



***GENIUS DRIVER***

## CURSO

DAM2A 2016-2017

## AUTORES

Abdel Ali el Jaouhari  
Wilson Philip  
Javier Montes Carrasco

## AGRADECIMIENTOS

*“Agradecimientos a todas aquellas personas que, sin ánimo de lucro, de alguna manera u otra ofrecieron parte de su valioso tiempo para ayudarnos a hacer realidad esta idea. Este proyecto contiene una pequeña parte de todos aquellos internautas que se dedican a la programación en cualquiera de sus lenguajes. En especial a un grupo de compañeros que sin sus consejos no hubiera sido lo mismo: Albert Ortells, Arnau Infante, Emil-Jay y Jean Pierre.*

*Y un fuerte abrazo Gabriel González que durante una larga noche nos explicó a grandes rasgos cómo implementar el envío de datos entre lenguajes.”*

## ABSTRACT

Decidimos involucrarnos en este proyecto en particular porque nos pareció una idea original y novedosa que no habíamos visto en anteriores propuestas. Apuntamos alto para triunfar en el gran mercado con una propuesta insólita.

Otro de los motivos en los que pensamos es que podría ser la base de una conducción autónoma y más segura, ayudando a las personas que tienen un impedimento de movilidad.

Por otra parte, sería un avance poder utilizar este tipo de tecnología en el servicio de transporte público el cual siempre debe de estar disponible para cualquier persona independientemente de la hora que sea. Además, aquellos usuarios que también quieran disponer de estas comodidades (particulares) podrán disfrutar de los modelos comerciales que tenemos a la venta.

En la obtención de los resultados necesarios para corroborar que son precisos y fiables hemos utilizado una tecnología de simulación, donde podemos reproducir cualquier tipo de escenario que puede suceder en la vida real. De esta forma, confirmamos que los cálculos y métodos empleados son los correctos para que no se pueda producir un accidente por parte de la tecnología implantada por Genius Driver.

Si comparásemos nuestro proyecto con otros del mercado podemos decir con orgullo que ayudamos al medio ambiente empleando vehículos eléctricos reduciendo las emisiones de CO<sub>2</sub> tan dañinas para el planeta y logrando por otra parte disminuir los accidentes de tráfico.

El mundo necesita un cambio y cuanto antes lo aceptemos menos consecuencias negativas habrán en el futuro. Somos la alternativa del mañana

We have decided to work in this project because we thought that it was a good idea.

We have been proposed to work in the creation on this idea because we saw that it was original and we could succeed in the labour market.

Other clearly reasons are the ideas of Autonomous cars and efficiency at Driving. This project helps people that are not able or have difficulties to travel with the car.

From Genius Driver we thought that our idea it could be introduce in the public services and/or particulars.

To get more results we have been using a simulations technology which has allowed us to do some tests that the car need. Doing this we verify if the results are those we want to achieve, at the same time we prevent accidents and save money .

If we compare our project with others in the labour market we can say in our favour that help the environment using electric cars reducing CO2 pollution emissions and at the same time we support the use of public transportation.

This world needs a change, and as soon as we can do things, future will be better. We have to maintain it, it is unique and at the same time we have to progress.

## ÍNDICE

AGRADECIMIENTOS.....	1
ABSTRACT .....	2
WEBGRAFIA.....	5
DIAGRAMA UML.....	6
DIAGRAMA ENTIDAD-RELACIÓN (CHEN).....	7
REPOSITORIO WEB .....	13
DISEÑO APLICACIÓN MOVIL.....	14
DISEÑO APLICACIÓN MOVIL.....	18
START UP/START DOWN .....	29
EXPLICACION DE LOS ELEMENTOS INVESTIGADOS.....	31
WEBGRAFIA.....	65
¿QUE APORTA EL TRABAJO HECHO?.....	67
OBJETIVOS.....	69
OBJETIVOS ALCANZADOS .....	70
AMPLIACIONES.....	71
ASPECTOS NEGATIVOS .....	72

## WEBGRAFIA

Listado de las herramientas con su correspondiente versión utilizada para realizar el proyecto descrito:

Windows NT Genius-Genius 6.1 BUILD 7601 (Windows 7 Ultimate Edition SP 1)

XAMPP for Windows 5.6.28 [compiled Nov 12th 2015]

phpMyAdmin 4.5.1

PHP 5.6.28 (x86)

Apache 2.4.23 (x86)

<https://www.apachefriends.org/es/download.html>

Processing 3.3.3 (x64)

<https://processing.org/download/>

Android Studio 2.2.3 (JRE: 1.8.0\_76-release-b03 amd64)

<https://developer.android.com/studio/index.html>

NetBeans IDE 8.2 (x64)

<http://www.oracle.com/technetwork/java/javase/downloads/jdk-netbeans-jsp-142931.html>

Atom 1.16 (x64)

<https://atom.io/>

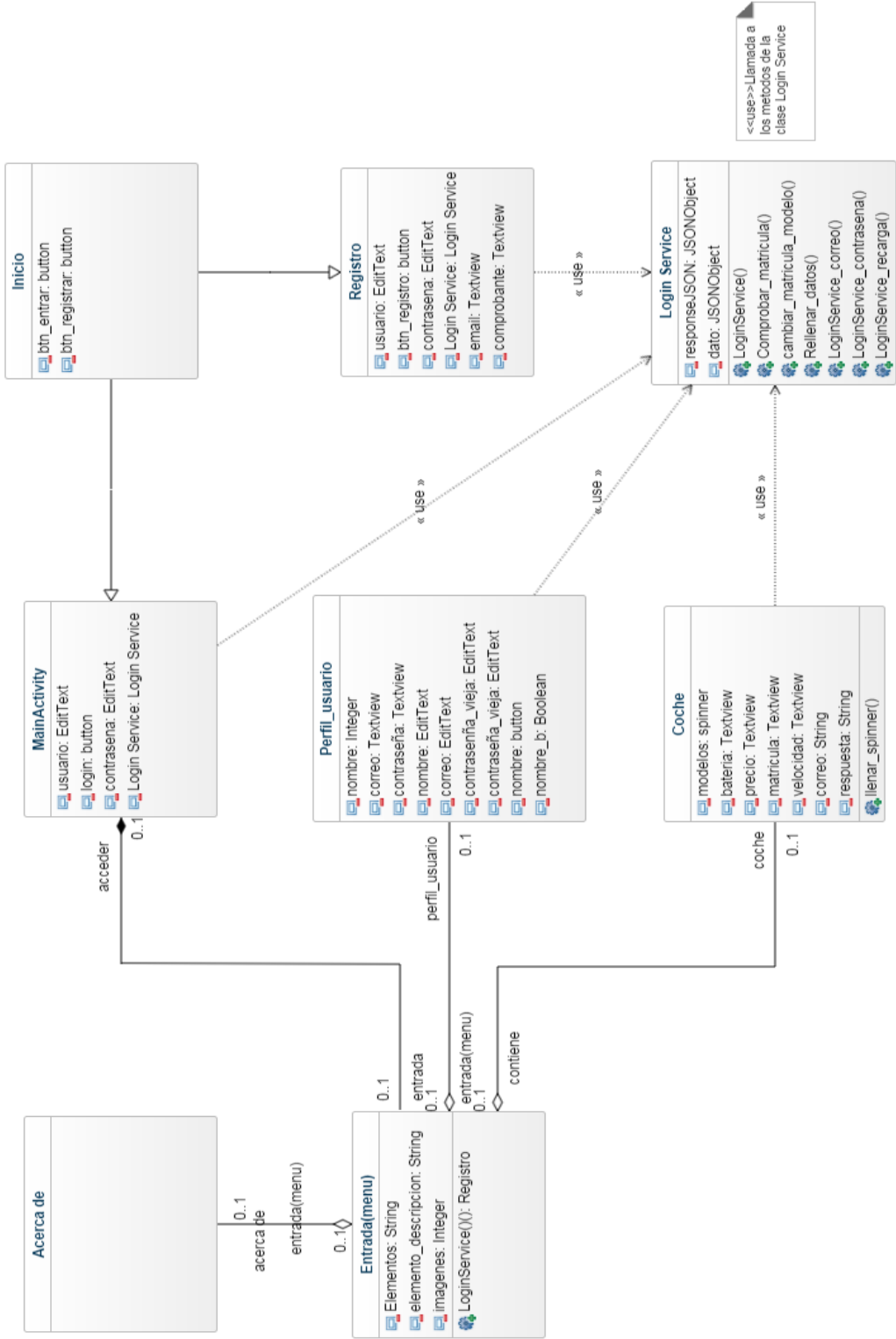
Notepad++ 7.3.3 (x86)

<https://notepad-plus-plus.org/download/v7.3.3.html>

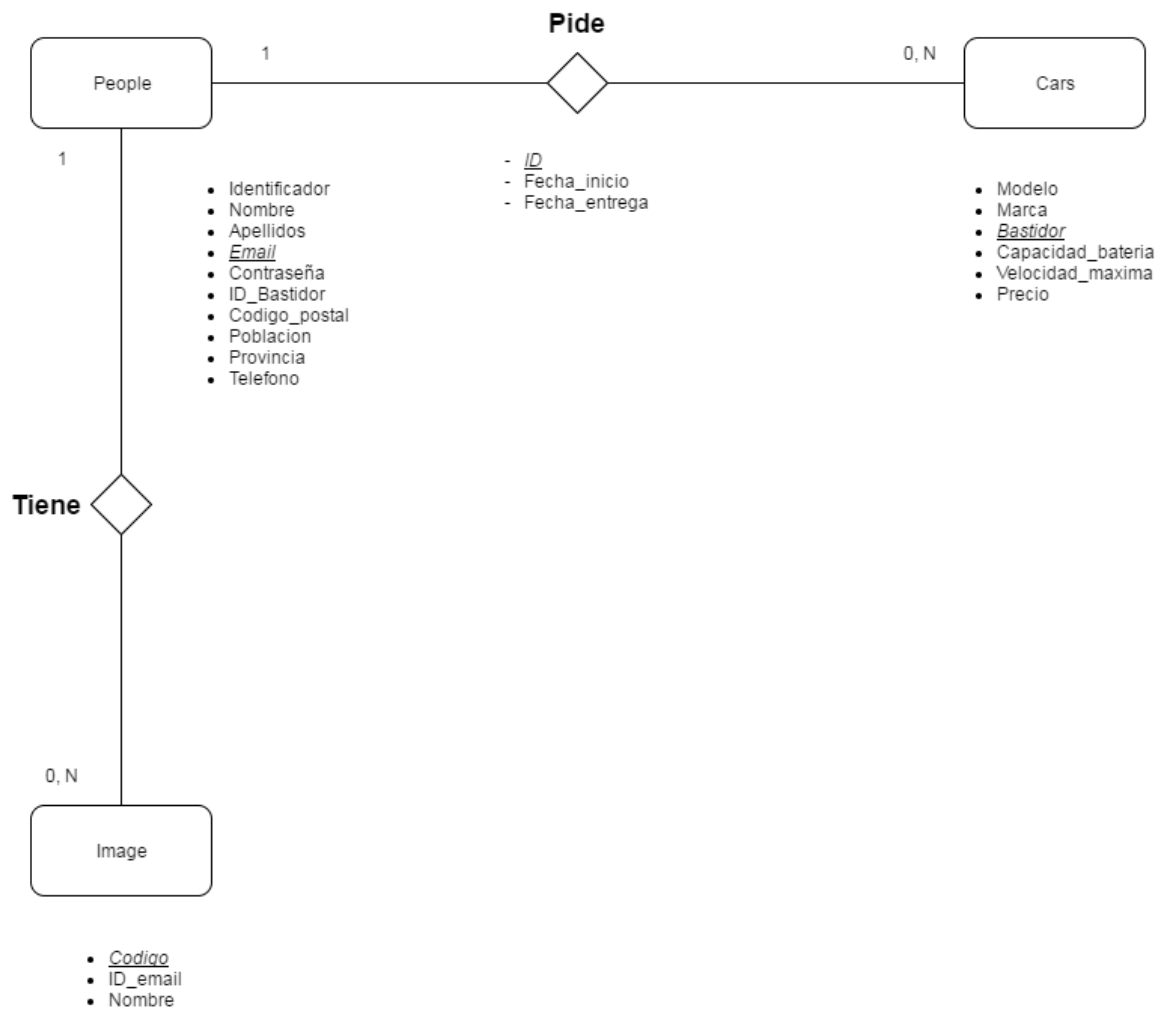
BlueStacks 2.5.90 (x86)

<http://www.bluestacks.com/es/index.html>

DIAGRAMA UML



## DIAGRAMA ENTIDAD-RELACIÓN (CHEN)




---

Usuario (Identificador, Nombre, Apellidos, Email, ID\_Bastidor, Codigo\_Postal, Poblacion, Provincia, Telefono)

Vehiculo (Modelo, Marca, Bastidor, Velocidad\_maxima, Capacidad\_bateria, Precio)

ID\_Bastidor es FK de Bastidor(Vehiculo)

Imagen(ID, ID\_Email, Nombre)

ID\_Email es FK de Usuario(Email)





### Genius Driver

Trata sobre la conducción autónoma de un vehículo seguido por otro, sin que este se choque, prediciendo los movimientos que tiene que hacer para poder llegar a su destino, eligiendo el camino más corto.



- Wilson Philip



- Abdel Ali (WebService) y
- Wilson Philip (Processing)



- Abdel Ali y Javier Montes



- Ivan Abe



- Javier Montes (Memoria)



- Todo el grupo

Portaveu i responsable de comunicar les necessitats de l'equip // Responsable de repositor de codi font // Responsable de pissarra de treball // Responsable de timing i qualitat // Responsable de documentació

Història Vertical 1:	Història Vertical 2:	Història Vertical 3:	Història Vertical 4:	Història Vertical 5:	Lliurament Final:
Tasques HV1: Data inici: 15/09/2016					
Data demo: 12/12/2016  Idea principal sobre el proyecto e investigación sobre el lenguaje de programación Processing.  Investigación sobre el lenguaje web PHP para realizar la WebService.  Cambio de antiguo logo a nuevo logo mejorado.	Tasques HV2: Data inici: 12/01/2017				
	Data demo: 25/04/2017  Investigación e implementación de la arquitectura RESTFUL por metodo POST en la WebService para encriptar los datos enviados.	Tasques HV3: Data inici: 26/04/2017			

	<p>Creada la Base de Datos con tablas esenciales para que guarde información sobre la aplicación android.</p> <p>Interfaz básica en web para conectarse a la WebService (Login y Registro).</p>			
		<p>Data demo: 20/04/2017</p> <p>Diseño básico sobre la interfaz gráfica de la aplicación android que incluye Login, Registro e Inicio de sesión.</p> <p>Conseguido el login, registro e inicio de sesión en web a través de WebService con diseño básico.</p> <p>Creación de las consultas a la Base de Datos Peopel para obtener los datos que necesitamos tanto para web como para la aplicación android.</p>	<p>Tasques HV4: Data inici: 24/04/2017</p>	

		Anotación sobre algunos puntos clave sobre documentación de proyecto (Portada, Indice, Abstract, tarjetas de visita,...).		
			<p>Data demo: 12/05/2017</p> <p>Logro conseguido en diseñar las pestañas “Perfil”, “Coche”, “Mapa” y “Acerca de” con su respectivo contenido funcionando.</p> <p>Diseño de página web terminado con todas sus pestañas operativas.</p> <p>Queda por hacer el apartado del administrador y añadir estadísticas de uso y gráficas dinámicas.</p> <p>Seguimiento de los diversos puntos en documentación</p>	<p>Tasques HV5: Data inici: 15/05/2017</p>

			para ir redactando la memoria del proyecto.		
				Data demo: 29/05/2017	Tasques lliurament: Data inici: ____/____/____
					Data lliurament:  01 /Maig /2017

## REPOSITORIO WEB

En el siguiente enlace se puede consultar el código generado hasta la fecha para hacer funcional el proyecto que tenemos entre manos y que día tras día sigue mejorando.

En él podemos consultar el código que tenemos estructurado para la aplicación android, la base de datos generada y la webservice que conecta estos dos apartados. A parte del faltante código html perteneciente a la página web que se incluirá más adelante.

<https://github.com/Potterniker/GeniusDriver>

## DISEÑO APLICACIÓN MOVIL

Cuando iniciamos la aplicación una pantalla de bienvenida nos saluda anunciando que estamos entrando en la aplicación Genius Driver. Trascurridos unos segundos nos invita a iniciar sesión o registrarnos para poder acceder o crear nuestra cuenta personal.



Ilustración 1 – Pantalla de Bienvenida

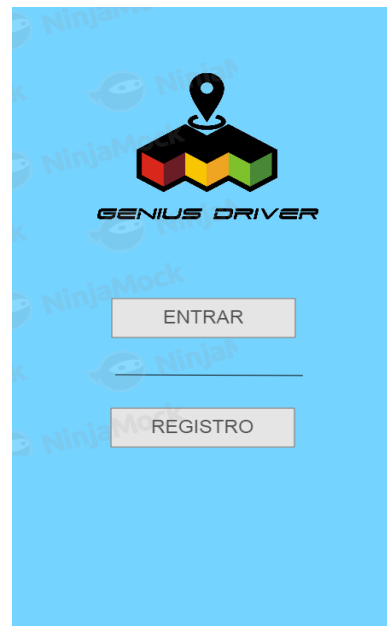


Ilustración 2 – Pantalla Principal

Si escogemos el camino de registrarnos nos tomara poco tiempo, ya que, son pocos los apartados que nos piden, pero los más esenciales. Cuando el usuario tenga tiempo disponible podrá completar los demás campos una vez inicie sesión.

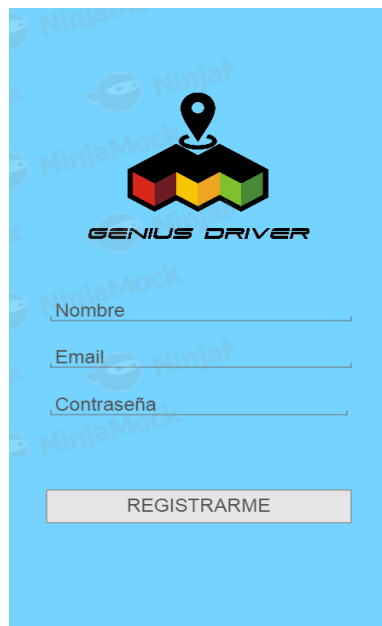
The image shows a registration screen for 'GENIUS DRIVER'. At the top, there is a logo consisting of a black location pin above a stylized map with red, yellow, and green segments. Below the logo, the text 'GENIUS DRIVER' is displayed. The screen features three input fields: 'Nombre', 'Email', and 'Contraseña'. At the bottom, there is a button labeled 'REGISTRARME'. The background is light blue with a faint 'NinjaMock' watermark.

Ilustración 3 – Pantalla de registro

Una vez terminado el registro, o si el usuario ya disponía de una cuenta creada por medio de la web, será redirigido a la pantalla principal donde podrá iniciar sesión con su nueva cuenta.

The image shows a login screen for 'GENIUS DRIVER'. It features the same logo and 'GENIUS DRIVER' text as the registration screen. Below the logo, there are two input fields: 'Nombre' and 'Email'. At the bottom, there is a button labeled 'INICIAR SESIÓN'. The background is light blue with a faint 'NinjaMock' watermark.

Ilustración 4 – Pantalla de inicio de sesión

A continuación de haber accedido a su cuenta personal, se mostrará el **Menú Principal del usuario** donde tendrá disponible una serie de opciones, como:



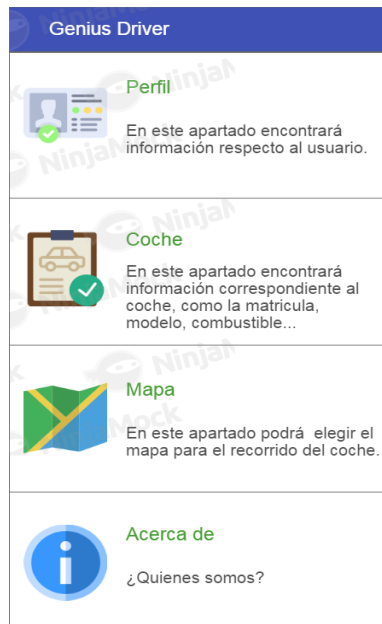


Ilustración 5 – Pantalla Menú Principal

- **Sección Perfil:** en este lugar podrá revisar la información personal que tenemos guardada de él en nuestra base de datos, con la posibilidad de poder cambiarlos si están erróneos o actualizar la información si se necesita.
- **Sección Coche:** en ella podrá seleccionar de un listado el modelo de coche que quiera conducir en su día, asimismo, se informa de las características más relevantes del coche escogido.



Ilustración 6 – Sección de Perfil



Ilustración 7 – Sección de Coche

- **Sección Mapa:** este apartado es uno de los más relevantes que contiene nuestra App debido a que brinda la posibilidad de poder capturar con la cámara de un dispositivo móvil el circuito que queremos recorrer. Una vez realizada la captura se previsualizará el plano del circuito y cuando el usuario este conforme con lo obtenido podrá subir la imagen al servidor donde se guardará en su cuenta personal como copia de seguridad, pero también tendrá disponible la captura en su dispositivo móvil por si desea compartirla. Esta acción la puede repetir varias veces y luego de todas las imágenes guardadas podrá seleccionar que circuito aplicar.
- **Sección Acerca de:** describe la empresa creadora de la aplicación y en que versión se encuentra.

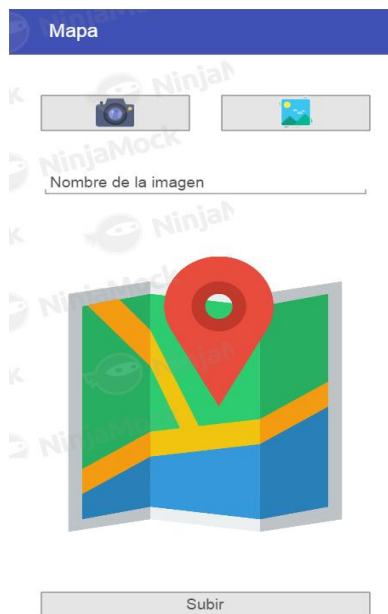


Ilustración 8 – Sección de Mapa



Ilustración 9 – Sección Acerca de

## DISEÑO APLICACIÓN MOVIL

A consecuencia de todos los imprevistos que nos hemos encontrado durante la creación del proyecto adjuntamos un croquis preliminar de cómo será el diseño de la página web. Este croquis engloba todas las partes con las que contará el diseño final de web, el cual se expondrá el día oficial de presentación de proyecto.

Por esta razón, las siguientes capturas describen como se visualizará la página web una vez terminado el diseño, tanto desde la parte visual para el usuario como la parte de administración para el encargado de mantenimiento de la página web.

Explicados los contratiempos y que estos se podrán solucionar a tiempo, mostramos las diferentes partes que tendrá la página web definitiva.

La primera pantalla que se muestra es la página principal que el público en general podrá visualizar al buscarla en un navegador. En ella se publicarán de manera habitual oferta sobre nuestros servicios y productos, además de aquellas empresas que nos patrocinan y hacen que esta idea se realice.



*Ilustración 1 – Pantalla principal donde se muestran novedades y empresas colaboradoras*

Cuando el usuario lo desee puede iniciar sesión para acceder a su cuenta privada donde encontrara información referente a su perfil, la lista de vehículos que tenga comprados y los mapas que haya guardado.

Genius Driver Inc. Inicio Sobre nosotros Contacto Login Registro

Inicio de sesión

Email:

Password:

☒ Recordar sesión

Login

[¿Olvidaste la contraseña?](#)

© GENIUS DRIVER 2017 - Todos los derechos reservados | Contacto | Términos y Condiciones de uso | Política de privacidad de datos y seguridad

*Ilustración 2 – Inicio de sesión tanto para el usuario particular como el administrador*

También podemos encontrar en esta pantalla un método de seguridad para recuperar la contraseña, por si en algún momento el usuario la extravía o la olvida pueda volver a conectarse.



The image shows a web browser window with a 'Recuperación de contraseña' (Password Recovery) form. The form is centered on the page and has a white background with a thin border. At the top of the form, the title 'Recuperación de contraseña' is displayed. Below the title, a message states: 'Se enviara un correo de confirmacion a la direccion introducida para proceder con el cambio de contraseña.' (A confirmation email will be sent to the address entered to proceed with the password change). Below this message, there is a label 'Correo @:' followed by a text input field. At the bottom of the form, there is a button labeled 'Enviar'. The background of the browser window shows the Genius Driver website header with navigation links: 'Inicio', 'Sobre nosotros', 'Contacto', 'Login', and 'Registro'. The footer of the website is visible at the bottom of the browser window, containing copyright information and links to 'Contacto', 'Términos y Condiciones de uso', and 'Política de privacidad de datos y seguridad'.

Ilustración 3 – Envío de correo de confirmación

Este método de seguridad consiste en que, a partir del correo introducido, que pertenecerá al usuario que no puede acceder a la cuenta, se le enviara un formulario que contendrá un link el cual le reenviara a otra página para que pueda realizar el cambio de contraseña correspondiente.

Una vez introducida la nueva contraseña esta se cambiará en nuestro servidor, el cual actualizar el campo contraseña por la nueva.

GENIUS DRIVER

[Inicio](#) [Sobre nosotros](#) [Contacto](#) [Login](#) [Registro](#)

### Cambiar contraseña

A continuación, escriba el nuevo valor de contraseña y repítala para confirma que la ha introducido correctamente.

Nueva contraseña:

Repita la contraseña:

© GENIUS DRIVE 2017 - Todos los derechos reservados | [Contacto](#) | [Términos y Condiciones de uso](#) | [Política de privacidad de datos y seguridad](#)

Ilustración 4 – Cambio de contraseña por perdida o descuido

Por otra parte, si el usuario aún no tiene cuenta con nosotros se podrá registrar sin la necesidad de introducir todos los datos, en primera instancia, con insertar aquellos campos que están marcados con asterisco será suficiente para completar la inscripción.

The image shows a web browser window displaying the 'Registro usuario' (User Registration) page of the Genius Driver website. The page has a light gray header with the Genius Driver logo on the left and navigation links 'Inicio', 'Sobre nosotros', 'Contacto', 'Login', and 'Registro' on the right. The 'Registro' link is highlighted. The main content area is titled 'Registro usuario' and contains a form with the following fields: 'Nombre(\*)', 'Apellidos(\*)', 'Correo @(\*)', 'Confirmar correo @(\*)', 'Contraseña(\*)', 'Confirmar contraseña(\*)', 'Telefono de contacto', 'Codigo postal', 'Población', and 'Provincia'. Each field is represented by a text input box. Below the fields, there is a paragraph of text: 'Al identificarte, aceptas nuestras Condiciones de uso, nuestro Aviso de privacidad y nuestras Condiciones de Cookies y publicidad en Internet.' At the bottom of the form, there are two buttons: 'Registrarse' and 'Cancelar'. A small note at the bottom left of the form states: '(\*): los campos marcados con asterisco son obligatorios.' The footer of the page contains the copyright notice '© GENIUS DRIVE 2017 - Todos los derechos reservados' and links to 'Contacto', 'Términos y Condiciones de uso', and 'Política de privacidad de datos y seguridad'.

Genius Driver

Inicio Sobre nosotros Contacto Login Registro

Registro usuario

Rellene todos los campos descritos a continuación para poder asesorarle con mayor criterio.

Nombre(\*):

Apellidos(\*):

Correo @(\*):

Confirmar correo @(\*):

Contraseña(\*):

Confirmar contraseña(\*):

Telefono de contacto:

Codigo postal:

Población:

Provincia:

Al identificarte, aceptas nuestras Condiciones de uso, nuestro Aviso de privacidad y nuestras Condiciones de Cookies y publicidad en Internet.

(\*) los campos marcados con asterisco son obligatorios.

Registrarse Cancelar

© GENIUS DRIVE 2017 - Todos los derechos reservados | Contacto | Términos y Condiciones de uso | Política de privacidad de datos y seguridad

Ilustración 5 – Apartado de nuevo usuario para que pueda inscribirse a la web

Una vez el usuario este registrado o tenga la sesión iniciada será redirigido a la pantalla principal donde se mostraran los apartados Perfil, Vehículo y Mapa que los podrá utilizar para administrar la información guardada en su cuenta.





Ilustración 6 – Mostrando los botones de administración para el usuario

- Uno de los apartados más llamativos es el **Perfil** donde se recopila toda la información personal que se conoce de él. Del mismo modo, cabe destacar que el usuario podrá consultar el Estado de un pedido, si es que ha realizado alguno, como el Historial de vehículos comprados anteriormente.

Perfil Usuario

Si lo desea puede modificar su información personal para ofrecerle el mejor asesoramiento posible.

### Datos personales

Nombre:

Apellidos:

Correo @:

Contraseña:

Confirmar contraseña:

Telefono de contacto:

Codigo postal:

Población:

Provincia:

### Estado del pedido

1 - Se está confirmando el pedido  
2 - Se está confirmando el pedido  
3 - Se está confirmando el pedido  
4 - Se está confirmando el pedido  
5 - Se está confirmando el pedido

### Historial de pedidos

Pedido N°	Fecha Pedido	Nombre	Apellidos	Marca	Modelo	Precio
#1	14/06/2015	Jacinto	Gonzalez Ortiz	Genius	Alcon	180.000€
#2	21/02/2016	Jose	Garcia Meña	Genius	Barton	90.000€

Guardar Cancelar

© GENIUS DRIVE 2017 - Todos los derechos reservados | Contacto | Términos y Condiciones de uso | Política de privacidad de datos y seguridad

Ilustración 7 – Información recopilada de la cuenta de usuario

- En la sección **Vehículo** encontrara definidas las características del modelo cuando seleccione un vehículo de la lista desplegable. Solamente se mostrarán aquellos modelos que el usuario haya comprado.

Escoge un coche

Eliga el vehículo con el que se desea desplazar hoy.

Modelo:

Marca:

Matricula:

Capacidad bateria (mAh):

Velocidad máxima (km/h):

Precio:

© GENIUS DRIVER 2017 - Todos los derechos reservados | Contacto | Términos y Condiciones de uso | Política de privacidad de datos y seguridad

Ilustración 8 – Características del vehículo seleccionado

- Por último, está el apartado de **Mapa** en el cual podrá seleccionar un archivo de imagen que corresponda a un circuito para previsualizarlo y luego subirlo al servidor si le agrada. De igual manera, puede seleccionar un circuito previamente cargado para previsualizarlo y después aplicarlo al simulador.

Visualización del circuito

Puede elegir un archivo de imagen guardado en su ordenador o seleccionar un circuito que tenga archivado en su cuenta para previsualizar y aplicar al simulador.

Elige una foto para subir:

File upload Choose...

--- O ---

Selecciona un circuito que hayas subido:

Select

Aplicar

© GENIUS DRIVE 2017 - Todos los derechos reservados | Contacto | Términos y Condiciones de uso | Política de privacidad de datos y seguridad

Ilustración 9 – Subir un circuito o seleccionarlo para cargarlo en el servidor

La siguiente sección se puede visualizar tanto si tenemos la sesión iniciada como si no. Esto puede ser un inconveniente si necesitamos consultar el estado de algún pedido o enviar un tiquet de incidencia al departamento técnico, ya que se necesitan los datos de inicio de sesión para poder acceder a estos apartados. De lo contrario podemos visualizar la página si necesitamos llamar a algún número de atención al cliente.

Genius Driver

Inicio Perfil Vehículo Mapa Sobre nosotros Contacto Login Registro

Contacta con Genius Driver

### Compras y consultas

#### Asistencia de ventas

Accede a los datos del vehículo reservado. Para hacer un seguimiento o devolver el vehículo entra en la página Estado del Pedido. Para cualquier otra consulta, ponte en contacto con el Servicio de Atención al Cliente de Genius Driver en el teléfono 900 150 503. Atención telefónica de lunes a viernes, de 09:00 a 21:00h, y de 09:00 a 18:00h los sábados.

#### Compras para empresas

Si tienes una empresa o eres un usuario profesional, llama al 900 812 683 y te asesoraremos gratuitamente para brindarte el mejor vehículo para tu actividad empresarial o si lo deseas visita una de nuestras tiendas. El horario de atención es de lunes a viernes de 09:00 a 21:00h. Ofertamos una gran variedad de servicios de negocio y financiación.

#### Encuentra una tienda

Los centros de ventas autorizados los puedes consultar en el siguiente link para localizar el mas cercano a tu zona. En ellos hallaras asesores que te informaran sobre todo lo que necesitas.

### Soporte Técnico

#### Contacta con el soporte tecnico

Por medio de correo electrónico:

¿Necesitas ayuda? Explicanos tu problema en esta página rellenando un formulario indicando la(s) incidencia(s) que hayas podido observar en alguno de nuestros productos y tan rápido como hayamos obtenido el formulario te llamamos nosotros para darte una solución al incidente.

A través de teléfono:

O nos puedes llamar de forma gratuita al numero: 900 812 703, de lunes a viernes de 09:00 a 17:00h donde hablaras con uno de nuestros expertos mecánicos aconsejándote la solución mas oportuna para solventar el problema.

© GENIUS DRIVE 2017 - Todos los derechos reservados | Contacto | Términos y Condiciones de uso | Política de privacidad de datos y seguridad

Ilustración 10 – Sección de contacto de Genius Driver

## START UP/START DOWN

La idea principal del proyecto surgió a partir de un compañero de clase, el cual quería montar algo referente a un coche por radiocontrol. Principalmente lo que deseaba era montarlo con Arduino y que este pudiera recorrer un pequeño circuito dirigido con un mando por el usuario.

El compañero se decantó por Arduino ya que su hermano tenía mucha información sobre el tema porque se dedicó a ello hacía tiempo y él sabía que podría sacarle partido a la situación. A parte que disponía de muchas fuentes de información con libros, videos,...

Al principio empezamos a investigar sobre la idea y vimos que “nuestra idea” la había desarrollado un japonés. Su proyecto consistía en un coche pequeño, el cual circulaba por una pista hecha de papeles para separar los carriles por donde circula el vehículo. En su maqueta también tenía señales de tráfico como semáforos, señales de tráfico y obstáculos que simulaban otros coches en la carretera.

En verdad es que cuando vimos que nuestra idea ya estaba conseguida, por decirlo de algún modo, nos motivó aún más y queríamos hacer algo que fuese diferente, más innovador, más llamativo.

Fue pasando el tiempo hasta que llego el día de la **lluvia de ideas** y estábamos dispuestos a presentar nuestra nueva idea en grupo, la cual consistía en hacer más o menos lo mismo pero esta vez apuntamos más alto y nos dijimos, ¿Por qué no hacemos que el coche vaya por una maqueta donde haya calles con cruces y señales de tráfico de control de velocidad, en vez de simplemente que el coche vaya en línea recta como el proyecto japonés?

Llego el día y presentamos la idea y nadie aparte de nuestro grupo formado anteriormente se interesó en unirse.

Al principio nos pareció un poco extraño que nadie se ofreciera a formar parte de nuestro equipo, y fue a causa de que en ese momento no tuvimos la sensatez de darnos cuenta que apuntábamos muy alto y a lo mejor los demás compañeros no querían enfrentarse a tal reto.

Después de la presentación de ideas llego el momento de formar los grupos, pero paso algo que no esperábamos. El líder del grupo, el creador de la idea ya no quería formar parte del proyecto porque se decantó por otro que le gusto más. En ese momento el equipo se desmonto por completo y los miembros pensamos en unirnos a otros proyectos donde faltara algún miembro.

Antes de tomar tal decisión nos reunimos y nos pusimos de acuerdo en tirar hacia adelante la idea propuesta por el anterior líder. Es cierto que al principio tuvimos un poco de miedo al no tener el material necesario ni recursos ya que habíamos perdido nuestra principal fuente de información, pero a parte de ese inconveniente, nos pareció una brillante idea y quisimos aceptar el desafío.

**Nombre del proyecto:** Genius Driver

**Idea ofrecida por:** Xavier Crespo

**Descripción de la idea:** montar un coche con Arduino y que este pueda recorrer una maqueta esquivando obstáculos y respetando las señales de tráfico.

## EXPLICACION DE LOS ELEMENTOS INVESTIGADOS

En el proyecto hemos utilizado diversidad de lenguajes de programación que no se han estudiado en clase, por tanto hemos tenido que buscar mucha información al respecto.

### **Lenguaje PHP**

Uno de los principales lenguajes que hemos tenido que investigar a fondo ha sido el conocido PHP, ya que ha sido y es el pilar fundamental en el proyecto.

Para empezar, vamos a adentrarnos explicando un poco este lenguaje:

PHP (Hypertext Preprocessor) es un lenguaje de código abierto muy popular especialmente adecuado para el desarrollo web y que puede ser incrustado en HTML. Es popular porque un gran número de páginas y portales están creadas con PHP. Código abierto significa que es de uso libre y gratuito para todos los programadores que quieran usarlo. Incrustado en HTML representa que en un mismo archivo vamos a poder combinar código PHP con código HTML, siguiendo unas reglas.

PHP se utiliza para generar páginas web dinámicas, por ejemplo, los contenidos pueden cambiar a raíz de los valores que haya en una Base de Datos.

Ventajas:

- Lenguaje multiplataforma.
- Fácil de aprender.
- Orientado para desarrollar aplicaciones donde la información este en una base de datos.
- Lenguaje modular.
- Programación orientada a objetos.
- Lenguaje de código libre y gratuito.



**Desventajas:**

- Se necesita instalar en un servidor web.
- Al realizarse todo el trabajo en la parte del servidor, el rendimiento de nuestra aplicación podría verse afectado.
- Difícil de mantener.
- Seguridad. Al ser un lenguaje de código abierto, todas las personas pueden ver el código fuente, y si hay errores, la gente puede aprovecharse.

**Ejemplo:**

```
<!DOCTYPE html>
<html lang="es">
  <head>
    <meta charset="utf-8">
    <title>Ejemplo Página PHP</title>
  </head>
  <body>
    <?php echo "<p>Hello World</p>"; ?>
  </body>
</html>
```

El principal uso que hemos dado a PHP ha sido para poder crear el pilar más importante del proyecto, la **Web Service**.

*¿Qué es la “Web Service”?*

- Es una tecnología que utiliza un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones principalmente en XML.

*¿Qué protocolo hemos utilizado? ¿Qué es RESTful, y cual son sus características principales ?*

- RESTful es el acrónimo de Representational State Transfer, y resumiendo, representa un tipo de arquitectura de interfaces de comunicación basado en un protocolo de cliente/servidor sin estado (protocolo HTTP), cacheable, con unas operaciones bien definidas en el que los recursos están identificados de forma única por URLs. Es importante indicar que la arquitectura REST no es un estándar ni un protocolo, sino una serie de principios de arquitectura.

#### Ventajas:

- Aportan interoperabilidad entre aplicaciones de software independientemente de sus propiedades o de las plataformas sobre las que se instalen.
- Los servicios Web fomentan los estándares y protocolos basados en texto, que hacen más fácil acceder a su contenido y entender su funcionamiento.
- Permiten que servicios y software de diferentes compañías ubicadas en diferentes lugares geográficos puedan ser combinados fácilmente para proveer servicios integrados.

#### Desventajas:

- Para realizar transacciones no pueden compararse en su grado de desarrollo con los estándares abiertos de computación distribuida como CORBA (Common Object Request Broker Architecture).
- Su rendimiento es bajo si se compara con otros modelos de computación distribuida, tales como Java Remote Method Invocation (RMI), CORBA o Distributed Component Object Model (DCOM). Es uno de los inconvenientes derivados de adoptar un formato basado en texto. Y es que entre los objetivos de XML no se encuentra la concisión ni la eficacia de procesamiento.
- Al apoyarse en HTTP, pueden esquivar medidas de seguridad basadas en firewall cuyas reglas tratan de bloquear o auditar la comunicación entre programas a ambos lados de la barrera.

Hay que tener en cuenta los requisitos previos para poder crear un Web Service, que son los siguientes:

- Servidor Web (nosotros hemos utilizado Apache).
- Una base de datos MySQL.
- PHP.

*¿Cómo hemos instalado los requisitos?*

- Hemos utilizado el XAMPP el cual incluye los 3 requisitos para poder crear un Web Service.

Una vez creado el servidor web, vamos a ver que está formado por 2 ficheros que explicaremos a continuación:

- El primero fichero que veremos es el “.htaccess” es un archivo de configuración muy popular en servidores web basados en Apache que permite a los administradores **aplicar distintas políticas de acceso a directorios o archivos** con la idea de mejorar la seguridad de su página web y, por tanto, evitar acceso a terceros.

```
1 RewriteEngine On
2 RewriteRule ^([a-zA-Z_-]*)$ index.php?action=$1
3 RewriteRule ^([a-zA-Z_-]*)/([0-9]+) index.php?action=$1&id=$2 [L,QSA]
4
5
```

- El segundo fichero es el encargado de hacer la conexión con la base de datos, así que lo que hemos hecho es hacer un constructor vacío, el cual contendrá los datos de la conexión con la IP, usuario, contraseña y la base datos a la que se tendrá que conectar.

```
<?php
2 /**
3  * @web http://www.jc-mouse.net/
4  * @author jc mouse
5  */
6
7 class PeopleDB {
8
9     protected $mysql;
10     const LOCALHOST = '127.0.0.1';
11     const USER = 'root';
12     const PASSWORD = '';
13     const DATABASE = 'dbtest';
14
15
16     /**
17      * Constructor de clase
18      */
19     public function __construct() {
20         try{
21             //conexión a base de datos
22             $this->mysql = new mysqli(self::LOCALHOST, self::USER, self::PASSWORD, self::DATABASE);
23         }
24     }
25 }
```

- Una vez creada la conexión para poder extraer los datos y poder utilizarlos hemos creado métodos con los comandos MySQL.
- Este comando por ejemplo es para extraer todos los usuarios que hay en la base de datos correspondiente a la tabla “people” que explicare más adelante.

```
public function recuperar_usuarios() {  
    $stmt = $this->mysqli->prepare("SELECT * FROM people");  
  
    $stmt->execute();  
    $result = $stmt->get_result();  
    $peoples = $result->fetch_all(MYSQLI_ASSOC);  
    $stmt->close();  
    return $peoples;  
}
```

- Si nos fijamos devuelve los datos dentro de un array que luego vamos a utilizarlo en cada fichero oportuno.
- Para cada determinada petición o comprobación que hacemos ya sea desde web o Android creamos un fichero PHP y dentro llamamos al método que hayamos creado en “PeopleDB” y este lo devolverá a Android o a la Web codificado en JSON.

*Ejemplo en el cual extraeremos todos los coches de la tabla “cars” y los enviamos por PHP a través del método POST y con la función “response”:*

```
public function get_modelos() {  
    $stmt = $this->mysqli->prepare("SELECT Modelo FROM cars");  
    $stmt->execute();  
    $result = $stmt->get_result();  
    $peoples = $result->fetch_all(MYSQLI_ASSOC);  
    $stmt->close();  
    return $peoples;  
}
```

```

<?php
function response($code = 200, $error = "", $Exito = false, $message = "", $dato) {
    http_response_code($code);
    if (!empty($Exito) && !empty($message)) {
        $response["Exito"] = $Exito; $response["message"] = $message; $response["data"] => $dato;
        echo json_encode($response, JSON_PRETTY_PRINT);
    }
}

```

```

<?php
include_once 'PeopleDB.php';
include_once 'Respuesta.php';

$json = file_get_contents('php://input');

if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    $persona = new PeopleDB();
    $ret = $persona->get_modelos();
    $dato = $ret;
    response($code = 200, $error = "", $Exito = true, $message = "", $dato);
}

```

El método `response` lo que hace es enviar todo codificado en JSON y lo que tenemos nosotros en el método es un array dentro de otro array. En el segundo array es donde ira el listado de modelos.

Más adelante explicaremos como recibimos los datos en Android.

Asimismo, si repasamos todo el uso de PHP en el proyecto también podemos ver que lo hemos utilizado en la Web para poder enviar y recibir los datos del servidor tal y como lo hacemos en la parte de Android.

## Android Studio

Asimismo, también hemos tenido que adentrarnos más en el mundo del Android, pero no ha sido tan complicado ya que teníamos una base con lo aprendido en clase.

Por una parte, hemos tenido bastante suerte porque el lenguaje de programación de Android es Java y prácticamente es el lenguaje que más hemos tocado en este curso.

No vamos adentrarnos mucho en la explicación sobre este lenguaje, pero explicaremos lo básico:

El lenguaje de programación de Android se hace habitualmente con lenguaje de programación similar a Java y el conjunto de herramientas de desarrollo SDK, pero hay otras opciones disponibles.

Ejemplo:

```
package net.sgoliver.android.holausuario;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

En la parte realizada con Android hacemos un inciso para explicar cómo enviamos y recibimos los datos desde el servidor web.

En Android es muy frecuente lanzar nuevos hilos. Tendremos que hacerlo siempre que exista la posibilidad de que una tarea pueda bloquear el hilo de la interfaz de usuario. Esto suele ocurrir en cálculos complejos o en accesos a la red.

Tras ver el uso de las herramientas estándares en Java para crear hilos; en este apartado veremos una clase creada en Android que nos ayudará a resolver este tipo de problemas de forma más sencilla, la clase AsyncTask.

Una tarea asincrónica (Async-Task) se define por un cálculo que se ejecuta en un hilo secundario y cuyo resultado queremos que se publique en el hilo del interfaz de usuario.

Una tarea asíncrona esta de la siguiente forma:

```
class MiTarea extends AsyncTask {

    @Override protected void onPreExecute() {

        ...

    }

    @Override protected Resultado doInBackground(Parametros... par) {

        ...

    }

    @Override protected void onProgressUpdate(Progreso... prog) {

        ...

    }

    @Override protected void onPostExecute(Resultado result) {

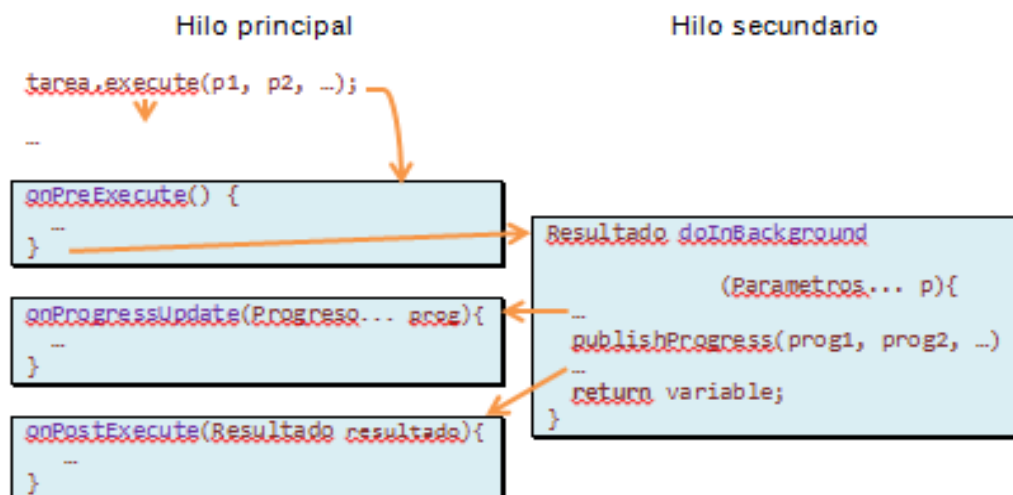
        ...

    }

}
```

- `onPreExecute()`: en este método tenemos que realizar los trabajos previos a la ejecución de la tarea. Se utiliza normalmente para configurar la tarea y para mostrar en el la interfaz de usuario que empieza la labor.

- `doInBackground(Parametros...)`: es llamado cuando termina `onPreExecute()`. Es la parte más importante donde tenemos que realizar la tarea propiamente dicha. Es el único método de los cuatro que no se ejecuta en el hilo del interfaz de usuario. Lo va a hacer en un hilo nuevo creado para este propósito. Como hemos visto en la clase `AsyncTask` ha de ser parametrizada con tres tipos de datos, es decir, cuando creas un `AsyncTask` la clase `Parámetros` ha de ser reemplazada por una clase concreta que será utilizada para indicar la información de entrada a la tarea.
- `onProgressUpdate(Progress...)`: este método se utiliza para mostrar el progreso de la tarea al usuario. Se ejecuta en el hilo interfaz de usuario, por lo que podremos interactuar con las vistas. El progreso de una determinada tarea ha de ser controlado por el programador llamando al método `publishProgress(Progress...)` desde `doInBackground()`. La clase `Progress` es utilizada para pasar la información de progreso. Un uso frecuente es reemplazarla por `Integer` y representar el porcentaje de progreso en un valor entre el 0 y el 100.
- `onPostExecute(Result)`: este método se usa para mostrar en el interfaz de usuario el resultado de la tarea. El parámetro de entrada (de la clase `Result`) corresponde con el objeto devuelto por el método `doInBackground()`.



Una vez explicado todo el funcionamiento de la tarea asincrónica, pasaremos a mostrar como lo hemos aplicado nosotros a nuestro proyecto.



A continuación, pondremos una captura de pantalla de la parte de Registro para que se vea claramente como hemos utilizado el “AsyncTask”:

```
public void onClick(View v) {  
    switch (v.getId()) {  
        case R.id.btnRegistro:  
            String txtUsr = usr.getText().toString();  
            String txtemail = email.getText().toString();  
            String txtPwd = pwd.getText().toString();  
  
            if (txtUsr.equals("") || txtPwd.equals("") || txtemail.equals("")) {  
                Toast.makeText(this, "Por favor, introduce todos los campos!",  
                    Toast.LENGTH_LONG).show();  
            }  
            else {  
                loginService = new LoginService();  
                loginService.execute(txtUsr,txtPwd,txtemail);  
            }  
            break;  
  
        default:  
            break;  
    }  
}
```

```
protected void onPreExecute() {  
    super.onPreExecute();  
    pDialog = new ProgressDialog(Registro.this);  
    pDialog.setMessage("Actualizando Servidor, espere...");  
    pDialog.setProgressStyle(ProgressDialog.STYLE_SPINNER);  
    pDialog.show();  
}
```

Como hemos explicado, el “onPreExecute” se ejecutara antes del “doInBackground” y lo que hemos hecho ha sido lanzar una ventana de carga que luego quitamos cuando se acaba de hacer el “doInBackground”, es decir, cuando se lanza el OnPostExecute.

Como podemos ver comprobamos que todos los campos estén llenos. Una vez lo están, iniciamos la tarea asincrónica que la hemos llamado “LoginService”, a la cual le hemos pasado por parámetro:

- Usuario.
- Contraseña

- Email

```
// @SuppressWarnings("NewApi")
private class LoginService extends AsyncTask<String, String, Boolean> {
    ProgressDialog pDialog;
    JSONObject responseJSON = new JSONObject();
    @Override
    protected Boolean doInBackground(String... params) {
        boolean result = false;
        HttpClient httpClient = new DefaultHttpClient();
        HttpPost request = new HttpPost("http://" + IP + "/SampleWS/Registro_android.php");
        request.setHeader("content-type", "application/json");
        JSONObject dato = new JSONObject();
        try {
            dato.put("email", params[2]);
            dato.put("nombre", params[0]);
            dato.put("contrasena", params[1]);

            StringEntity entity = new StringEntity(dato.toString());
            entity.setContentType(new BasicHeader(HTTP.CONTENT_TYPE, "application/json"));

            request.setEntity(entity);

            HttpResponse response = httpClient.execute(request);
            //Realizamos la petición y capturamos la respuesta en el objeto response
            String respStr = EntityUtils.toString(response.getEntity());
            //lo volvemos a parsear a JSON porque lo teniamos en string igual que antes
            JSONObject respJSON = new JSONObject(respStr);
            if (respJSON.getBoolean("Exito") == false) {
                result = false;
            } else {
                result = true;
                responseJSON = respJSON;
            }
        }
    }
}
```

Para poder realizar la conexión al servidor web hemos creado un objeto de la clase “HttpClient” para poder abrir una conexión del cliente al servidor web. Seguidamente hemos creado un objeto de la clase “HttpPost” por tal de poder enviar los datos al PHP por el método POST.

Una vez enviados los parámetros en formato JSON, es hora de capturarlos por el PHP. que lo captura de la siguiente manera:

```
<?php
2
3 include_once 'PeopleDB.php';
4 include_once 'Respuesta.php';
5
6 $json = file_get_contents('php://input');
7
8 $obj = json_decode($json);
9 $nombre = $obj->{'nombre'};
10 $email = $obj->{'email'};
11 $contrasena = $obj->{'contrasena'};
12
13
14
15
16 if ($_SERVER['REQUEST_METHOD'] == 'POST') {
17     $dato = array('nombre' => $nombre, 'email' => $email, 'contrasena' => md5($contrasena));
18
19     $persona = new PeopleDB();
20
21     $ret = $persona->insert($nombre, $email, md5($contrasena));
22
23     if (!$ret == '1') {
24         response($code = 200, $error = "", $exit = false, $message = "ERROR,CORREO ELECTRONICO EN USO!", $dato);
25     } else {
26         response($code = 200, $error = "", $exit = true, $message = "Registrado correctamente", $dato);
27     }
28 } else {
29     response($code = 200, $error = "", $exit = false, $message = "ERROR,CORREO ELECTRONICO EN USO!", $dato);
30 }
```

En el PHP una vez capturados los datos, los introducimos dentro de variables para poder utilizarlas y pasarlas por parámetro en los métodos creados para el registro.

En ese ejemplo, os mostramos el método que hemos creado para hacer el registro de un usuario en la base de datos “dbtest” en la tabla “people”:

```
public function insert($name='', $email='', $contrasena='') {

    $stmt = $this->mysqli->prepare("INSERT INTO people(name,email,contrasena) VALUES (?, ?, ?); ");
    $stmt->bind_param('sss', $name, $email, $contrasena);
    $r = $stmt->execute();
    $stmt->close();
    return $r;
}
```

Como podemos ver este método nos devuelve una variable. Lo que contiene esta variable es un “0” o un “1”. ¿Qué significa cada número?

Si vemos que el método nos ha devuelto el número 1, significa que ha ido todo bien y se han insertado los datos correctamente. En cambio, si el método devuelve un 0 significa que no ha salido bien. ¿Qué puede haber ido mal?

Principalmente es porque el método está mal escrito o lo más frecuente, la repetición de una clave primaria al hacer el “INSERT”.

Y como hemos explicado anteriormente llamamos a este método que hemos creado en “PeopleDB” y le pasamos las variables que hemos capturado, las que han llegado desde

Android. Y dependiendo de lo que nos devuelva el método tendremos que enviar al Android una respuesta u otra.

```

<?php
2
3 include_once 'PeopleDB.php';
4 include_once 'Respuesta.php';
5
6 $json = file_get_contents('php://input');
7
8 $obj = json_decode($json);
9 $nombre = $obj->{'nombre'};
10 $email = $obj->{'email'};
11 $contrasena = $obj->{'contrasena'};
12
13
14
15
16 if ($_SERVER['REQUEST_METHOD'] == 'POST') {
17     $dato = array('nombre' => $nombre, 'email' => $email, 'contrasena' => md5($contrasena));
18
19     $persona = new PeopleDB();
20
21     $ret = $persona->insert($nombre, $email, md5($contrasena));
22
23     if (!$ret == '1') {
24         response($code = 200, $error = "", $exit = false, $message = "ERROR,CORREO ELECTRONICO EN USO!", $dato);
25     } else {
26         response($code = 200, $error = "", $exit = true, $message = "Registrado correctamente", $dato);
27     }
28 }

```

Como hemos explicado antes, para poder enviar la respuesta al Android tendremos que hacer servir el método “response” que tenemos dentro en el fichero “Respuesta.php”. Una vez enviada la respuesta en formato JSON, toca capturarla desde Android. Así que vamos a volver al método asincrónico de antes.

```

HttpResponse response = httpClient.execute(request);
//Realizamos la petición y capturamos la respuesta en el objeto response
String respStr = EntityUtils.toString(response.getEntity());
//lo volvemos a parsear a JSON porque lo teniamos en string igual que antes
JSONObject respJSON = new JSONObject(respStr);
if (respJSON.getBoolean("Exito") == false) {
    result = false;
} else {
    result=true;
    responseJSON = respJSON;
}
} catch (JSONException e) {
    e.printStackTrace();
} catch (UnsupportedEncodingException e) {
    e.printStackTrace();
} catch (ClientProtocolException e) {
    e.printStackTrace();
} catch (IOException e) {
    e.printStackTrace();
}
return result;

```

Como podéis ver hemos creado un objeto de la clase “HttpResponse” que es el que captura el mensaje en JSON enviados desde el PHP. Una vez recibido comprobamos que si el “boolean” que ha llegado está en “TRUE”. Si ha llegado en “TRUE”, en el método “OnPostExecute” informamos al usuario de que se ha registrado correctamente y lo redirigimos a la pantalla de **Inicio de sesión**. Si no, le informamos del error.



```
protected void onPostExecute(Boolean result) {  
    pDialog.dismiss();  
    if(result.equals(true)){  
        try {  
            JSONObject datos=responseJSON.getJSONObject("dato");  
            comprobante.setText("Registrado Correctamente!");  
  
            Toast.makeText(Registro.this, "Introduce tus datos registro!",  
                Toast.LENGTH_LONG).show();  
            Intent intent =  
                new Intent(Registro.this, MainActivity.class);  
            startActivity(intent);  
            finish();  
        } catch (JSONException e) {  
            e.printStackTrace();  
        }  
    }  
    else  
    {  
        Toast.makeText(Registro.this, "Correo en uso!",  
            Toast.LENGTH_LONG).show();  
        usr.setText("");  
        pwd.setText("");  
        email.setText("");  
        usr.requestFocus();  
    }  
}
```

## Entorno de desarrollo Processing

Por último, el lenguaje utilizado para la simulación del coche en pista es Processing. Nos hemos decantado por este lenguaje porque no es difícil utilizarlo y el código como en Android está basado en Java.

A continuación, haremos una breve introducción a este “lenguaje”:

Processing es un lenguaje de programación y entorno de desarrollo integrado de código abierto basado en Java, de fácil uso, y que sirve como medio para la enseñanza y producción de proyectos multimedia e interactivos de diseño digital.

Ejemplo:

```
/*  
  
  Programa de ejemplo introductorio a Processing  
  Pinta un texto de saludo y una línea acorde a la posición del mouse  
  
*/  
  
void setup(){  
  //se ajusta el tamaño de la ventana  
  size(500,500);  
}  
  
void draw (){  
  //pon el fondo de color gris  
  background (128);  
  //ajusta el tamaño del texto  
  textSize(25);  
  //define el texto y sus coordenadas  
  text("Hola al Mundo de processing",50,200);  
  //dibuja una línea entre el origen de coordenadas y la posición del mouse  
  line(0,0,mouseX,mouseY);  
}
```

A través de processing hemos podido realizar una simulación de cómo debería actuar un coche autónomo. Para ello hemos creado un proyecto que contiene las siguientes clases: Genius Driver, Coche, Giro, Mapa, Radar. A continuación explicamos cada una de ellas.

## 1. GENIUS DRIVER

La clase Genius Driver representa el **Main** de nuestro proyecto. En el indicarnos tanto la información como la disposición y tamaño del proyecto, backgrounds, mapas/circuitos y también creamos los objetos y métodos necesarios para iniciar la simulación del coche.

```
Mapa mp, mp2;
Coche c1, c2;

void setup() {
  size(1200, 700);
  smooth();

  // Creamos los Objetos
  c1= new Coche("Coche1", 100, 250, 60, 40, 3); //Parámetros: posX,posY, tamX,tamY,speed
  mp= new Mapa();
  mp2= new Mapa();
}
```

```
void draw() {
  background(255);
  iniciarGeniousDriver();
}

void iniciarGeniousDriver()
{
  // Mostramos mapa
  mp.mostrarMapaImagen("mapaCurva.png"); //Iniciar en pos y = 250 para mejor resultado

  //Recuadro blanco para los datos
  noStroke();
  fill(-1);
  rect(width-120,100,200,300);

  // Iniciamos el coche 1
  c1.iniciarCoche();
}
```

## 2. CLASE COCHE

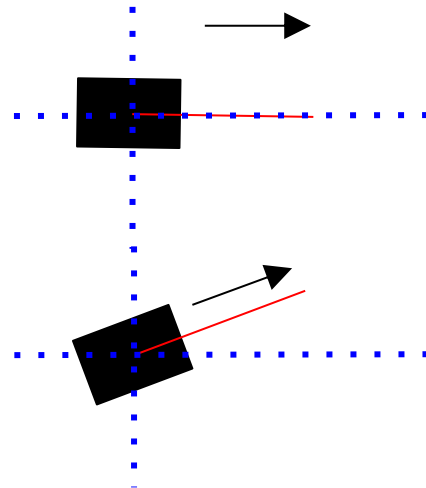
Esta clase nos permite dibujar el coche y crear los métodos o funciones básicas de las cuales dispone el coche.

### Movimiento del Coche

Uno de los primeros problemas que nos hemos encontrado ha sido el mover el coche.

Si se tiene conocimiento de Processing a simple vista es fácil dibujar un rectángulo y hacer que este avance, puesto que simplemente hay que incrementar el valor de X para avanzar horizontalmente y controlar el valor de Y para subir o bajar el objeto.

Sin embargo si hacemos esto el resultado no sería el esperado puesto que si modificamos los valores de X e Y solo conseguimos cambiar los puntos en el cual se encuentra el objeto. Lo que realmente queremos es que el Coche avance siguiendo una dirección y que este pueda rotar. La solución como se explica más adelante es la trigonometría.



### Métodos y Atributos de la Clase Coche

Aquí vemos las variables que nos permitirán dibujar el coche, situarlo en processing y hacer que avance y rotate.

### Constructor Coche

```
//Atributos
float x;
float y;
float midaX;
float midaY;
float speed, speedAux;
String idCoche;
float rotacion=0, rot=0;
float posInix=0, posIniY=0;
boolean stopRadar=false;
```

Con este constructor recibimos los datos para iniciar un coche desde la Clase Genius Driver.



```
//Constructor
public Coche(String id, int x, int y, int midaX, int midaY, float speed)
{
    this.posIniX=x;
    this.posIniY=y;
    this.x=x;
    this.y=y;
    this.midaX=midaX;
    this.midaY=midaY;
    this.idCoche=id;
    this.speed=speed;
    this.speedAux=speed;
}
```

### Método iniciarCoche

Aquí simplemente llamaremos a los métodos que harán posible el desarrollo del coche. Con el **display Car** dibujamos el coche, seguidamente el programa **centrará el coche** en la carretera de tal manera que si el vehículo está muy a la derecha tenderá a girar a la izquierda y viceversa, hasta lograr que este circule lo más al centro posible. **Radar frontal** controlara si hay algún objeto delante del coche, de ser así dejará de avanzar. Si el obstáculo desaparece el coche reanudará la marcha. **Control de Giro** nos permite rotar el vehículo con los teclados, UP para rotar a la izquierda DOWN para rotar derecha. Esto nos es útil a la hora de hacer pruebas. Finalmente **Datos**, este recogerá los datos más importantes y los mostrará por pantalla para hacer un seguimiento de lo que está ocurriendo en processing.

A continuación se detallarán más los métodos utilizados.

```
public void iniciarCoche()
{
    //Creamos Objetos
    rFrontal= new Radar2();
    rIzq= new Radar2();
    rDer= new Radar2();
    giro=new Giro();

    //Mostramos el coche con o sin rotación
    displayCar(x, y, rotacion);

    //Centrar coche
    centrarCoche(x, y, rotacion);

    //Radar Frontal - Si detecta un Obstáculo se para si no, sigue
    radarFrontal(x, y, rotacion);

    //Control de giro con Teclado
    float rot = giro.giroTeclado();
    rotacion=rot+rotacion; //rotacion=0;// Con esta línea hacemos que no gire

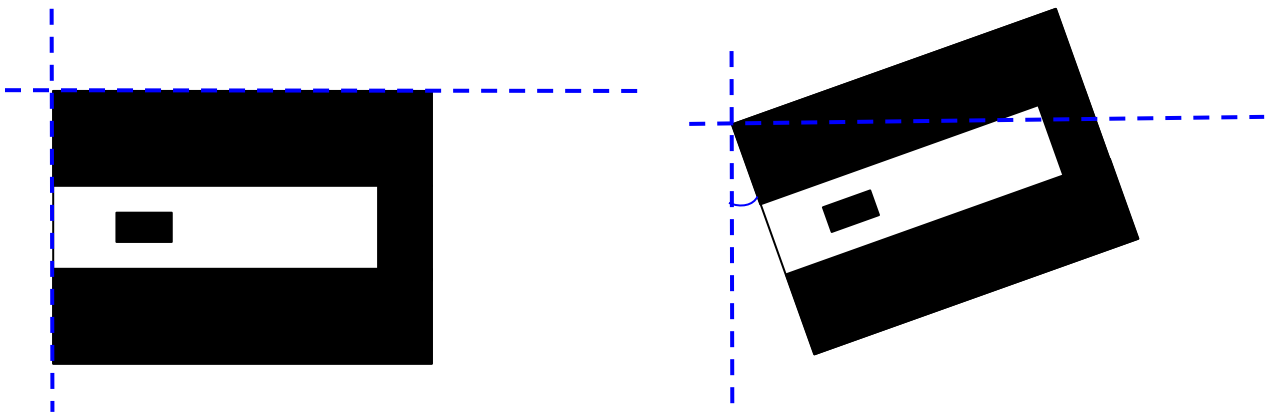
    // Llamamos al método datos
    datos();
}
```

### Método displayCar

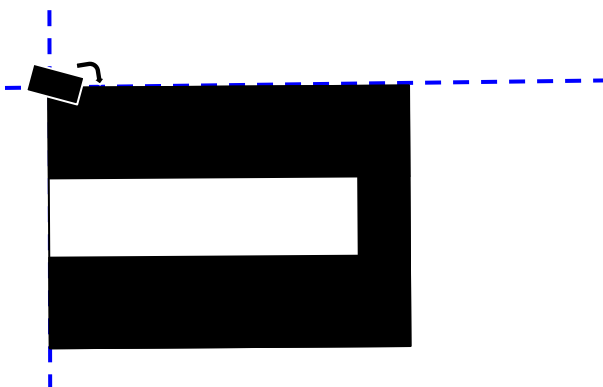
Display car sin duda ha sido uno de los problemas que nos hemos encontrado a la hora de hacer el trabajo.

Como podemos ver, disponemos de un `pushMatrix` un `Translate` y un `popMatrix`. Con esto lo que hacemos es hacer un `Translate` a todo lo que se encuentre dentro de del `push/popMatrix`. Pero ¿Qué es lo que hace el `translate`? Pues bien, primero es necesario explicar qué `processing` a la hora de dibujar lo hace por capas, es decir, el `background` es una capa, el coche es otra capa, los obstáculos otras capas, etc

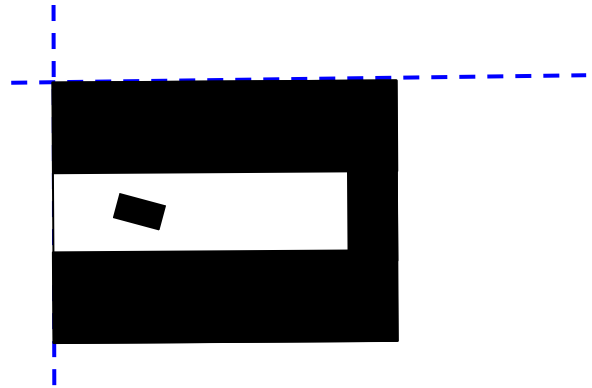
Lo que hace el `translate` es coger dichas capas y moverlas a las coordenadas que se le indica. El `translate` en este caso tiene un la función de acompañar al `rotate`. Como se ilustran en las imágenes si simplemente realizaremos un `rotation`, sin ningún tipo de filtro como es el `push/popMatrix` lo que haría `processing` es rotar todas las capas juntas, como se ve en la segunda imagen.



Esto no es lo que queremos, ya que solo queremos rotar el coche y no el `background`. Pues bien como podemos ver el centro de rotación son las coordenadas 0,0 y para poder rotar el coche hemos de encontrar su centro y hacer que rote sobre si mismo. Así que lo que hacemos es indicar con el `rectMode(Center)` que los puntos se encuentra en el centro del coche, seguidamente con el `translate` situamos el coche en las coordenadas 0,0, procedemos a rotar si hay que rotar y devolvemos todo a la normalidad



Básicamente aquí el problema es que desde processing no se puede visualizar de igual manera los pasos que se realizan para hacer el translate y el rotate ya que lo hace automático, así que era confuso saber que se estaba rotando que no y sobre que eje.



```
void displayCar(float x, float y, float rotate)
{
  pushMatrix();
  rectMode(CENTER);
  translate(x, y);
  rotate(radians (rotate));
  dibujarCar();
  popMatrix();
  rotate=0;
  //Informacion para mostrar
  fill(0);
  textSize(10);
  text("Coordenadas ["+x+" , "+y+"]", width-210, 10);
}
```

### Método dibujarCar

Para aquellos que sepan processing, al dibujar el coche verán que estamos situando las coordenadas del coche en (0,0) y puede parecer que no tiene sentido ya que significa que el coche está en todo momento en esa posición, sin embargo hay que recordar que este método lo llamamos dentro de el pushMatrix y popMatrix y que la posición del coche se hace provisionalmente 0,0 sin embargo una vez fuera de estos, vuelve a tener las coordenadas X,Y en relación a el proyecto.

```
public void dibujarCar()
{
  // Creación del coche
  noStroke();
  fill(34);
  rect(0, 0, 70, 50);

  fill(250);
  rect(0, 0, 35, 50);
}
```

### Método MoverDelante

Como podemos ver la X y la Y la sacamos con la multiplicación de cos y sen de los grados que rota y le suma la x e y en la que se encuentran.

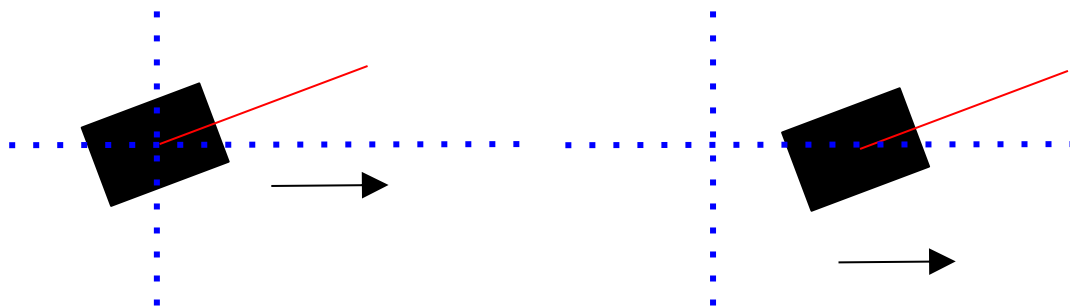
```

public void moverDelante()
{
    // Mover coordenada X e Y
    x = speed * cos(radians(rotacion)) + x;
    y = speed * sin(radians(rotacion)) + y;

    // Cuando el coche se salga de la pista volverá a la posición original
    if (x>width || x<0 || y>height || y<0)
    {
        x=posInix;
        y=posIniY;
    }
}

```

De caer en la idea de x,y incrementar ++, el resultado sería erróneo y saldrá lo siguiente... El vehículo se movería en relación al eje de coordenadas y no al de la dirección del coche.



#### Método Parar

Método tan simple como indicarle que la velocidad es 0 de tal manera que cuando el coche recibe esta variable no avanza.

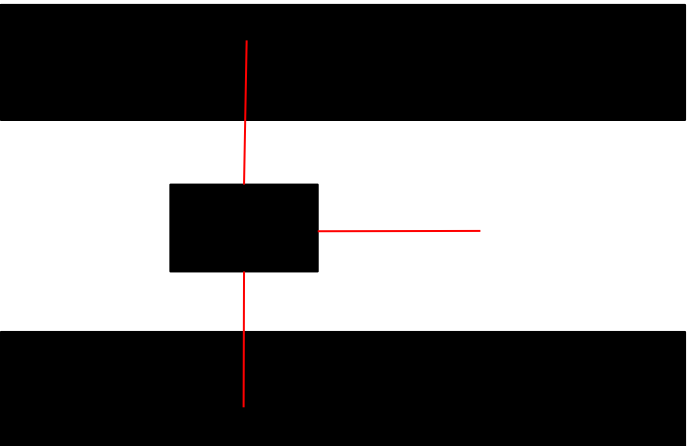
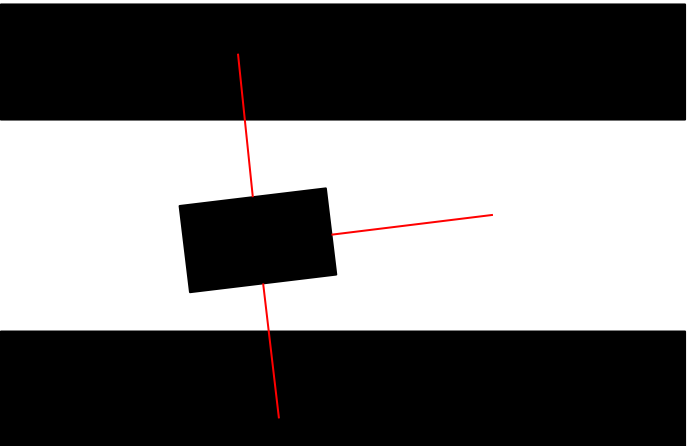
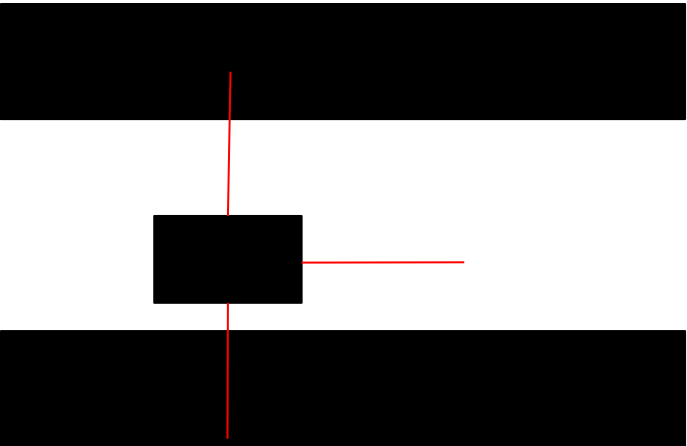
```

public void parar()
{
    if (stopRadar==true) {
        speed=0;
    }
}

```

#### Método CentraCoche

Este método lo que hace simplemente es recoger las distancias de los radares laterales, pasarlos por parametro al metodo centrarCoche de la clase Giro y este último nos devolverá el giro necesario para poder centrar el coche. A su vez indicamos los datos por pantalla.



```

public void centrarCoche(float x, float y, float rotacion)
{
    //RadarIzquierdo
    float distanciaIzq = radarIzquierdo(x, y, rotacion);

    //RadarDerecho
    float distanciaDer = radarDerecho(x, y, rotacion);

    //CentrarCoche
    int respuesta=giro.centrarCoche((int)distanciaIzq, (int)distanciaDer);
    giro.Rotar(respuesta);

    //Informacion para mostrar
    fill(0);
    textSize(10);
    text("Distancia Izq = "+distanciaIzq, width-210, 25);
    text("Distancia Der = "+distanciaDer, width-210, 40);
}

```

### Método radarIzquierdo

En este método simplemente recogemos todas las funciones del radar Izq. Como podemos ver primero pasamos al método recibirDatos de la clase Radar, la información del coche para posteriormente a través del método dibujarVector poder mostrar el radar y con método getPixel contar los pixeles del mismo radar. Nota, como se puede ver para poder iniciar el vector izq le pasamos la rotación -90°.

```

public int radarIzquierdo(float x, float y, float rotacion)
{
    rIzq.recibirDatos("rIzq", x, y, rotacion-90, 100);
    rIzq.dibujarVector();
    int pixelesBlancosIzq=rIzq.getPixeles();
    return pixelesBlancosIzq;
}

```

### Método radarDerecho

Este método actúa de igual manera que el anterior pero con el radar Derecho.

```

public int radarDerecho(float x, float y, float rotacion)
{
    rDer.recibirDatos("rDer", x, y, rotacion+90, 100);
    rDer.dibujarVector();
    int pixelesBlancosDer=rDer.getPixeles();
    return pixelesBlancosDer;
}

```

### Método radarFrontal

Radar frontal nos permite através de un recuento de píxeles saber si hay obstáculos delante del vehículo.

Primero enviamos los datos al método recibirDatos de la clase Radar, este se encarga de mirar los píxeles que tiene delante con un cierto rango y seguidamente devolverá un booleano indicando si ha encontrado un obstáculo próximo o no, para que pueda parar o seguir adelante.

Apretando las teclas s y d hacer que pare o siga avanzando el coche.

```
public void radarFrontal(float x, float y, float rotacion)
{
    rFrontal.recibirDatos("RadarFrontal", x, y, rotacion, 100);
    stopRadar=rFrontal.iniciarRadar();

    if (stopRadar || key == 's')
    {
        parar();
    } else if (!stopRadar || key == 'd')
    {
        // Mover cocher
        speed=speedAux;
        moverDelante();
    }
    text("Obstáculo = "+stopRadar, width-210, 55);
}
```

### Método Datos

Este método muestra datos de los controles por pantalla.

```
public void datos()
{
    //Datos Coche
    text("Controles s=parar, d=continuar", width-210, 70);
    text("Up=rotacioDer, Down=rotacionIzq", width-210, 85);
}
```

### 3. CLASE GIRO

La clase tendrá como función indicar el número de rotación que ha de realizar el coche,

#### Método CentrarCoche

Este método es el que realiza la valoración de si ha de girar a la izquierda o a la derecha en función de las distancias de los dos radares laterales. Si hay mas distancia de separación del lado izquierdo que al derech, este girara a la izquierda para compensar la separación y viceversa, Una vez elegido el giro envía la ordenar a través de un número que posteriormente el método Radar interpretará.

```
public int centrarCoche(int pbi, int pbd)
{
    int respuesta=0;
    print("\nPBI =" + pbi + " PBD =" + pbd);
    if (pbi > pbd)
    {
        print("\nRotaIzq");
        respuesta=3;
    } else if (pbi < pbd)
    {
        print("\nRotaDer");
        respuesta=4;
    } else
    {
        respuesta=0;
    }
    return respuesta;
}
```

#### Método Rotar

La clase giro recibe por parámetro una respuesta y dependiendo de esta realiza una rotación más amplia o menos y en una dirección u otra.

Un ejemplo, como aparece arriba, a la hora de centrar el coche en una calle, este nos devolvía dos números 3,4 según el giro a realizar y las pasamos a este método que devuelve el valor de la rotación.



```
public float Rotar(int respuesta)
{
    if (respuesta==0)
    {
        rotate=0;
    } else if ( respuesta==1)
    {
        valorRotacion=3;
        rotate=rotate-valorRotacion;
    } else if (respuesta==2)
    {
        valorRotacion=3;
        rotate=rotate+valorRotacion;
    } else if (respuesta==3)
    {
        valorRotacion=0.6;
        rotate=rotate-valorRotacion;
    } else if (respuesta==4)
    {
        //Rotacion para centrarCoche
        valorRotacion=0.6;
        rotate=rotate+valorRotacion;
    }
    return rotate;
}
```

### Método giroTeclado

Este método nos permite controlar los giros a través de las flechas del teclado UP, DOWN y poder realizar pruebas.

```
public float giroTeclado()
{
    // Control de la rotación por teclado...
    if (keyPressed == true ) {
        if (keyCode == DOWN)
        {
            //devuelve rotación para girar a la Derecha
            rotate=rotate+valorRotacion;
        } else if (keyCode==UP)
        {
            //devuelve rotación para girar a la Izquierda
            rotate=rotate-valorRotacion;
        }
    }
    return rotate;
}
```

#### 4. CLASE MAPA

La clase mapa contiene dos métodos a los cuales le pasaremos un string con el nombre de la imagen del mapa a mostrar.

##### Método mostrarMapaImagen

La clase mostrarMapaImagen va a un directorio y busca la imagen con el nombre del String pasado por parámetro. Seguidamente la muestra.

```
public void mostrarMapaImagen(String mapa)
{
    //creamos objeto Imagen
    PImage laFoto = loadImage("\\ImágenesMapas\\"+mapa);
    //Mostramos la imagen y la colocamos en la posición 0,0
    image(laFoto, 0, 0);
}
```

##### Método mostrarMapaProcessing

La clase giro está diseñada para que al igual que el anterior método, al pasarle un string este nos devuelva un mapa. La diferencia es que no nos mostrará una imagen, sino, un background diseñado desde processing. Aquí solo vemos un mapa creado, sin embargo pueden diseñar más.

```
public void mostrarMapaProcessing(String mapa)
{
    switch(mapa)
    {
        case "mapa1":
        {
            noStroke();
            fill(0);
            rect(0, 0, 1200, 250);
            rect(0, 400, 1200, 250);
        }
    }
}
```

##### Cargar imagen desde una cámara (Problema)

Una de los objetivos que se nos ha planteado pero que aún no hemos podido resolver es demostrar que el coche es realmente autónomo. Esto se puede probar pasándole a processing una imagen que hacemos con una cámara de un circuito dibujado en una hoja de papel.

Por el momento el resultado no se ha podido lograr puesto que aunque sí que hemos podido cargar la fotografía, esta, aunque se trate de una hoja en blanco, no contiene realmente los píxeles en blanco, por lo que el radar no sabría reconocer con exactitud los píxeles de la imagen.

Una posible solución en la que estamos trabajando es poder tratar la imagen y conseguir reducir la escala de grises y tener una imagen completamente en blanco y negro.

## 5. CLASE RADAR

La función del radar básicamente es la de devolver la distancia a la cual encontramos un obstáculo en relación al coche.

Para lograrlo lo que hacemos es dibujar una línea imaginaria delante del coche que lo que hará es recorrer cada píxel de dicha línea y seguidamente un recuento de cuántos píxeles blancos (camino libre) y cuántos píxeles negros (calles/obstáculos) encuentra. Una vez hecho el recuento, devolveremos la información de tal manera que el coche sepa qué acción tomar, si girar, parar, centrar el coche o seguir adelante.

### Constructor recibirDatos

A través de este constructor recibimos los datos necesarios para poder dibujar el radar e iniciar sus funciones.

```
public void recibirDatos(String id, float x, float y, float rotate, float alcanceRadar)
{
    this.id=id;
    this.x=x;
    this.y=y;
    this.rotacion=rotate;
    this.alcanceRadar=alcanceRadar;
}
```

### Método iniciarRadar

```
public boolean iniciarRadar()
{
    stop=false;
    dibujarVector();
    getPixeles();
    return stop;
}
```

### Método dibujarVector

**DibujarLinea** nos permitirá ver el alcance del radar. Esto lo hacemos tomando dos puntos de referencia que como podemos ver en el dibujo son PuntoA (PA) y PuntoB (PB).

PA = (x,y)

PB = (vec1.X, vec1.Y)



```
public void dibujarVector()
{
    PVector vec1= calculoPuntosAlcance();
    stroke(250, 0, 0);
    line(x, y, vec1.x, vec1.y);
}
```

Sin embargo no es tan simple como parece ya que como observamos en el método para encontrar el Punto B llamamos a la función **calculoPuntosAlcance**.

### Método calculoPuntosAlcance

La importancia de este método está en que este el puntoB no siempre formará parte del eje de coordenadas o de ordenadas, puesto que el coche rotará, de tal manera que para conseguir dicho punto hay que hacer uso de trigonometría.

En nuestro caso con la clase Giro tenemos la rotación y con ello sabemos cuántos grados sobre la horizontal se ha girado el vehículo.

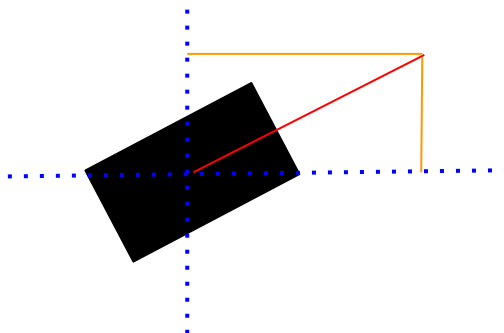
Seguidamente sabemos que el Cateto Opuesto (CO) o como llamamos en el método y1, es igual a el seno del ángulo multiplicado por la hipotenusa (alcance del radar) y le sumamos la y de la posición en la que se encuentra el coche.

```
public PVector calculoPuntosAlcance()
{
    float angulo = radians(rotacion);
    float x1=cos(angulo)*alcanceRadar+x;
    float y1=sin(angulo)*alcanceRadar+y;

    // Prueba del Punto
    /*noStroke();
    fill(0);
    rect(x1, y1, 1, 1);
    */

    // PuntoAlcanceMáximo
    PVector vec1 = new PVector(x1, y1);
    return vec1;
}
```

De igual manera hallamos el Cateto Adyacente que equivale a la x1, sacando el seno del ángulo y multiplicando por la hipotenusa y añadiendo la x de la posición del coche.



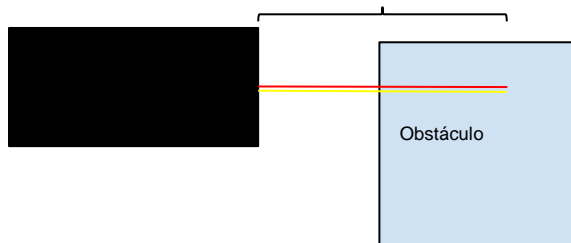
Dado que hemos pintado una línea roja como radar, si buscamos los píxeles que tenemos que se encuentran en la misma línea nos devolverá color rojo en todo momento. Es por eso que los píxeles que se toman son con misma dirección a los de la misma línea lo que le sumamos una altura más.

La zona a escanear se representa con la línea amarilla.

Un problema que nos ha consumido mucho tiempo ha sido a la hora de indicar la rotación, puesto que la rotación que pasamos es en grados, sin embargo la función cos/sen devuelve el resultado en radianes, por lo que el puntoB siempre era diferente. La solución a esto ha sido pasar a radianes la rotación, previamente, antes de hacer el cálculo cos/sen.

### Método getPixeles

Las distancia se devolverán con el método **getPixeles( )** que como se comenta arriba recuperará los colores para distinguir el blanco, camino libre, de otro , obstáculo.

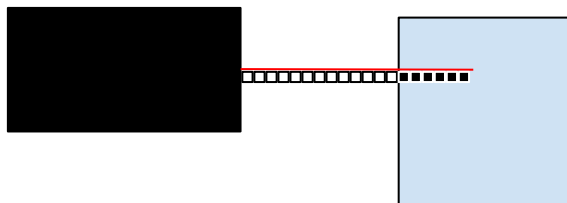


#### Explicación Gráfica 1

Como podemos ver si el radar tiene un alcance de 100 px. El resultado será:  
Blancos: 50px

Negros: 50px

Lo que significa que el vehículo esta a 50px de un objeto frontal.



#### Explicación Gráfica 2.

De 19px 13 son blancos y 6 son negros. Lo que da a entender que de el vehículo al obstáculo hay **13px de distancia**.

```

public int getPixeles()
{
    int i=0;
    int negro=0, blanco=0;
    float angulo = radians(rotacion);

    for ( i=34; i<alcanceRadar; i++)
    {
        float x1=cos(angulo)*i+(x+1);
        float y1=sin(angulo)*i+(y+1);
        color c = get((int)(x1+1), (int)(y1+1));

        //print("\n"+i+"--" + c);
        if (c<=-3735552)
        {
            negro++;
            //print("\nRadar: "+id+" Color"+ c +" x1 "+ x1 +" x1 " + y1);
        } else if (c>=-3733496 )
        {
            blanco++;
        } else
        {
        }
    }
    if (negro>=10)
    {
        stop=true;
    }
    print("\nID "+id+" Negros: "+negro+"\nBlancos: "+blanco);
    print("\n-----");
    print("\nRadar devuelve true Si hay un obstaculo =" +stop);
    return blanco;
}

```

Como vemos creamos un for que va desde el punto A del radar al punto B, y coge cada píxel de la misma recta.

Seguidamente con un contador comprobar cuántos son negro y cuantos blancos.

Al final vemos como hace un return de los píxeles blancos que ha encontrado, para que cuando llamemos al mismo método, sepamos cuánta distancia hay hasta encontrar un obstáculo y saber si seguir adelante o parar el vehículo.

### Detectar Píxeles

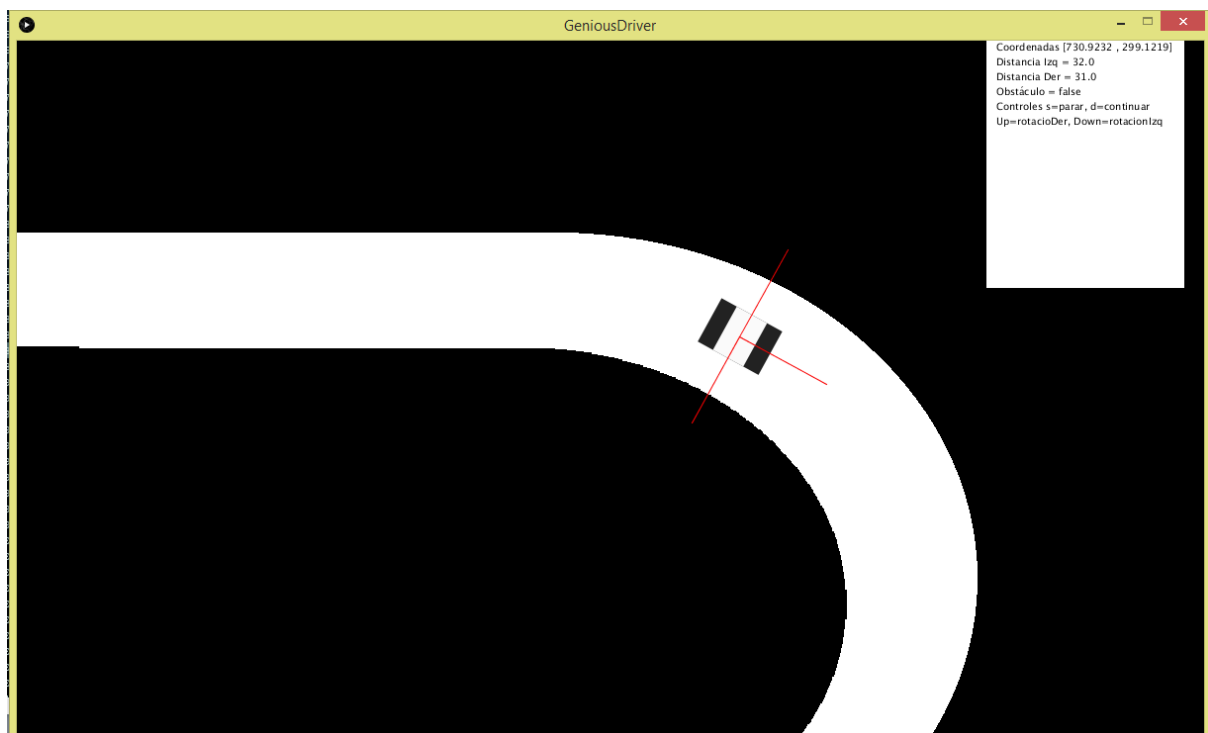
Un problema que nos ha llamado la atención es que pese a haber creado el background a partir de bloques negros y fondo blanco, hay momentos que en cuando hacemos un getColor, que nos permite conocer el color del píxel, este a veces muestra valores que no son ni negro ni blanco. Sin embargo no supone un problema ya que solo ocurre en contadas veces por lo que el método getPixel del radar no pierde eficacia.

### Pruebas

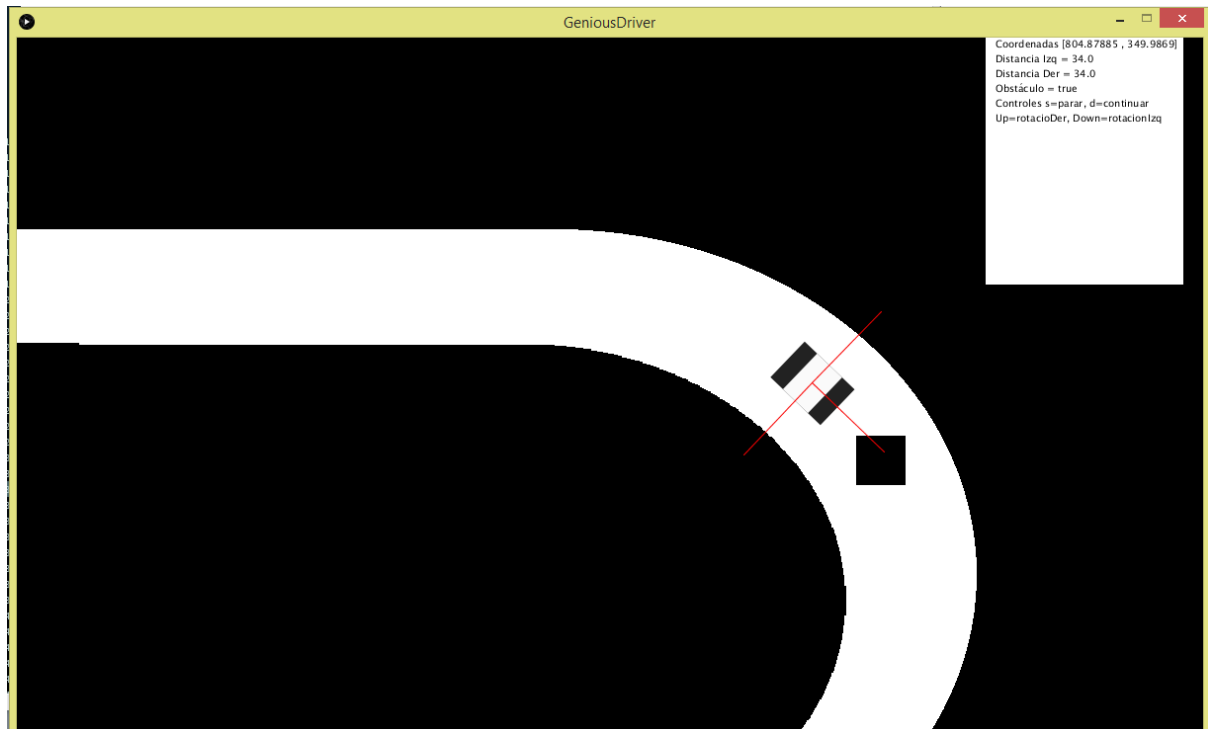
En esta imagen vemos como el coche se encuentra totalmente centrado ya que los radares laterales devuelven una distancia de 25px con relación al borde de la calle. También vemos que no hay ningún obstáculo por lo que la misma variables se encuentra en false.



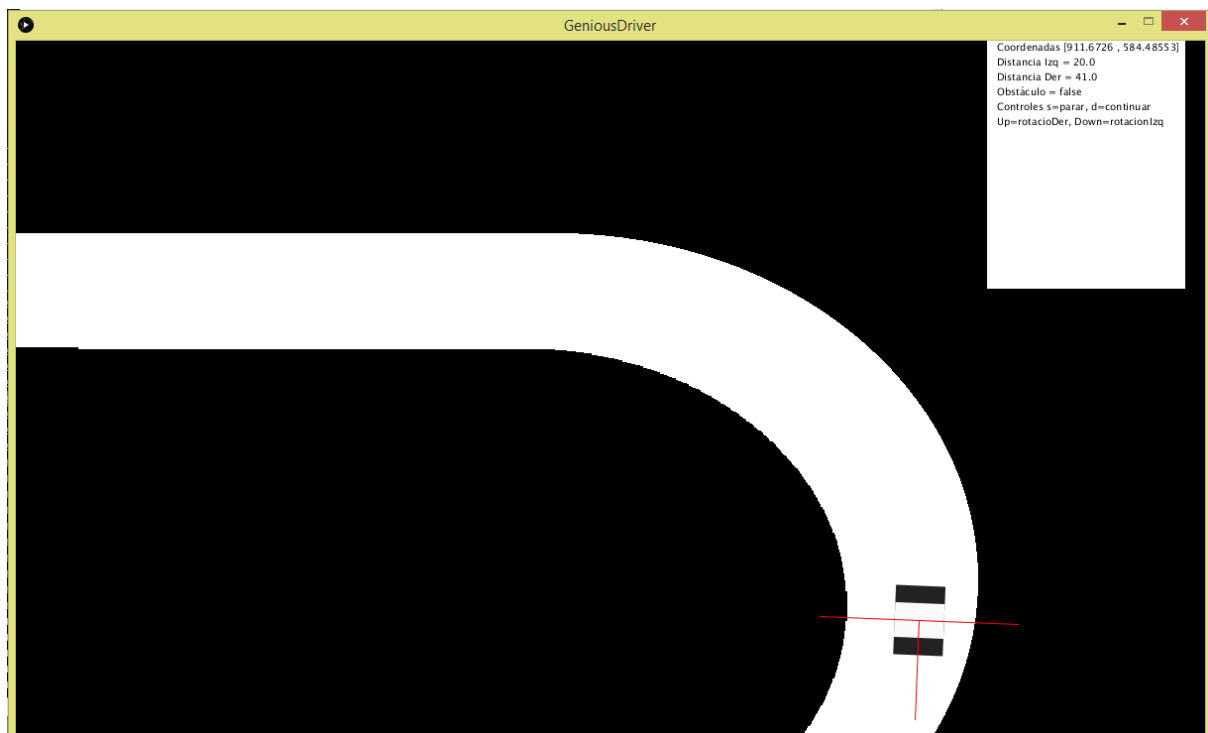
En esta otra imagen vemos como empieza a realizar el giro, que básicamente es ir centrando el coche a medida que avanza. Como se puede ver la distancia de los radares son similares.



En este fotograma hemos incluido un obstáculo que el radar Frontal ha detectado, por lo que se detiene y se puede comprobar como la variable obstáculo ha pasado a true.



Aquí vemos que si retiramos el obstáculo el coche procede con la marcha y sigue girando para completar la curva.



### Conclusión

Hasta el momento hemos conseguido recrear un coche autónomo que es capaz de circular por un circuito y es capaz de detectar distancias y actuar según ellas, ya sea seguir



avanzando, parar o girar. Sin embargo desde Genius Driver estamos trabajando para afinar más lo que hay programado hasta el momento,hacerlo más eficiente y poder solucionar los puntos donde nos hemos quedado atascados.

Además estamos pensando en acoplar otras funciones como las de detectar semáforos y poder probar de interactuar con otros coches.

Si tuviéramos que decantarnos por el lenguaje que más nos ha costado afrontar podríamos decir sin duda Android y PHP. Aunque cabe al fin y al cabo que todos los lenguajes son iguales en cuanto a

## WEBGRAFIA

<https://es.stackoverflow.com/>

<https://www.google.es>

<https://www.youtube.com>

<https://developer.android.com/index.html>

<https://es.wikipedia.org/>

<http://stackoverflow.com/questions/2986627/returning-php-multi-dimension-array-to-javascript-ajax>

<http://stackoverflow.com/questions/2628798/print-array-to-a-file>

<http://stackoverflow.com/questions/15462031/jsonarray-to-spinner>

<http://stackoverflow.com/questions/867518/how-to-make-an-android-spinner-with-initial-text-select-one>

<http://stackoverflow.com/questions/8456835/how-to-disable-action-bar-permanently>

<http://stackoverflow.com/questions/28954445/set-toolbar-title>

<http://stackoverflow.com/questions/5329542/php-mysql-insert-null-values>

<http://stackoverflow.com/questions/3802256/mysql-update-multiple-columns>

<https://es.stackoverflow.com/questions/9677/obtener-valores-de-array-dentro-de-otro-array-php>

<http://stackoverflow.com/questions/3075009/android-how-can-i-pass-parameters-to-asynctasks-onpreexecute>

<http://stackoverflow.com/questions/41906204/java-how-to-send-images-from-a-restful-web-service>

<http://stackoverflow.com/questions/15219007/change-folder-permission-to-777-using-php-temporarily>

<http://stackoverflow.com/questions/41906204/java-how-to-send-images-from-a-restful-web-service>

<http://picarcodigo.blogspot.com.es/2014/05/webservice-subir-imagen-servidor-desde.html>

<http://stackoverflow.com/questions/1337424/android-spinner-get-the-selected-item-change-event>

<http://stackoverflow.com/questions/13132535/mysql-duplicate-entry-error-even-though-there-is-no-duplicate-entry>

<http://stackoverflow.com/questions/20713321/httpclient-getconnectionmanager-is-deprecated-what-should-be-used-instead>

<https://www.youtube.com/watch?v=OFIoifUm6tA>

<https://www.youtube.com/playlist?list=PLnWAzeXp9V4lX35XTr0EDG0tCJLHevnhO>

[https://www.youtube.com/channel/UC79o33I\\_T\\_jUMC6u9CeIbBw](https://www.youtube.com/channel/UC79o33I_T_jUMC6u9CeIbBw)

## ¿QUE APORTA EL TRABAJO HECHO?

Esta es la típica pregunta que deja al equipo de proyecto un poco descolocado siempre y cuando se pregunte en los inicios del mismo. Gracias a internet y a los conocimientos que nos brinda, hemos podido llegar a una gran conclusión. Nuestro proyecto parte de 3 conceptos básicos y fundamentales en el mundo de la carretera:

- Seguridad.
- Confianza.
- Control.

A medida que hemos ido ampliando el proyecto la respuesta a esa pregunta ha ido haciéndose más y más grande pero no nos dábamos cuenta, estábamos ciegos ante una respuesta evidente.

Hoy en día por desgracia un gran número de personas mueren al volante por culpa de una mísera distracción. Todos sabemos que una persona se puede distraer fácilmente con una simple mosca, ¿Vale, y que tiene que ver esto con nuestro proyecto?

Pues la verdad, se podría decir que más de lo normal. ¿Por qué no relacionamos los 3 conceptos básicos de nuestra aplicación? Ahí está!

Nuestro proyecto podría evitar un gran número de accidentes por culpa de una mísera distracción gracias a las nuevas tecnologías de las que dispone para poder tomar el control del mismo vehículo en un momento determinado.

Si nos ponemos a pensar lógicamente, si el conductor desde un principio sabe que dispone de un “robot” el cual es capaz de reaccionar y tomar decisiones en un momento crítico, el mismo podrá estar mas tranquilo.

Pero desgraciadamente ya podemos suponer desde un principio que todo puede fallar de un momento para otro, incluso el mismo “robot”. Sí, todos los sabemos. Hoy en día no hay nada perfecto, pero si tenemos una herramienta la cual puede evitar algo tan terrorífico como un accidente, quieras o no ayuda.

Si exprimiéramos mas todo el funcionamiento podríamos ver que nos estamos dejando de lado un tipo de transporte muy importante y muy poco fomentado últimamente, el transporte público. ¿Nunca te has preguntado cuantas veces para en la misma parada un autobús? Muchas, ¿verdad?

Genius Driver podría ser el complemento perfecto para fomentar el transporte público.. basta con que el “robot” memorice todas y cada una de las paradas para luego poder repetirlas automáticamente sin la necesidad de un piloto al volante.

Sin quererlo salen cada vez más cosas en las que aporta el proyecto al mundo actual. No solamente fomentaríamos el transporte público sino que también ayudaríamos a los minusválidos a poder utilizar el coche para poder ir de un sitio a otro sin tener que mover ni un solo dedo.

Hacer que el coche sea un método de transporte apto para todas las personas es nuestro principal objetivo en el proyecto.

¿Nunca te has imaginado que los coches se pudieran comunicar entre ellos para llegar antes al destino esperado?

Podría ser el fin de los atascos y el inicio de una nueva evolución.

Pero siempre se tiene que sacar una conclusión de los trabajos y en este caso ha sido la “gran pregunta” con múltiples respuestas válidas.

## OBJETIVOS

### **Objetivos no alcanzados:**

Uno de los principales objetivos que teníamos en mente, era poder implementarlo en un coche pequeño con un Arduino y este poder conectarse a una unidad de proceso que calculara todas las distancias y decisiones que debería tomar el coche. Otro de los objetivos el cual no hemos podido lograr por falta de tiempo lógicamente poder poner el coche en una pista pequeña y que este la recorriera sin problemas superando todos los obstáculos y respetando las señales de tráfico, pero al vernos limitados en el aspecto comentado anteriormente decidimos hacerlo en un emulador.

Otro objetivo que también teníamos pensador hacer también, poder poner una cámara web encima de la maqueta y separar el “mapa” por pixel y decirle exactamente a que pixel queremos que se dirija el coche. Básicamente definirle un destino previo. O también para saber donde esta situado el coche en el mapa exactamente.

Hablando del tema de la cámara web al principio también dijimos de poder dibujar en una hoja en “directo” el circuito que tiene que realizar el coche, pero de momento solo nos ha dado tiempo a hacerlo con una fotografía.

Ponerle una cámara web frontal al coche y que este retransmita en directo al usuario todo lo que esta viendo a través de una pantalla. Ya sea por un ordenador, tableta, móvil, etc. Como hemos explicado antes al no hacer la maqueta de coche lógicamente no hemos podido lograr el objetivo de poner un detector óptico delante del coche. El cual su función principal es hacer de radar de proximidad.

## OBJETIVOS ALCANZADOS

El coche ya es capaz de ir por una pista simulada sin chocarse y sin salirse del circuito.

Otro objetivo conseguido sería el de poder hacer una foto al mapa y que el coche simulado por Processing pueda interpretarlo como una pista y recorrerlo como si de una carretera se tratase.

Poder saber qué es lo que tiene delante y pararse si fuera necesario gracias a los radares de los que dispone. Y en caso de haber parado, reanudar la marcha cuando el obstáculo haya desaparecido.

## AMPLIACIONES

Principalmente lo que nos gustaría ampliar es el tiempo, pero como no es posible a continuación explicaremos las posibles ampliaciones que habríamos hecho en caso de disponer del mismo:

- Como hemos comentado anteriormente algo que nos entusiasmaba más de la cuenta era la de la cámara web. Ya que era una buena idea poder saber dónde está situado el coche en todo momento y poder decirle hacia donde tiene que ir de nuevo.
- Que el coche pueda reconocer las señales de tráfico y así poder respetarlas.
- Poder arreglar el comportamiento que tiene el coche cuando se encuentra con un cruce.
- Hacer la maqueta del coche y ponerle la tarjeta de Arduino y hacerlo mas realista en una maqueta de verdad (es una ciudad pequeña con calles).
- Si se hubiera logrado lo de la maqueta del coche, poder ponerle una cámara web como he dicho anteriormente en los objetivos incumplidos. De esta forma el coche seria como más “interactivo” con el usuario y así en todo momento podríamos ver la visa del “conductor”.
- Ponerle un sensor óptico al coche para poder calcular a la distancia que tiene un obstáculo.



## ASPECTOS NEGATIVOS

### **Grupo:**

El primer aspecto negativo claramente es la falta de tiempo, ya que un año es poco si se quiere desarrollar un buen proyecto en perfectas condiciones.

El otro aspecto negativo ha sido el trabajo en grupo, no ha ido del todo bien en el proyecto, por algunas razones. Una de las razones ha sido el desconocimiento de la programación por parte de dos miembros del grupo al no haber aprobado la programación básica de primero, enfrentarse a un proyecto donde el 99% es programación orientada a objetos eso ha dificultado mucho la distribución del proyecto y a tener que asumir el mando de los demás para que el proyecto no se ahogara el primer mes siendo sinceros.

Y cuando un grupo no funciona correctamente siempre suele salir todo negativamente desde la relación con los miembros del grupo hasta el trabajo a realizar en sí.

Trabajar en grupo suele ser bueno porque la faena se reparte y es menos pesada, pero cuando ves que no puedes repartirla porque los demás desconocen el lenguaje o no saben cómo afrontarlo es difícil y se te viene un mundo encima y lo primero que piensas es en el tiempo que tiene para poder acabar la faena de dos miembros en el tiempo disponible de 1 año.

Como he explicado antes, si no has aprobado la programación básica hacer el proyecto es un sacrificio, eso significa que te tienes que aplicar al 100% y eso a veces es un problema si tienes otras asignaturas ya que no tienes que dejarlas de lado tampoco.

### **Proyecto:**

Hemos tenido bastantes problemas en cuanto al lenguaje de programación. No haber visto depende de que lenguajes en clase y utilizados en proyecto, eso ha dificultado mucho la detección de errores en el código. Todos sabemos que cuando uno hace un programa, casi nunca funciona a la primera. Pero luego buscas el simple error y te das cuenta de que no era más que un simple “punto y coma”.

El principal problema el cual nos hizo retrasarnos bastante y perder más de un mes del proyecto, fue el tema del servidor web. Este tema era totalmente desconocido por parte de todos los miembros del equipo, así que tuvimos que buscar primero su

funcionamiento y luego, el cómo aplicarlo al proyecto. Fueron pasando los días hasta que fuimos entendiendo un poco más del tema del servidor web y su gran importancia en el proyecto, “más vale tarde que nunca”. Una vez entendido el funcionamiento volvimos a buscar otra vez la información de cómo poder “crear” dicho elemento. Encontramos muchos videos y mucha información, pero nada suficientemente claro y adaptable a nuestro proyecto. Al estar el pleno siglo XXI todas las tecnologías cambian radicalmente todos los días y ese fue nuestro “infierno”. Encontrábamos una manera de hacerlo pero después veíamos que la manera de hacerlo se había quedado obsoleta por completo y no nos interesaba, porque tener luego problemas de incompatibilidad por otro lado era nuestro miedo en un futuro.

Los protocolos disponibles que encontramos fueron:

- SOAP.
- REST.

<b>Ventajas REST</b>	<b>Ventajas SOAP</b>
· Pocas operaciones con muchos recursos	· Muchas operaciones con pocos recursos
· Se centra en la escalabilidad y rendimiento a gran escala para sistemas distribuidos hipermedia	· Se centra en el diseño de aplicaciones distribuidas
· HTTP GET, HTTP POST, HTTP PUT, HTTP DEL	· SMTP, HTTP POST, MQ
· XML auto descriptivo	· Tipado fuerte, XML Schema
· Síncrono	· Síncrono y Asíncrono
· HTTPS	· WS SECURITY
· Comunicación punto a punto y segura	· Comunicación origen a destino seguro

Están son las principales diferencias que tiene cada protocolo.

La primera vez que lo hicimos fue en SOAP, pero claramente tuvimos más de un problema ya que vimos que se quedo obsoleto el protocolo. Aparte de carecer de muchos métodos como PUT, DELETE, GET, toda la información que encontrábamos no estaba actualizada y la mayoría poco fiable al 100%.

Así que nos decidimos afrontar directamente con el segundo protocolo REST. La primera vez que hicimos el servidor web fue desde un Framework llamado “CodeIgniter Rocks”, pero realmente aun estábamos probando ya que no sabíamos a la perfección como íbamos a afrontarlo.

Después de estar “trasteando” en el Framework conseguimos que nos devolviera los usuarios que habían en la base de datos en formato JSON. Pero claro ahí fue donde nos estancamos, porque no sabíamos como haríamos las otra consulta, como por ejemplo el “registro” de Android o de web.

A sí que decidimos dejar el Framework de lado e intentar hacer el servidor web desde cero nosotros. Tuvimos que estar otras dos semanas buscando de nuevo como poder hacerlo.

Finalmente encontramos unos video que explicaban de cómo hacerlo y unos documentos también, así que decidimos hacer una “fusión” de lo encontrado porque era muy similar. Y de una vez por todas resolvimos el problema.

El desconocimiento siempre causa problemas y lo peor es cuando tenías la solución ante tus ojos hace tiempo.

Si resumimos los problemas más grandes que hemos tenido, aun queda uno que nos desquicio totalmente, android.

El gran problema que tuvimos fue el de subir la imagen al servidor. Pero como dijimos antes nada sale a la primera. Encontramos poca información referente a nuestro caso, porque todos los video que encontrábamos lo hacían de diferente manera o de diferente forma la cual no podíamos hacer nosotros.

Lo que hacíamos era enviar la imagen por un lado al servidor y el nombre de la imagen por otro lado para poder insertarlo en la base de datos.

Y aquí viene el gran problema. Sin saber la razón, el nombre siempre lo insertaba en la base de datos pero en cambio la imagen en el servidor había veces que si otras veces que no y no sabíamos el porqué. Este tema nos desquicio muchísimo porque no hay nada peor que no encontrar el problema. Teníamos todo bien pero alguna razón había veces que subíamos una fotografía y funcionaba y otras no.

El problema podía estar relacionado desde la resolución de la fotografía, error al enviarla o permisos en el terminal, etc.

Nosotros deducimos que tenía algo que ver con los permisos de escritura en la tarjeta interna del terminal, porque en el emulador android “BlueStacks” funcionaba.

Entonces llegamos a la misma conclusión que la del servidor web, hacerlo de nuevo otra vez y que funcionara a la perfección. Y así fue.