

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

		FORMATO DE INFORME DE PRÁCTICA DE LABORATORIO / TALLERES / CENTROS DE SIMULACIÓN – PARA ESTUDIANTES	
CARRERA: Computación		ASIGNATURA: Programación Aplicada	
NRO. PRÁCTICA:		TÍTULO PRÁCTICA: Proyecto_Integrador_Final	
OBJETIVO ALCANZADO: Consolidar los conocimientos adquiridos en clase sobre Java.			
ACTIVIDADES DESARROLLADAS			
<p>1. Se pidió crear una aplicación que simule varios juegos de un casino. Para la simulación del casino se crearon seis paquetes.</p> <ul style="list-style-type: none"> • Ec.edu.ups.modelo • Ec.edu.ups.controlador • Ec.edu.ups.vista • META-INF • Ec.edu.ups.utilidades • Ec.edu.ups.imagenes 			
<p>2. Ec.edu.ups.modelo En esta clase se crearon todos los modelos que interactuaran con los controladores.</p> <ul style="list-style-type: none"> • Cliente • Parqueadero • Ticket • Usuario • Vehículo 			
<p>3. Cliente Para esta clase se utilizó el lenguaje de JPA y anotaciones para crear las columnas en la base de datos de PostgreSQL.</p> <p>Atributos:</p> <ul style="list-style-type: none"> • @Id • @Column • private String cedula; @Column • private String tipo; @Column • private String nombre; @Column • private String apellido; 			

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

- @Column
- private String direccion;
- @Column
- private String telefono;
- @OneToMany(mappedBy = "cliente", cascade = CascadeType.ALL)
- private List<Vehiculo> vehiculos;

Se crearon todos los getter y setters para los atributos.

4. Parqueadero

Para esta clase se utilizó el lenguaje de JPA y anotaciones para crear las columnas en la base de datos de PostgreSQL.

Atributos:


- @Id
- @Column
- private int puesto;
- @Column
- private boolean vacio;

5. Ticket

Para esta clase se utilizó el lenguaje de JPA y anotaciones para crear las columnas en la base de datos de PostgreSQL.

Atributos:

- private static final long serialVersionUID = 1L;
- @Id
- @GeneratedValue(strategy = GenerationType.AUTO)
- private int id;
- @Column
- private String tipo;
- @Column
- @Temporal(TemporalType.DATE)
- private Date fechaDeIngreso;
- @Column
- @Temporal(TemporalType.DATE)
- private Date fechaDeSalida;
- @Column
- private double total;
- @Column
- private int fracciones;
- @Column
- private int horas;
- @Column

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

- private String vehiculo;
@Column
- private int puesto;
@Column
- private String tiempo;

se crearon todos los getter y setter de los atributos. De la misma manera se ha creado los métodos hashCode y equals para el atributo id.

6. Usuario

Para esta clase se utilizó el lenguaje de JPA y anotaciones para crear las columnas en la base de datos de PostgreSQL.

Atributos:

- @Id
@Column
- private String cedula;
@Column
- private String nombre;
@Column
- private String apellido;
@Column
- private String correo;
@Column
- private String contrasenia;
@Column
- private String direccion;

se han creado todos los métodos getters y setters de los atributos.


7. Vehiculo

Para esta clase se utilizó el lenguaje de JPA y anotaciones para crear las columnas en la base de datos de PostgreSQL.

Atributos:

- @Id
@Column
- private String placa;
@Column
- private String marca;
@Column
- private String modelo;
@Column
- private String color;
@ManyToOne
@JoinColumn(name = "fk_persona")
- private Cliente cliente;

se han creado todos los métodos getters y setters de los atributos.

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

8. **Ec.edu.ups.utilidades**

Este paquete solamente contiene la clase utils para poder conectar la base de datos con el controlador.

- Utilidades

9. **Utilidades**

Esta clase realiza la tarea de conseguir el Entity Manager.

Atributos:

- `private static final EntityManagerFactory emf = Persistence.createEntityManagerFactory("ParqueaderoJPAPU");`

métodos:

- `getEntityManager`: Este método solamente realiza la tarea de devolver el entity manager.

10. **Ec.edu.ups.imagenes**

Este paquete realiza la tarea de guardar las imágenes que se utilizaran para la interfaz.

11. **META_INF**

Este paquete contiene una clase de tipo persistence.xml.

12. **Ec.edu.ups.controlador**

En este paquete se han creado todas las clases para que interactúen con la interfaz del usuario

- `controladorCliente`
- `ControladorGenerico`
- `ControladorParqueadero`
- `controladorRegex`
- `controladorTicket`
- `controladorUsuario`
- `controladorVehiculo`

13. **ControladorGenerico**

Esta clase es una clase abstracta, de la misma manera esta clase es una clase genérica que permite que otras clases herede sus métodos y a la misma vez


Atributos:

- `private Class<T> clase;`
- `private EntityManager em;`

esta clase solamente tiene un controlador la cual inicializa todos sus atributos y realiza la conexión a la base de datos. En esta clase también se han creado sus métodos getters y setters para todos sus atributos.

métodos:

- `Créate`: Este método recibe un objeto en su parámetro y crea el objeto dentro de la base de datos, al final realiza un commit.

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

- Update: Este método recibe un objeto en su parámetro y este realiza un merge en la base de datos para poder actualizar el objeto dentro de la base de datos.
- Read: Este método recibe un objeto id dentro de su base de datos. Este método realiza la tarea de devolver un objeto de la base de datos y realiza la búsqueda por su primary key.
- Delete: Este método recibe un objeto dentro de su base de datos. Este método realiza la tarea de eliminar un objeto dentro de la base de datos.
- findAll: Este método realiza la tarea de devolver un listado de objetos que exista dentro de la base de datos.

14. ControladorCliente

Este método hereda del controlador genérico, de la misma manera se reutiliza todos sus métodos.

15. ControladorParqueadero

Este controlador hereda del controlador genérico, se reutilizan todos los métodos de la clase padre.

Atributos:

- private Parqueadero parqueadero;

Métodos:

- Liberar: Este método recibe un objeto de tipo Integer en su parámetro. Este método realiza la tarea de reservar un parqueadero y por último manda a actualizar el parqueadero dentro de la base de datos.
- Reservar: Este método recibe un objeto de tipo Integer en su parámetro. Este método realiza la tarea de liberar un parqueadero y por último manda a actualizar el parqueadero dentro de la base de datos.

16. ControladorRegex

Esta clase se utiliza para poder crear patrones y poder autenticar los datos de ingreso.

Atributos:

- private Pattern patron;
- private Matcher corpus;

metodos:

- ingresoRegex: Este método recibe un objeto de tipo String en su parámetro. Este método realiza la tarea de ingresar un patrón e incorporarlo dentro de los atributos.
- validar: Este método recibe un objeto de tipo String dentro de su parámetro. Este método valida si el String recibido en su parámetro cumple con las condiciones del patrón.

En esta clase se han creado los métodos getters y setters de los atributos.


17. ControladorTicket

Este controlador hereda del controlador genérico, se reutilizan todos los métodos de la clase padre.

Atributos:

- private Ticket ticketInterno;
- private ControladorParqueadero controladorp;

se ha creado un constructor para este controlador en el cual solo se inicializa el controlador para evitar que la base de datos se encuentre con un punto nulo.

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

Métodos:

- IngresarParqueadero: Este método recibe un objeto de tipo Ticket dentro de su parámetro. Este método crea el ticket dentro de la base de datos y llama al controlador parqueadero para que reserve ese puesto del ticket.
- salidaParqueadero: Este método recibe un objeto de tipo Ticket dentro de su parámetro. Este método actualiza el ticket dentro de la base de datos y llama al controlador parqueadero para que libere el puesto del ticket.
- calcularPago: Este método recibe un objeto de tipo integer dentro de su parámetro. Este método solamente realiza la tarea de devolver un ticket con los datos de pago.
- calcularTotal: Este método realiza la tarea de calcular la forma de pago dado a el tipo de ticket que es.
- ticketPorCobrar: Este método retorna una lista de tickets que falta por cobrar.
- ticketCobrados: Este método realiza la tarea de devolver una lista de tickets que ya están pagados.

18. ControladorUsuario

Este controlador hereda del controlador genérico, se reutilizan todos los métodos de la clase padre.

Método:

- Validar: este método recibe dos objetos de tipo String. Este método realiza la tarea de validar si es que un usuario existe dentro de la base de datos. en caso de que si existe el sistema retorna un boolean verdadero, caso contrario retorna un boolean falso.

19. ControladorVehiculo

Este controlador hereda del controlador genérico, se reutilizan todos los métodos de la clase padre.

Método:

- vehiculosDeCliente: Este método recibe un objeto de tipo String dentro de su parámetro. Este método realiza la tarea de devolver un listado de vehículos bajo la cedula de un cliente.

20. Ec.edu.ups.vista

Este paquete contiene todas las interfaces y hilos para que interactué el usuario.


- HiloAuto
- IniciarSession
- Inicio
- MenuPrincipal
- PanelAuto
- RegistrarAdministrador
- RegistrarCliente
- RegistrarVehiculo
- ReservarParqueadero
- Tipo
- VentanaAdministrador

21. HiloAuto

Esta clase hereda de la clase Thread y se utiliza para simular el movimiento del vehículo a su parqueadero.

Atributos:

- private int x;
- private int y;
- private boolean continuar;

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

- private JPanel panel;
- private JLabel label;

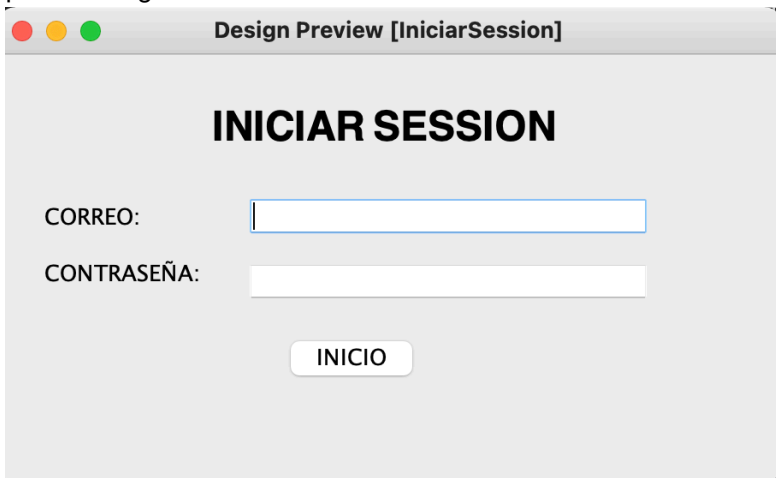
El constructor de esta clase recibe un JPanel y un JLabel dentro de su parámetro y inicializa los atributos:

Metodo:

- Run: Este método recibe las coordenadas del label y realiza la tarea de mover la imagen cada 25 milisegundos hasta llegar a las coordenadas del label.


22. iniciarSession

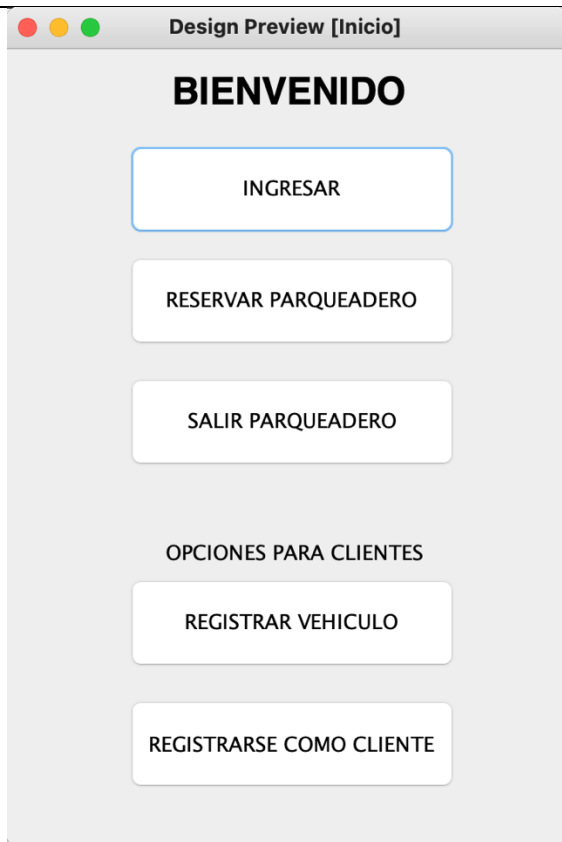
Esta interfaz permite que el usuario ingrese su correo y contraseña, en caso de que no exista el usuario dentro de la base de datos el programa no le habilitara la ventana de administrador, caso contrario si le permitirá ingresar.



23. inicio


Esta ventana se visualizara al inicio del programa, este menú permite que un cliente ingrese al parqueadero y permita que se registre.

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		



24. MenuPrincipal

Esta ventana contiene todas las interfaces, permitiendo visualizar todas las interfaces que va a utilizar el usuario.

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

MENU ADMINISTRADOR




25. RegistrarAdminsitrador

Esta ventana permite que un administrador se pueda registrar dentro del sistema.



26. RegistrarCliente

Esta interfaz permite que el usuario se pueda registrar como cliente y guardar sus datos dentro de la base de datos.

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		



Design Preview [RegistrarCliente]

CLIENTE

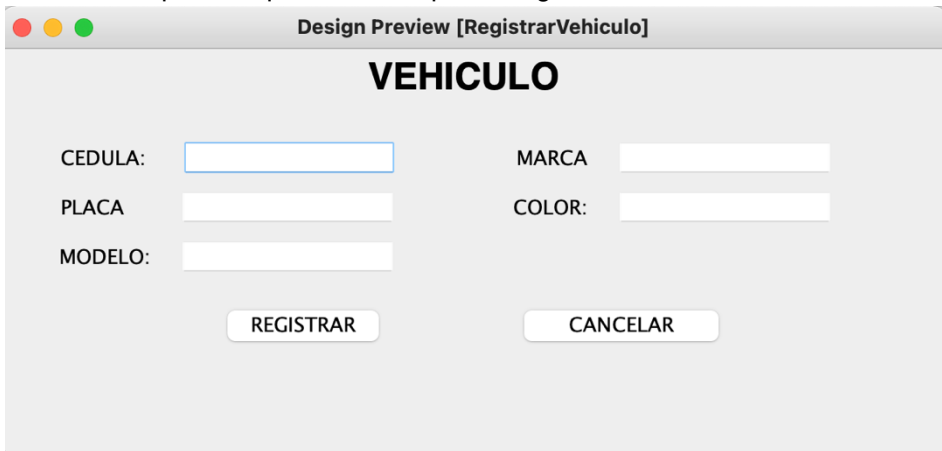
CEDULA: DIRECCION:

NOMBRE: TELEFONO:

APELLIDO:

27. RegistrarVehiculo

Esta interfaz permite que un cliente pueda registrar su vehículo dentro del sistema



Design Preview [RegistrarVehiculo]

VEHICULO

CEDULA: MARCA:

PLACA: COLOR:

MODELO:

28. Tipo

Esta ventana sirve como una ventana intermediaria para determinar si un usuario es cliente o no.



Design Preview [Tipo]

TIPO

29. VentanaAdministrador

Esta ventana permite que un administrador pueda emitir diversos reportes.

Design Preview [VentanaAdministrador]

ADMINISTRADOR

# TICKET	FECHA DE INGRESO	FECHA DE SALIDA	TOTAL	PLACA	LOTE
----------	------------------	-----------------	-------	-------	------

CREAR ESPACIOS DE PARQUEADEROS:

CREAR

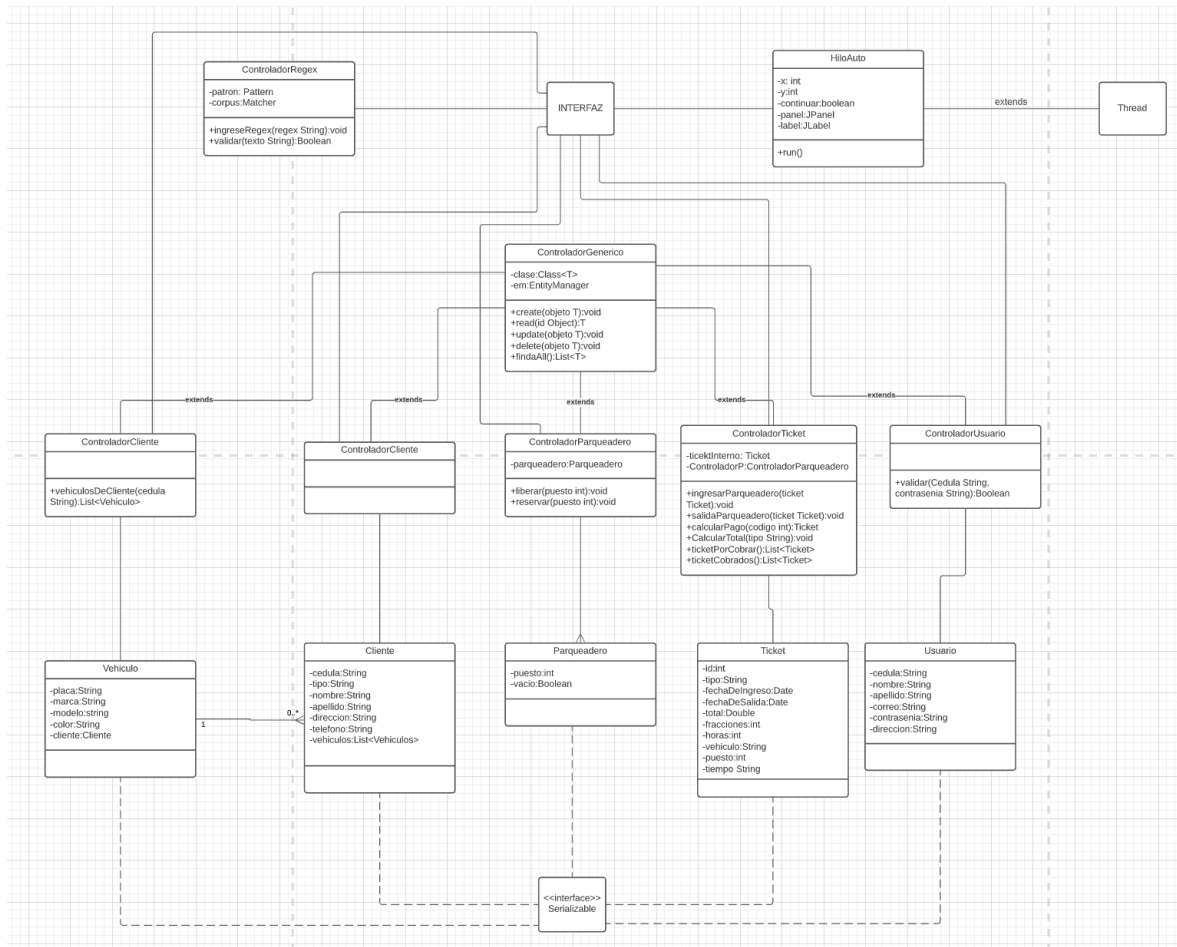
TODOS TICKETS

LISTA TICKETS POR PAGAR

IMPRIMIR

LISTA TICKETS PAGADOS

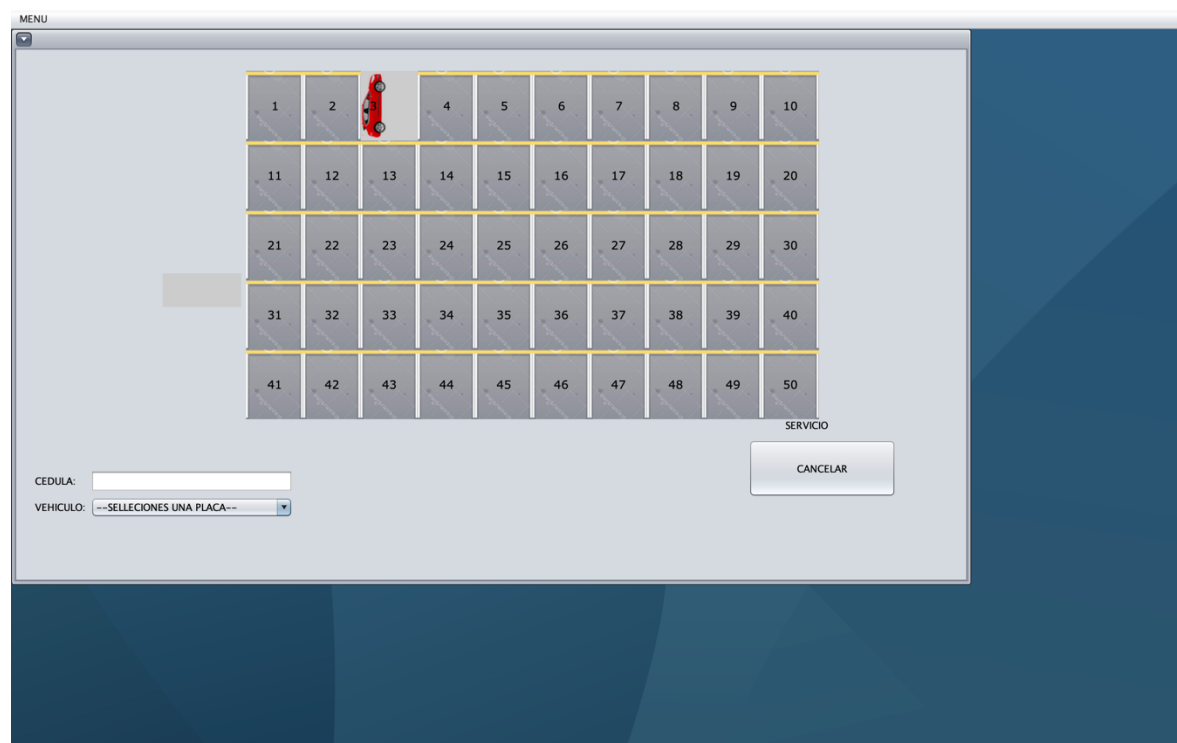
30. UML



31. Link de presentación

<https://prezi.com/view/oV8hrEShH21OXz6xcCbu/>

RESULTADO(S) OBTENIDO(S):



CONCLUSIONES:

En conclusión, se debe tener un entendimiento de como utilizar las ultimas actualizaciones de java, debemos tener un entendimiento de como utilizar expresiones regex, o regulares. Un elemento fundamental que se debe tener en cuenta es el manejo de la base de datos postgresSQL y como utilizar el lenguaje de programación JPQL para poder realizar consultas y realizar tablas y columnas dentro de la base de datos.

RECOMENDACIONES:

Nombre de estudiante: Denys Dutan

Firma de estudiante: 