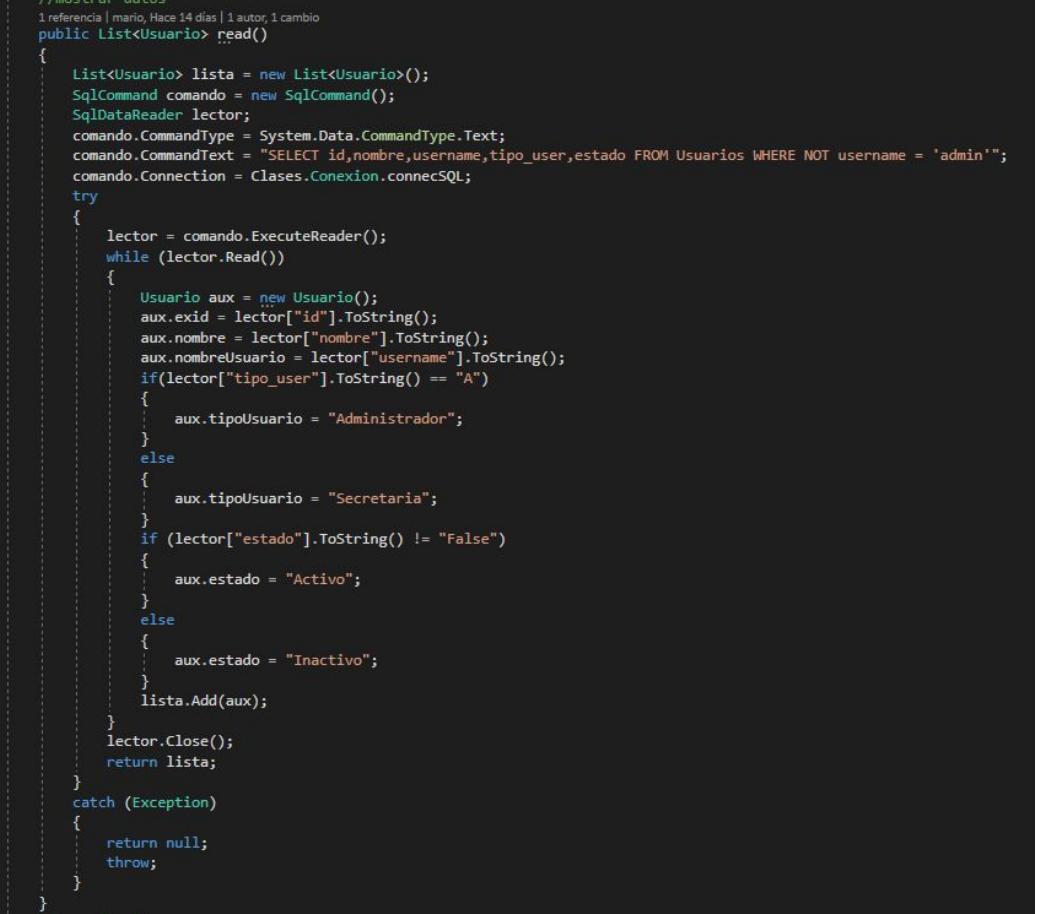
The screenshot shows two code files in a Microsoft Visual Studio environment:

- Carnets.cs**: This file contains C# code for a Windows Form named "Carnets". It includes a method named "actualizarTabla()". The code sets up a DataGridView control (dgcarnet) to display data from a database table. It configures the columns with headers like "Nombre", "Apellido", "DUI", etc., and sets widths for each column. Error handling is included to show a dialog if an exception occurs during the update process.
- Clases.Carnet.cs**: This file contains C# code for a class named "Carnet". It includes a static method "read()" which performs a SQL query to retrieve data from a table named "Carnet_trabajadores". The method uses a SqlDataReader to read the results and adds them to a list of "Carnet" objects. Each object is initialized with values from the database rows, including "estado" which is checked for "True" or "False" to determine its status.

Formulario Usuarios.

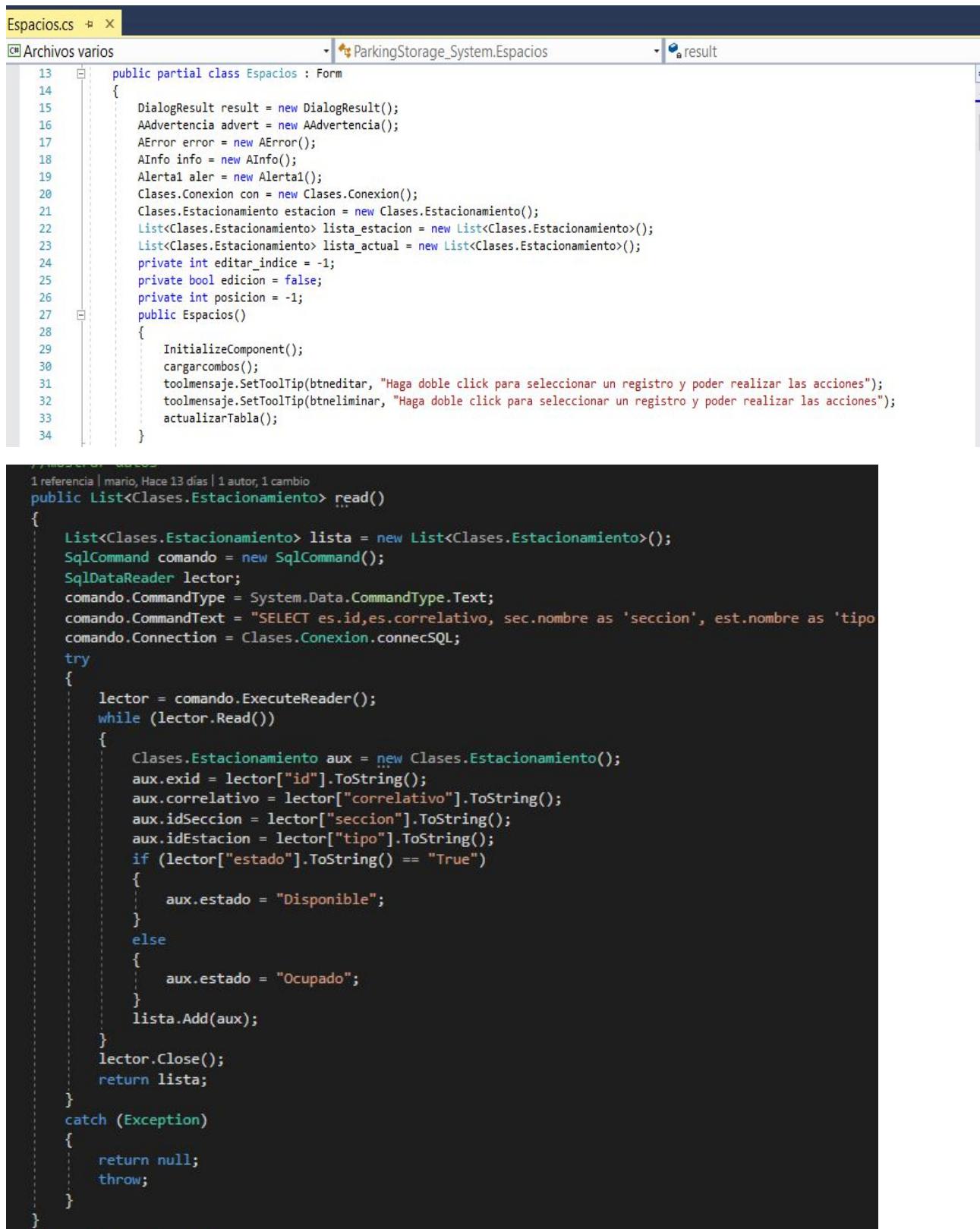


```
10  using System.Windows.Forms;
11
12  namespace ParkingStorage_System
13  {
14      public partial class Usuarios : Form
15      {
16          DialogResult result = new DialogResult();
17          AAdvertencia advert = new AAdvertencia();
18          AError error = new AError();
19          AInfo info = new AInfo();
20          Alerta1 aler = new Alerta1();
21          Clases.Conexion con = new Clases.Conexion();
22          Usuario users = new Usuario();
23          List<Usuario> lista_usuario = new List<Usuario>();
24          List<Usuario> lista_actual = new List<Usuario>();
25          private int editar_index = -1;
26          private bool edicion = false;
27          private bool ver = false;
28          private bool ver1 = false;
29          private bool passw = false;
30          private int posicion = -1;
31          public Usuarios()
32          {
33              InitializeComponent();
34              toolmensaje.SetToolTip(btnmostrar1, "Click para ver la contraseña");
35              toolmensaje.SetToolTip(btnmostrar2, "Click para ver la contraseña");
36              toolmensaje.SetToolTip(btneditar, "Haga doble click para seleccionar un registro y poder realizar las acciones");
37              toolmensaje.SetToolTip(btndesactivar, "Haga doble click para seleccionar un registro y poder realizar las acciones");
38              cmbusuario.SelectedIndex = 0;
39              actualizarTabla();
40          }
41
42          private void txtnombre_KeyPress(object sender, KeyPressEventArgs e)
```

```
//MOSTRAR DATOS
1 referencia | mario, Hace 14 días | 1 autor, 1 cambio
public List<Usuario> read()
{
    List<Usuario> lista = new List<Usuario>();
    SqlCommand comando = new SqlCommand();
    SqlDataReader lector;
    comando.CommandType = System.Data.CommandType.Text;
    comando.CommandText = "SELECT id,nombre,username,tipo_user,estado FROM Usuarios WHERE NOT username = 'admin'";
    comando.Connection = Clases.Conexion.connecSQL;
    try
    {
        lector = comando.ExecuteReader();
        while (lector.Read())
        {
            Usuario aux = new Usuario();
            aux.exid = lector["id"].ToString();
            aux.nombre = lector["nombre"].ToString();
            aux.nombreUsuario = lector["username"].ToString();
            if(lector["tipo_user"].ToString() == "A")
            {
                aux.tipoUsuario = "Administrador";
            }
            else
            {
                aux.tipoUsuario = "Secretaria";
            }
            if (lector["estado"].ToString() != "False")
            {
                aux.estado = "Activo";
            }
            else
            {
                aux.estado = "Inactivo";
            }
            lista.Add(aux);
        }
        lector.Close();
        return lista;
    }
    catch (Exception)
    {
        return null;
        throw;
    }
}
```

Formulario Espacios.



The screenshot shows the Microsoft Visual Studio IDE interface with the following details:

- Title Bar:** Shows the file name "Espacios.cs" and the project name "ParkingStorage_System.Espacios".
- Code Editor:** Displays the C# code for the "Espacios" form. The code includes declarations for various classes like AAdvertencia, AError, AInfo, Alerta1, Clases.Conexion, and Clases.Estacionamiento, along with their properties and methods. It also contains a "read()" method for retrieving data from a database using SQL commands and SqlDataReader.
- Status Bar:** Shows the commit information: "1 referencia | mario, Hace 13 días | 1 autor, 1 cambio".

```
13     public partial class Espacios : Form
14     {
15         DialogResult result = new DialogResult();
16         AAdvertencia advert = new AAdvertencia();
17         AError error = new AError();
18         AInfo info = new AInfo();
19         Alerta1 alert = new Alerta1();
20         Clases.Conexion con = new Clases.Conexion();
21         Clases.Estacionamiento estacion = new Clases.Estacionamiento();
22         List<Clases.Estacionamiento> lista_estacion = new List<Clases.Estacionamiento>();
23         List<Clases.Estacionamiento> lista_actual = new List<Clases.Estacionamiento>();
24         private int editar_index = -1;
25         private bool edicion = false;
26         private int posicion = -1;
27         public Espacios()
28     {
29         InitializeComponent();
30         cargarcombos();
31         toolmensaje.SetToolTip(btneditar, "Haga doble click para seleccionar un registro y poder realizar las acciones");
32         toolmensaje.SetToolTip(btneliminar, "Haga doble click para seleccionar un registro y poder realizar las acciones");
33         actualizarTabla();
34     }
35
36     // Implementación de los métodos y constructores restantes
37
38     /***** MÉTODOS *****/
39
40     public List<Clases.Estacionamiento> read()
41     {
42         List<Clases.Estacionamiento> lista = new List<Clases.Estacionamiento>();
43         SqlCommand comando = new SqlCommand();
44         SqlDataReader lector;
45         comando.CommandType = System.Data.CommandType.Text;
46         comando.CommandText = "SELECT es.id,es.correlativo, sec.nombre as 'seccion', est.nombre as 'tipo' FROM estacionamiento es INNER JOIN sección sec ON es.id_sección = sec.id INNER JOIN tipo est ON es.id_tipo = est.id";
47         comando.Connection = Clases.Conexion.conneSQL;
48         try
49         {
50             lector = comando.ExecuteReader();
51             while (lector.Read())
52             {
53                 Clases.Estacionamiento aux = new Clases.Estacionamiento();
54                 aux.exid = lector["id"].ToString();
55                 aux.correlativo = lector["correlativo"].ToString();
56                 aux.idSeccion = lector["seccion"].ToString();
57                 aux.idEstacion = lector["tipo"].ToString();
58                 if (lector["estado"].ToString() == "True")
59                 {
60                     aux.estado = "Disponible";
61                 }
62                 else
63                 {
64                     aux.estado = "Ocupado";
65                 }
66                 lista.Add(aux);
67             }
68             lector.Close();
69             return lista;
70         }
71         catch (Exception)
72         {
73             return null;
74             throw;
75         }
76     }
77 }
```

Formulario Zonas.

```
Zonas.cs ✘ X
Archivos varios ParkingStorage_System.Zonas result

7  using System.Text;
8  using System.Threading.Tasks;
9  using System.Windows.Forms;
10
11  namespace ParkingStorage_System
12  {
13      public partial class Zonas : Form
14      {
15          DialogResult result = new DialogResult();
16          AAdvertencia advert = new AAdvertencia();
17          AError error = new AError();
18          AInfo info = new AInfo();
19          Alerta aler = new Alerta();
20          Clases.Conexion con = new Clases.Conexion();
21          Clases.Secciones_Estacionamiento sec = new Clases.Secciones_Estacionamiento();
22          List<Clases.Secciones_Estacionamiento> lista_secciones = new List<Clases.Secciones_Estacionamiento>();
23          List<Clases.Secciones_Estacionamiento> lista_actual = new List<Clases.Secciones_Estacionamiento>();
24          private int editar_index = -1;
25          private bool edicion = false;
26          private int posicion = -1;
27          public Zonas()
28          {
29              InitializeComponent();
30              toolmensaje.SetToolTip(btneditar, "Haga doble click para seleccionar un registro y poder realizar las acciones");
31              toolmensaje.SetToolTip(btneliminar, "Haga doble click para seleccionar un registro y poder realizar las acciones");
32              actualizarTabla();
33          }
34      }

//buscar datos
2 referencias | mario, Hace 14 días | 1 autor, 1 cambio
public List<Clases.Secciones_Estacionamiento> buscar(string datos)
{
    List<Clases.Secciones_Estacionamiento> lista = new List<Clases.Secciones_Estacionamiento>();
    SqlCommand comando = new SqlCommand();
    SqlDataReader lector;
    comando.CommandType = System.Data.CommandType.Text;
    comando.CommandText = "SELECT * FROM Secciones_estacion WHERE nombre LIKE '%' + @p1 + '%'";
    comando.Connection = Clases.Conexion.conneSQL;
    try
    {
        comando.Parameters.AddWithValue("@p1", datos);
        lector = comando.ExecuteReader();
        while (lector.Read())
        {
            Clases.Secciones_Estacionamiento aux = new Clases.Secciones_Estacionamiento();
            aux.exid = lector["id"].ToString();
            aux.nombre = lector["nombre"].ToString();
            aux.descripcion = lector["descripcion"].ToString();
            lista.Add(aux);
        }
        lector.Close();
        if (lista.Count == 0)
        {
            return null;
        }
        else
        {
            return lista;
        }
    }
    catch (Exception)
    {
        return null;
        throw;
    }
}
```

Bibliografía.

- BarracodetoPC. (2020). *Barracode to Pc*. Obtenido de <https://barcodetopc.com/>
- Corel Corporation. (2020). *Gravit Designer*. Obtenido de <https://www.designer.io/es/download/>
- GNOME Fundation. (2018). *Listas*. Obtenido de <http://www.calcifer.org/documentos/librognome/glib-lists-queues.html>
- Google. (2019). *ZXing*. Obtenido de <https://opensource.google/projects/zxing>
- Marinero, J. (08 de Agosto de 2019). *Exceso de vehículos sin estacionamientos*. Obtenido de <https://elmundo.sv/exceso-de-vehiculos-sin-estacionamientos/>
- Micrisoft. (2020). *Clipboard Clase*. Obtenido de <https://docs.microsoft.com/es-es/dotnet/api/system.windows.clipboard?view=netframework-4.8>
- Microsoft. (2020). *SQL Server*. Obtenido de <https://www.microsoft.com/es-es/sql-server/sql-server-downloads>
- Microsoft. (2020). *System Generic Collection*. Obtenido de <https://docs.microsoft.com/en-us/dotnet/api/system.collections.generic?view=netframework-4.8>

Tabla de Participación.

Integrantes:	Carnets:	Porcentaje:
Beltrán García Mario Josué.	BG171969	
García Aparicio Sara Daniela.	GA190843	
Hernández Villanueva Edgar Ernesto.	HV191966	
Mejía Gámez José Ricardo.	MG190361	
Rosales Mendoza Patrick Ernesto.	RM181976	