



Los Santos Customs

**Hugo Sánchez Barroso, Roberto Velázquez Vázquez,
Antonio Muñoz García**

1 DAM
IES Torre de los Guzmanes

ÍNDICE

1. Introducción.....	2
2. Metodología usada.....	3
3. Diagrama de Gantt.....	4
4. Diagrama UML.....	5
5. Modelo Entidad/Relación.....	8
6. Base de datos relacional.....	10
7. Nuestro GitHub.....	11
8. Trello.....	12
9. Actas de ceremonias.....	13
10. Conexión de Base de datos a Eclipse.....	14
11. PDF explicativo Base de datos relacional.....	19
12. Creación de Menú en Java.....	21
13. Presentación.....	23

1. Introducción.

El proyecto que presentamos se centra en la implementación de la metodología ágil Scrum para llevar a cabo la distribución eficiente de concesionarios de coches. Es fundamental contar con una red de concesionarios estratégicamente distribuida para garantizar una cobertura óptima y una experiencia de compra satisfactoria.

Este proyecto propone aplicar las siguientes técnicas para distribuir el trabajo a realizar:

- Diagrama de Gantt.
- Diagrama UML.
- Modelo Entidad/Relación.

Todo esto será realizado por un **equipo** el cual estará compuesto por miembros multidisciplinarios, incluyendo un Scrum Master, un Development Team, y dos Product Owners. Cada miembro aportará sus habilidades únicas para colaborar en la definición, diseño y ejecución de la estrategia de distribución de concesionarios.

A lo largo de este proyecto, nos comprometemos a mantener una **comunicación** transparente y continua con todas las partes interesadas, incluyendo los **propietarios** de los concesionarios, y los **clientes** finales. De esta manera, aseguraremos que el proceso de distribución se adecue a las expectativas y necesidades del mercado.

Mediante la aplicación de **Scrum**, estamos seguros de que lograremos una distribución efectiva de concesionarios de coches que maximice la accesibilidad, la eficiencia operativa y la **satisfacción del cliente** en el mercado automovilístico actual y futuro.

2. Metodología usada.

Como trabajadores de Los Santos Customs, decidimos utilizar la metodología de trabajo **SCRUM** ya que pensamos que era la más óptima y la que mejor se adapta a nuestro proyecto. En cuanto a los diferentes roles asignados en este proyecto, nos encontramos:

- **SCRUM MASTER:** Hugo Sánchez Barroso es nuestro **Scrum Master**. Se encarga de liderar y servir a los diferentes equipos y/o trabajadores del proyecto. Además, soluciona y ayuda en los impedimentos del equipo.
- **DEVELOPMENT TEAM:** Antonio Muñoz García y Roberto Velázquez Vázquez son parte esencial del **equipo de desarrollo**. Desarrollan el producto pedido por el cliente para satisfacer sus necesidades.
- **PRODUCT OWNER:** Jesús Doña Calvo y Javier Prada Oliva son nuestros **product owners**. Ellos nos realizan consultas y peticiones acerca del producto final que desean.

3. Diagrama de Gantt.

La adopción de la metodología Scrum para el proyecto "Los Santos Customs" es fundamental para su gestión eficaz, y el diagrama de Gantt sirve como una representación visual de todas las tareas a seguir para ir completando los sprints del proyecto de forma satisfactoria.. Dividido en cuatro sprints de una semana cada uno, el diagrama muestra claramente cómo se planifica y ejecuta el trabajo en incrementos cortos y enfocados. Esta estructura iterativa permite una rápida adaptación a los cambios y una entrega continua de valor a lo largo del proyecto.

El proyecto se inicia con una fase de planificación donde se establecen los objetivos específicos y se priorizan las tareas del backlog del producto. Esta fase se refleja en el diagrama de Gantt con la asignación de roles, la organización del proyecto y la elaboración del backlog, destacando la importancia de una sólida preparación antes de cada iteración.

Durante el transcurso de cada sprint, el equipo se centra en la ejecución de las tareas prioritarias, como el desarrollo de funcionalidades, la resolución de problemas y las pruebas de calidad. Estas actividades se representan en el diagrama de Gantt con fechas de inicio y finalización específicas, lo que permite un seguimiento claro del progreso del trabajo a lo largo del tiempo.

El diagrama de Gantt no solo proporciona una visión general del cronograma del proyecto, sino que también refleja la metodología ágil Scrum adoptada. Al mostrar claramente la planificación y ejecución de cada sprint, así como la entrega continua de valor al cliente, el diagrama de Gantt se convierte en una herramienta esencial para la gestión efectiva del proyecto.

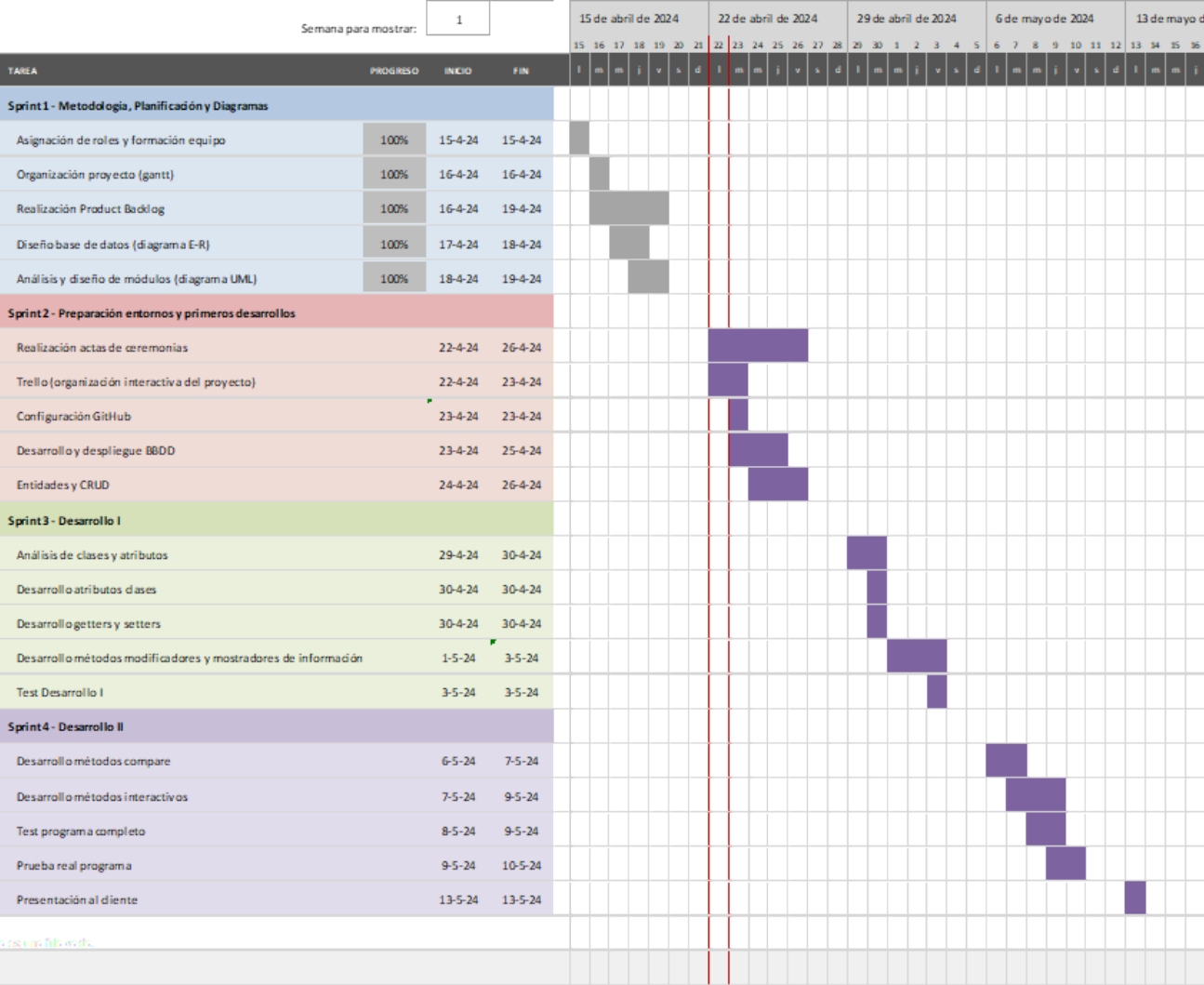
Los Santos Customs - Concesionario

Inicio del proyecto: 15 de abril de 2024

Fin del proyecto: 13 de mayo de 2024

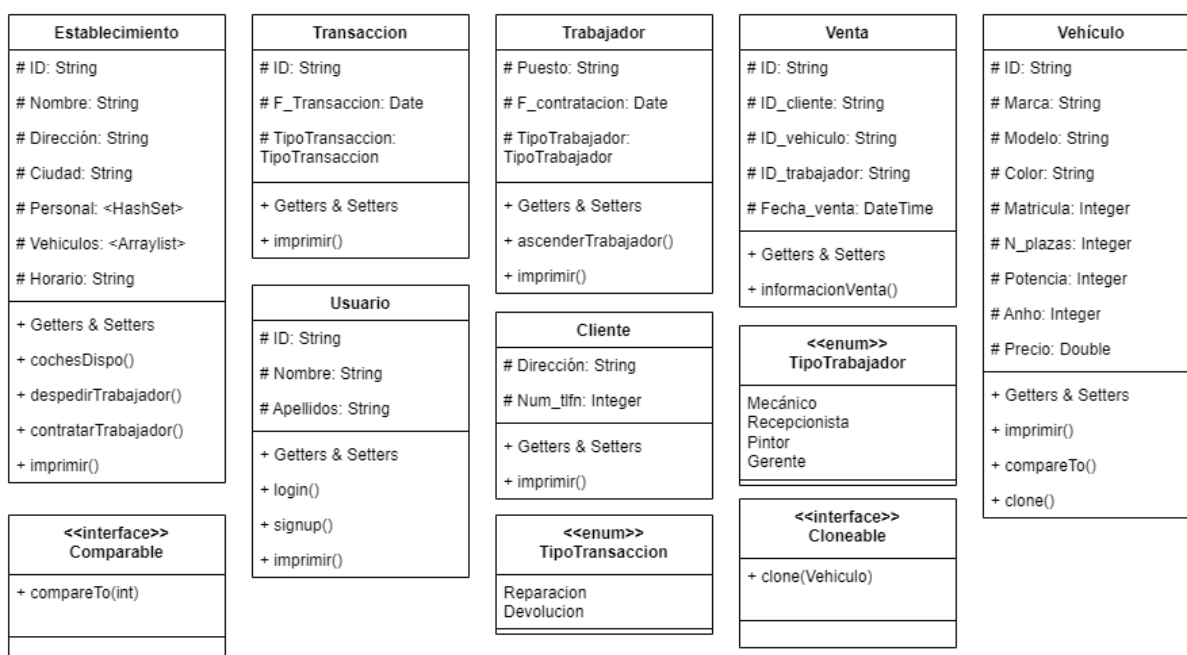
Inicio del proyecto: 15 de abril de 2024

Semana para mostrar: 1



4. Diagrama UML.

A la hora de definir las diferentes partes que formarán parte en el proyecto a nivel usuario, la forma más óptima de dividir las es recrear el ámbito laboral en un **diagrama UML**. De esta manera, conseguimos ver de forma concisa y detallada la estructura de nuestro negocio.



Como podemos observar, nuestro diagrama se divide en 5 clases diferentes. En primer lugar tenemos la clase **Vehículo**, la cual se compone de varios atributos, todos ellos protegidos para una mayor comodidad en su trabajo a la hora de desarrollar. Además contiene varios métodos bastante interesantes como por ejemplo clone, compareTo o imprimir.

Le sigue la clase **Trabajador**, la cual también contiene atributos protegidos, y varios métodos como imprimir o ascenderTrabajador. Además, este está complementado por el **enum TipoTrabajador**.

Establecimiento es una de las clases más importantes en nuestro proyecto, ya que todas las demás clases están relacionadas con esta.

A continuación tenemos la clase **venta**, la cual es otro pilar fundamental en nuestro proyecto a la hora de guardar información. Es importante destacar que esta clase será la encargada de guardar los datos de los partícipes de cada venta.

También tenemos la clase **cliente**. Esta clase contiene varios atributos protegidos y el método propio de imprimir.

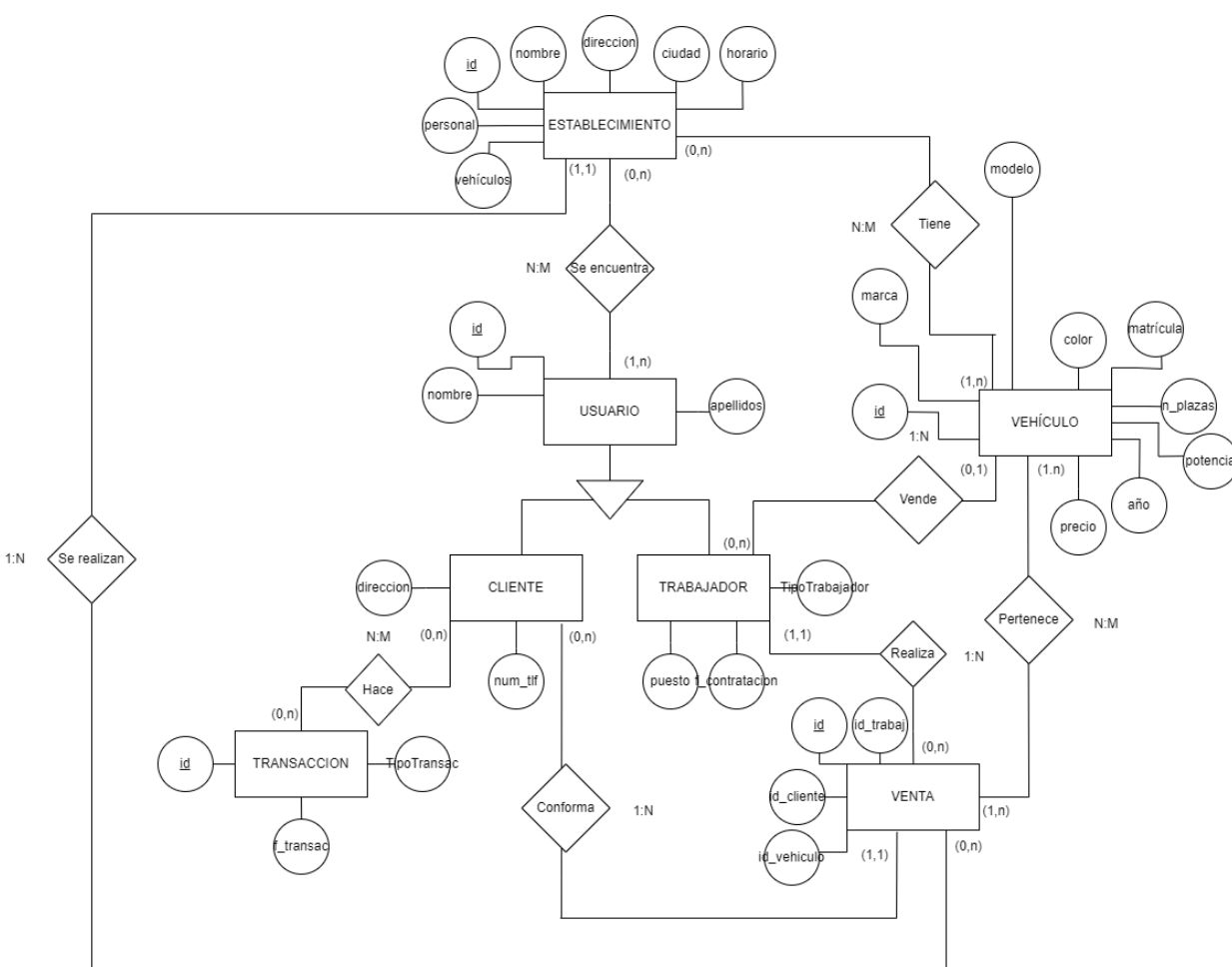
Le sigue la clase **Transaccion**, la cual junto con un **enum con los tipos de Transacciones** guardará cada transacción junto con un ID, una fecha y el tipo de transacción que se trata.

Por último hablar de las diferentes interfaces utilizadas, como **Comparable** y **Clonable**, las cuales serán muy útiles a la hora de desarrollar código.

Cabe destacar que hemos decidido establecer un **atributo ID** en cada clase ya que a la hora de consultar y organizar la información en una base de datos, nos será más óptimo y cómodo.

5. Modelo Entidad/Relación.

Con el objetivo de construir las diferentes relaciones que formarán parte en el desarrollo de las entidades, deberemos de realizar un **Modelo Entidad / Relación**. De esta manera, conseguiremos definir cómo interactúan entre sí las entidades.

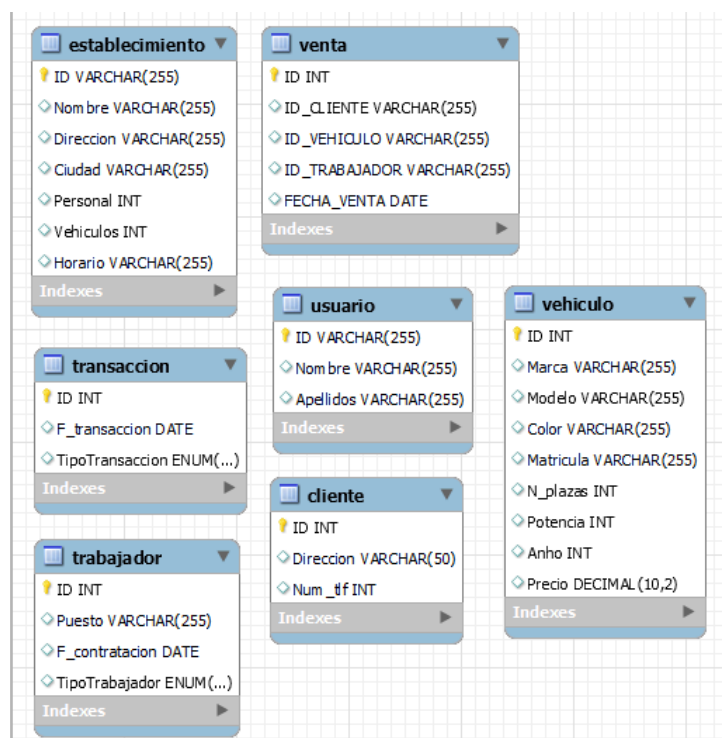


El Modelo de Entidad/Relación cuenta con un total de 7 entidades, las cuales son los siguientes:

- **Establecimiento**, el cuál será el propio Concesionario y se encuentra el usuario; en un establecimiento se encuentra uno o varios usuarios, y un usuario se puede encontrar en un establecimiento, pero puede que no. A continuación, en un establecimiento se realizan muchas ventas, pero puede ser que no realice ninguna, y una **venta** es realizada en un único concesionario. Por último, un concesionario tiene muchos vehículos, y un **vehículo** lo tiene muchos concesionarios, pero puede ser que no se encuentre en ninguno.
- Un **usuario** puede ser cliente o trabajador, los cuales heredan las propiedades de su clase Padre.
- Un **cliente** puede no hacer ninguna transacción, o hacer varias. A su misma vez, una Transacción puede ser hecha por muchos Clientes, o ninguno. Por último, un cliente conforma una sola **venta**, y una venta está conformada por muchos clientes, pero puede ser que ninguno conforme una venta.
- Un **trabajador** puede realizar ninguna venta, pero también podría realizar muchas. A su vez, una venta solo puede ser hecha por un solo Trabajador.
- Además, un **trabajador** puede no vender ningún **vehículo**, pero también puede vender muchos. Por su contraparte, un vehículo puede no ser vendido por ningún trabajador, sin embargo también puede ser vendido muchas veces.
- Un **vehículo** pertenece en mínimo a una venta, pero puede pertenecer a varias ventas. Una venta, a su misma vez, es perteneciente a uno o muchos vehículos.

6. Base de datos relacional.

Una vez creados tanto los diagramas como la planificación de nuestra aplicación, crearemos la base de datos y realizaremos algunos inserts para comprobar el funcionamiento de esta. La estructura básica será:



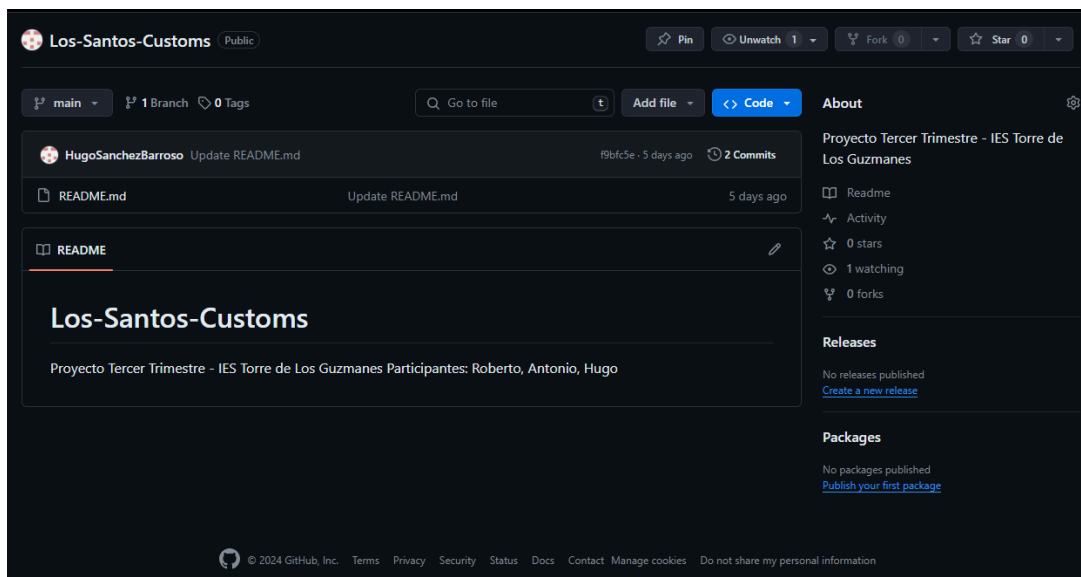
Como se puede observar, tendremos las tablas **establecimiento**, **transaccion**, **cliente**, **usuario**, **venta**, **vehiculo**, y **trabajo**. Cada tabla contiene sus propios atributos, además de tener en tablas como trabajador y transaccion los cuales atributos tipo **ENUM** como TipoTrabajador y TipoTransaccion.

Archivo .sql Base de datos:

[LosSantosCustoms-version_final.sql](#)

7. Nuestro GitHub.

A la hora de desarrollar nuestro proyecto, es conveniente que utilicemos algún software para almacenar, supervisar y trabajar de forma colaborativa. En nuestro caso vamos a usar la plataforma GitHub. En nuestro repositorio, previamente creado, subiremos las nuevas versiones modificadas y actualizadas de nuestro proyecto.

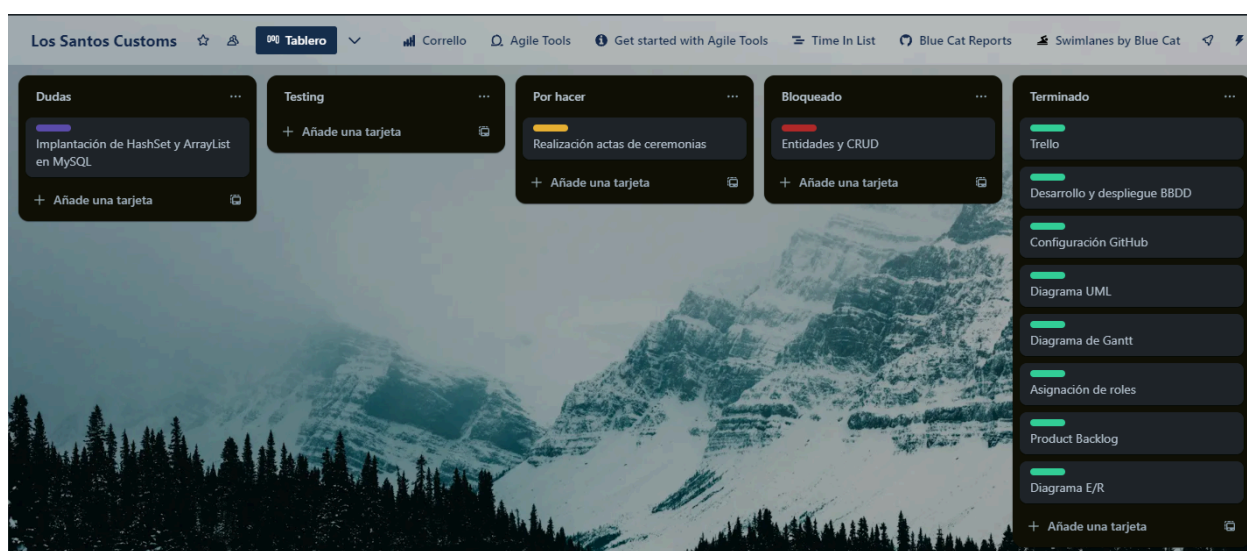


Enlace al repositorio de GitHub:

<https://github.com/HugoSanchezBarroso/Los-Santos-Customs>

8. Trello.

En cuanto a organización se trata, hemos decidido utilizar Trello. Trello es una herramienta visual que permite a los equipos gestionar cualquier tipo de proyecto y flujo de trabajo, así como supervisar tareas. Aquí nos administramos las tareas en tiempo real.



En este diagrama separamos las actividades en **Dudas**, ya que va a ser la base de nuestras actividades del apartado **Bloqueado**. También tenemos las actividades **Testing**, las cuales serán las actividades que están de prueba. En tercer lugar tenemos las actividades **Por hacer** donde estarán las actividades que aún no hemos terminado. Por último tenemos el apartado de actividades **Terminadas**.

Enlace a Trello:


<https://trello.com/invite/b/YaYJnFXq/ATT1b57e3749142ea011e21e99fa3ca5c1d19850D69C/los-santos-customs>

9. Actas de ceremonias.

Un **acta de ceremonias** es un documento formal que registra los procedimientos, decisiones y resultados de reuniones específicas relacionadas con el desarrollo del proyecto. Estas reuniones pueden incluir la revisión de avances, la planificación de tareas, la resolución de problemas y cualquier otra actividad relevante para el proyecto.

En nuestro caso, nos basamos en esta metodología que implementa **SCRUM** para organizar en un solo documento las diferentes reuniones que se han realizado a lo largo del proyecto. Estas incluyen tanto los temas discutidos como los participantes de cada reunión.

Enlace al Registro de las Actas de Ceremonias:

 [Actas de ceremonias - Los Santos Customs](#)

10. Conexión de Base de datos a Eclipse.

A la hora de trabajar con una base de datos relacional, decidimos integrar **phpMyAdmin** en **Eclipse**. Esto nos permite administrar la base de datos desde nuestro IDE. Podremos crear, consultar, modificar y eliminar tablas, además de gestionar usuarios y privilegios.

Cabe destacar que al tener la base de datos previamente creada en **MySQL**, sólo tuvimos que importarla a phpMyAdmin para obtener las respectivas tablas:

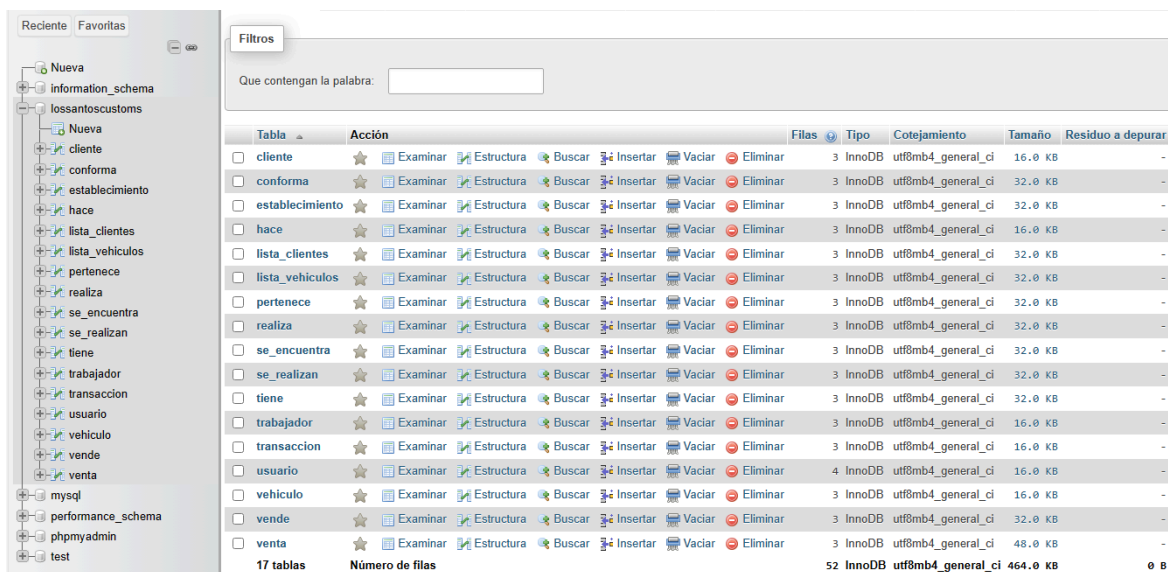


Tabla	Acción	Filas	Tipo	Cotejamiento	Tamaño	Residuo a depurar
cliente	Examinar Estructura Buscar Insertar Vaciar Eliminar	3	InnoDB	utf8mb4_general_ci	16.0 KB	-
conforma	Examinar Estructura Buscar Insertar Vaciar Eliminar	3	InnoDB	utf8mb4_general_ci	32.0 KB	-
establecimiento	Examinar Estructura Buscar Insertar Vaciar Eliminar	3	InnoDB	utf8mb4_general_ci	32.0 KB	-
hace	Examinar Estructura Buscar Insertar Vaciar Eliminar	3	InnoDB	utf8mb4_general_ci	16.0 KB	-
lista_clientes	Examinar Estructura Buscar Insertar Vaciar Eliminar	3	InnoDB	utf8mb4_general_ci	32.0 KB	-
lista_vehiculos	Examinar Estructura Buscar Insertar Vaciar Eliminar	3	InnoDB	utf8mb4_general_ci	32.0 KB	-
pertenece	Examinar Estructura Buscar Insertar Vaciar Eliminar	3	InnoDB	utf8mb4_general_ci	32.0 KB	-
realiza	Examinar Estructura Buscar Insertar Vaciar Eliminar	3	InnoDB	utf8mb4_general_ci	32.0 KB	-
se_encuentra	Examinar Estructura Buscar Insertar Vaciar Eliminar	3	InnoDB	utf8mb4_general_ci	32.0 KB	-
se_realizan	Examinar Estructura Buscar Insertar Vaciar Eliminar	3	InnoDB	utf8mb4_general_ci	32.0 KB	-
tiene	Examinar Estructura Buscar Insertar Vaciar Eliminar	3	InnoDB	utf8mb4_general_ci	32.0 KB	-
trabajador	Examinar Estructura Buscar Insertar Vaciar Eliminar	3	InnoDB	utf8mb4_general_ci	16.0 KB	-
transaccion	Examinar Estructura Buscar Insertar Vaciar Eliminar	3	InnoDB	utf8mb4_general_ci	16.0 KB	-
usuario	Examinar Estructura Buscar Insertar Vaciar Eliminar	4	InnoDB	utf8mb4_general_ci	16.0 KB	-
vehiculo	Examinar Estructura Buscar Insertar Vaciar Eliminar	3	InnoDB	utf8mb4_general_ci	16.0 KB	-
vende	Examinar Estructura Buscar Insertar Vaciar Eliminar	3	InnoDB	utf8mb4_general_ci	32.0 KB	-
venta	Examinar Estructura Buscar Insertar Vaciar Eliminar	3	InnoDB	utf8mb4_general_ci	48.0 KB	-
17 tablas	Número de filas	52	InnoDB	utf8mb4_general_ci	464.0 KB	0 B

Al tener la tabla en phpMyAdmin, realizaremos operaciones **CRUD** para corroborar que la conexión de nuestra base de datos con nuestro IDE es correcta.

Para realizar dichas pruebas **CRUD**, crearemos diferentes funciones las cuales se encuentran en la **clase principal** (main) del proyecto en **Eclipse**.

Dichas funciones son **insertarNomTabla()**, **consultarNomTabla()**, **actualizarNomTabla()** y **eliminarNomTabla()**, siendo "NomTabla" el nombre de la tabla que queremos utilizar.

A continuación mostramos la estructura de las funciones para la tabla Establecimiento. Cabe destacar que todas las funciones tienen una estructura muy similar, pero varían campos de cada tablas seleccionada.

Función insertarEstablecimiento()

Para introducir nuevos datos en nuestra base de datos, llamaremos a esta función junto con los valores que queremos añadir.

```
public static void insertarEstablecimiento(String ID, String Nombre, String
Direccion, String Ciudad, String Horario, String ID_vehiculo) {
    try {
        String url = "jdbc:mysql://localhost:3306/lossantoscustoms";
        Connection conn = DriverManager.getConnection(url, "root", "");
        Statement st = conn.createStatement();
        st.executeUpdate("INSERT INTO Establecimiento " + "VALUES
(""+ID+"", ""+Nombre+"", ""+Direccion+"", ""+Ciudad+"", ""+Horario+"", ""+ID_vehiculo+"")");
        conn.close();
    } catch (Exception e) {
        System.err.println("Got an exception! ");
        System.err.println(e.getMessage());
    }
}
```


Función consultarEstablecimiento()

A la hora de consultar y ver si los datos se han introducido de manera correcta, llamaremos a la función de consulta previamente creada.

```
public static void consultarEstablecimiento () {
    try (Connection connection =
DriverManager.getConnection("jdbc:mysql://localhost:3306/lossantoscustoms",
"root", "");
        Statement st = connection.createStatement();
        ResultSet resultSet = st.executeQuery("SELECT * FROM
Establecimiento")) {
        while (resultSet.next()) {
            String ID = resultSet.getString("ID");
            String Nombre = resultSet.getString("Nombre");
            String Direccion = resultSet.getString("Direccion");
            String Ciudad = resultSet.getString("Ciudad");
            String Horario = resultSet.getString("Horario");
            String ID_vehiculo = resultSet.getString("ID_vehiculo");
            System.out.println(ID + "\t" + Nombre + "\t" + Direccion +
"\t" + Ciudad + "\t" + Horario + "\t" + ID_vehiculo);
        }
        resultSet.close();
        st.close();
        connection.close();
    } catch (SQLException e) {
        System.out.println("Error en la conexión de la base de datos.");
        e.printStackTrace();
    }
}
```

Función actualizarEstablecimiento()

En el caso de que nuestro objetivo sea actualizar algún valor de la tabla, utilizaremos esta función.

```
public static void actualizarEstablecimiento (String campo, String nuevoValor) {

    try {
        String url =
"jdbc:mysql://localhost:3306/lossantoscustoms";
        Connection conn =
DriverManager.getConnection(url,"root","");
        String query = "update Establecimiento set " + campo + "=?
where ID='E001'";

        PreparedStatement ps = conn.prepareStatement(query);
        ps.setString(1, nuevoValor);
        ps.executeUpdate();
        System.out.println("Actualización realizada
correctamente.");
    }
    catch (Exception e) {
        e.printStackTrace();
    }
}
```

Si todo funciona correctamente podremos observar los cambios, tal y como se muestra a continuación:

```
22 //Actualizar la tabla Establecimiento - campo Ciudad de Ciudad A a Birmingham
23 actualizarEstablecimiento("Ciudad", "Birmingham");
24
25 //Tercera consulta a la BD.
26 consultarEstablecimiento();
27
28 //Eliminar la tabla insertada
29 //eliminarEstablecimiento();
30
31 //Cuarta consulta a la BD.
32 //consultarEstablecimiento();
33 }
34
35 //-- CRUD de Establecimiento
```

Problems Javadoc Declaration Console X

<terminated> Establecimiento [Java Application] C:\Users\Administrador\AppData\Local\Temp\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_17.0.11.v20240426-1830\jre\bin\javaw.exe (5 may 2024 21:3)

Actualización realizada correctamente.

E001	AutoStore	Calle Principal 123	Birmingham	Lunes a Viernes 9am-6pm	V001
E002	CarShop Avenida	Central 456	Ciudad B	Lunes a Sábado 10am-8pm	V002
E003	MotoWorld	Plaza Central 789	Ciudad C	Lunes a Domingo 8am-10pm	V003
E004	Gasoil City	Calle Sánchez Cotán s/n	Kuala Lumpur	Viernes a Domingo 10am-22pm	V004

Función eliminarEstablecimiento()

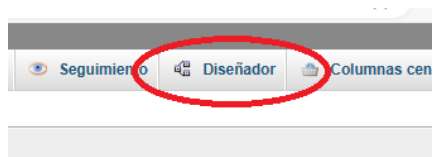
Si lo que queremos es eliminar una fila específica de nuestra tabla, utilizaremos esta función la cual define en su interior el campo el cual busquemos.

```
public static void eliminarEstablecimiento() {
    try {
        String url = "jdbc:mysql://localhost:3306/lossantoscustoms";
        Connection con = DriverManager.getConnection(url,"root","");
        String SQL = "DELETE FROM Establecimiento WHERE ID = 'E004' ";
        PreparedStatement pstmt = con.prepareStatement(SQL);
        pstmt.executeUpdate();
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

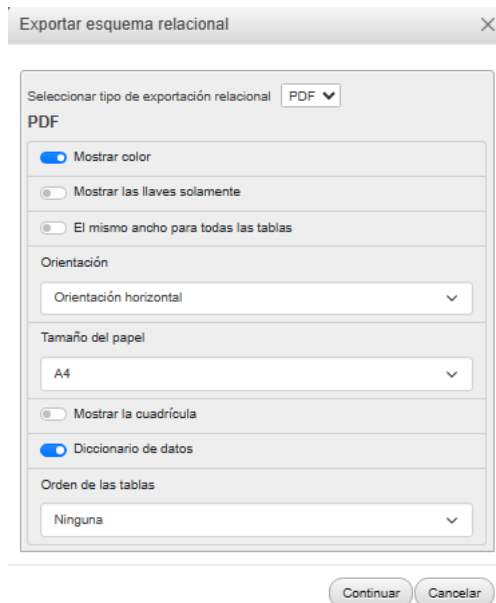
11.PDF explicativo Base de datos relacional.

Una vez realizada la **importación** de nuestra base de datos, seleccionamos en phpMyAdmin que nos genere un **PDF informativo** de las tablas que tenemos en la Base de Datos.

En el menú de la base de datos donde se nos muestran todas las tablas, veremos una pestaña llamada “**diseñador**”:




Al entrar en este apartado veremos un menú a la derecha de nuestra estructura de tablas. Seleccionamos en la opción de “**Exportar estructura**”. Una vez seleccionada esta opción, nos aparecerá el siguiente menú:



En este punto, podremos seleccionar en qué tipo de archivo queremos exportar la estructura de nuestra base de datos. En nuestro caso lo exportamos como PDF. Una vez seleccionada, pulsamos **Continuar**, ya que los otros valores nos interesan dejarlos tal y como están.

Resultado adjunto de la exportación:

 Estructura-LSC.pdf

Este archivo puede ser útil a la hora de revisar la estructura final de las tablas, incluyendo las relaciones que tienen entre sí. Además de poder observar el contenido detallado de cada una de las tablas.

12.Creación de Menú en Java.

Una gran empresa como es Los Santos Customs necesita ofrecer al cliente la mejor experiencia posible. A la hora de interactuar con la base de datos desde nuestro proyecto, previamente creado, en Eclipse, se puede hacer de diversas maneras. En nuestro caso optamos por introducir un menú muy intuitivo para el uso del cliente.

En este menú se diferencian las siguientes funcionalidades:

- **Insertar nuevos datos:**

El usuario tendrá la posibilidad de añadir un nuevo registro en cualquiera de las tablas existentes. El menú le pedirá, uno por uno, todos los datos necesarios para la agregación de un nuevo registro, y si el usuario los introduce correctamente, este se agregará exitosamente a la tabla de la base de datos.

- **Mostrar datos existentes:**

En segundo lugar se le proporcionará al usuario la opción de mostrar los registros que existen en cualquier tabla seleccionada de la base de datos. Primero el usuario deberá seleccionar la tabla que quiere que el sistema le muestre. Posteriormente, si la tabla existe, el sistema le mostrará la información que contiene dicha tabla.

- **Eliminar datos:**

A continuación, le ofreceremos al usuario eliminar registros de las tablas. En esta opción el usuario deberá de indicar de qué tabla desea eliminar el registro, y la ID a la que hace referencia dicho registro.

Aquí una breve descripción de cómo quedaría el menú desde la visión del cliente:

```
¡Bienvenido a Los Santos Customs!
```

```
-----  
                        MENÚ  
-----
```

- ```
1. Insertar uno o varios registros.
2. Mostrar todos los registros.
3. Eliminar un registro.
4. Salir

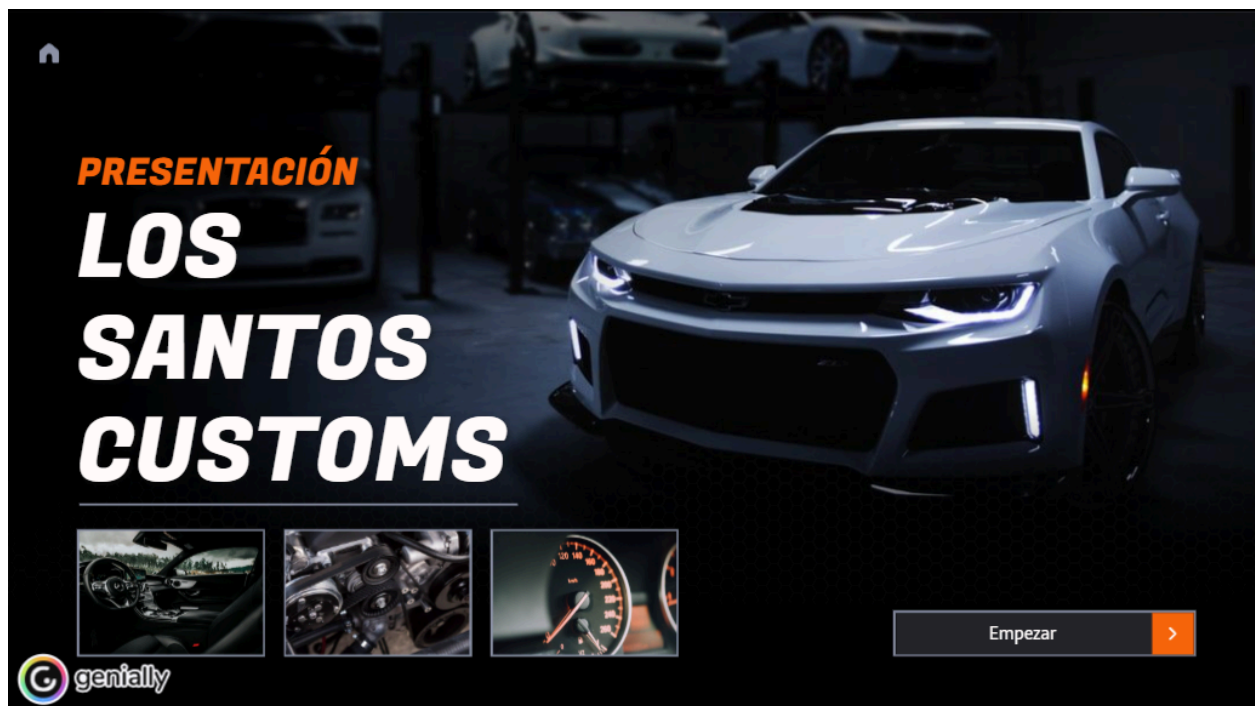
```

```
Introduzca una opción:
|
```

---

## 13. Presentación.

Para sorpresa de nadie, hemos realizado una presentación de nuestro proyecto. En esta, se explica de qué trata nuestra empresa, la metodología utilizada, los problemas albergados en la elaboración del mismo, y una breve demostración de la interfaz en Java que interacciona con la base de datos.



<https://view.genial.ly/664255612030e9001414cb10/presentation-presentacion-lsc>