PROYECTO RENTING

Entornos de Desarrollo

Programación

Base de datos

ALEJANDRA IZAGUIRRE TÉBAR

JOSE LUIS LÓPEZ

ALEJANDRO COBO YERA

RUBÉN NÚÑEZ VALLE

ROLES Y NORMAS

ROLES:

Rubén Núñez → Developer
José Luis Gómez → Developer
Alejandro Cobo → Developer
Alejandra Izaguirre → Developer, Scrum Master

NORMAS:

1. Hacer copias de seguridad de la base de datos si estamos trabajando en funciones que la alteren.
2. Tener una copia de seguridad de nuestro trabajo aunque esté subido en gitlab.
3. La comunicación debe ser fluida y continua entre los miembros, tanto para pedir ayuda como para proponer ideas o cambios en el proyecto y su organización.
4. Debemos reunirnos algunas veces para trabajar y programar juntos.
5. DEBEMOS DOCUMENTAR NUESTRO CÓDIGO para que todos los miembros del equipo podamos entenderlo, probarlo, mejorarlo o ayudar en su implementación.
6. NO SUBIR NADA A GITLAB EN RAMA O EN MASTER SI TENEMOS DUDAS SOBRE EL FUNCIONAMIENTO DE GIT

Relación de documentos:

Sobre GitLab: Pasos concretos a seguir para uso y subida del repositorio.

Git_Windows: Instalación de git en el sistema operativo windows

Ramas y trabajo: Ampliación de comandos para gitlab

Sobre_contrato_y_facturación_y_paneles-dialogos: Planteamiento e información sobre el contrato, la facturación y los paneles de dialogo.

Sobre_mysql: Uso de la base de datos e instalación de mysql.

Estructura de carpetas: Como su propio nombre indica , se trata de un documento en el que se explica la estructura de carpetas y algunos valores de los distintos modelos.

Conexión_java_con_Sql: Como hacer la clase conexión con mysql.

Generar_PDFs: Explicación sobre como generar la factura en pdf a partir de un xml.

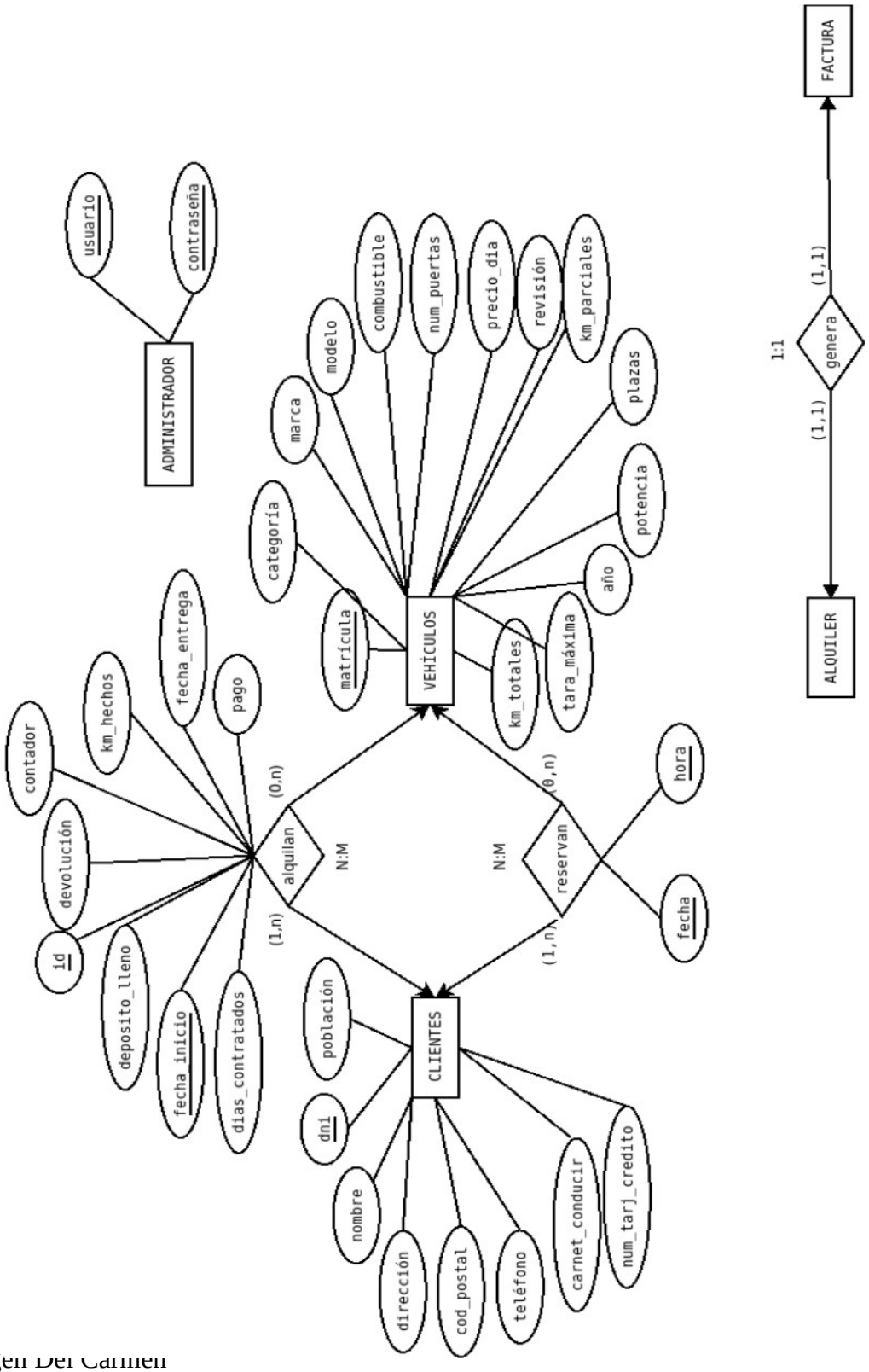
Análisis de requisitos:

- Partimos para empezar de una generalización “vehículo” que se divide en “turismo” y “transporte” los cuales tienen una serie de atributos por los cuales se les puede buscar.
- Cuando se compra un nuevo vehículo este debe registrarse en la aplicación lo cual debe existir una clase “VehiculoDAO” que tenga una función que inserte un vehículo a la base de datos.
- Cada vehículo tendrá un precio de alquiler por día que se refleja en una hoja de calcula, la cual sería conveniente volcarla a la base de datos y trabajar desde la misma.
- Solo se pueden contratar con hasta un máximo de 5 días, lo cual habrá que mostrar al cliente un desplegable del 1 al 5 para seleccione. Solo se podrá ampliar el plazo 3 veces, lo que significa que habrá un contador de ampliaciones para comprobar que no excede el máximo.
- También según el número de kilómetros que contrate subirá el precio del alquiler, aunque habrá una rebaja por cada día de más que se alquile. Todas estas condiciones irán sumando una cantidad de dinero al total del precio del alquiler.
- Tendremos una clase cliente con todos sus atributos, los cuales también se verán reflejados en la base de datos. Estos clientes podrán alquilar nuestros vehículos, pero dejando una fianza anteriormente, la cual será revisada por el administrador de turno y confirmará la operación ha sido realizada (la aplicación se desentiende de esto de momento).
- El administrador podrá añadir, modificar y eliminar todo tipo de vehículo y cliente, pero en ningún caso podrá eliminar el historial o los contratos en curso.
- La aplicación avisará cuando haya cumplido el contrato de un alquiler y el vehículo no haya sido devuelto. Esto se comprobará cada vez que se inicie el programa y todos los contratos que estén atrasados tendrán un recargo de renovación automáticamente que se sumará al total del precio del alquiler.
- Una vez entregado el vehículo se comprobará que el depósito está lleno, ya que así lo recibió, y en caso contrario se le añadirá un cargo al precio total. También se comprobará que el coche está en buen estado, en caso contrario habrá más cargos el precio total. Por último, se cobrará el precio total del alquiler y si todo lo anterior se ha cumplido con éxito se le devolverá la fianza al cliente (No nos fiamos de nadie).
- Habrá una serie de opciones que el administrador marcará como correcto si todo lo anterior se cumple adecuadamente. Al finalizar se guardará toda la transacción en el historial.
- Los vehículos podrán alquilarse si están disponible. Si un cliente devuelve un coche, hasta después de una hora el siguiente cliente no podrá llevárselo ni reservarlo ya que se tendrán que revisar los desperfectos y el depósito. Pasado 2 horas de la reserva del cliente, si aún no ha venido a por el coche se cancelará la reserva. En cualquier caso, el administrador podrá crear, modificar o eliminar cualquier reserva.
- A todo cliente se le entregará un documento con los datos del alquiler.
- Se actualizarán siempre los kilómetros de todos los vehículos a la entrega ya que la aplicación deberá avisar al administrador cuando superen los 20.000km para hacerle la revisión correspondiente. Para ello haremos una función encargada de ello.
- De momento la aplicación está limitada a una sola oficina.

Tabla comparativa requisitos, objetivos y funciones:

REQUISITOS	OBJETIVOS DEL EQUIPO	FUNCIONES IMPLEMENTADAS
Facturación controlando descuentos, cargos y precio de los vehículos	Generar factura manualmente	**Implementada** **Realización manual**
Añadir, modificar, dar de baja, clientes, vehículos, contratos de alquiler, reservas y facturas	Añadir, modificar, dar de baja todas las secciones	**Implementadas**
Auto cancelación de reservas tras dos horas	Avisar de la caducidad de la reserva guardando la hora.	Las reservas tiene atributo hora pero no controlan el período que llevan insertadas
Avisos de cumplimiento de días de contrato, y no devolución de vehículo	Implementar avisos por incumplimiento de condiciones	No genera ningún aviso
Generar documento por contrato, reserva y factura	Generar PDFs con toda la documentación	**Implementadas**
Controlar número de kilómetros totales y parciales de los vehículos para generar avisos de revisión	Controlar el número de kilómetros realizados	**Implementados**
Inclusión de imágenes de los vehículos	Incluir imágenes si nos daba tiempo al final de lógica	**No Implementados**
	Ordenar registros de la tablas, mediante botones preparados para ello	**No implementado**

Diagrama de E-R.



Diagramas de Clases.

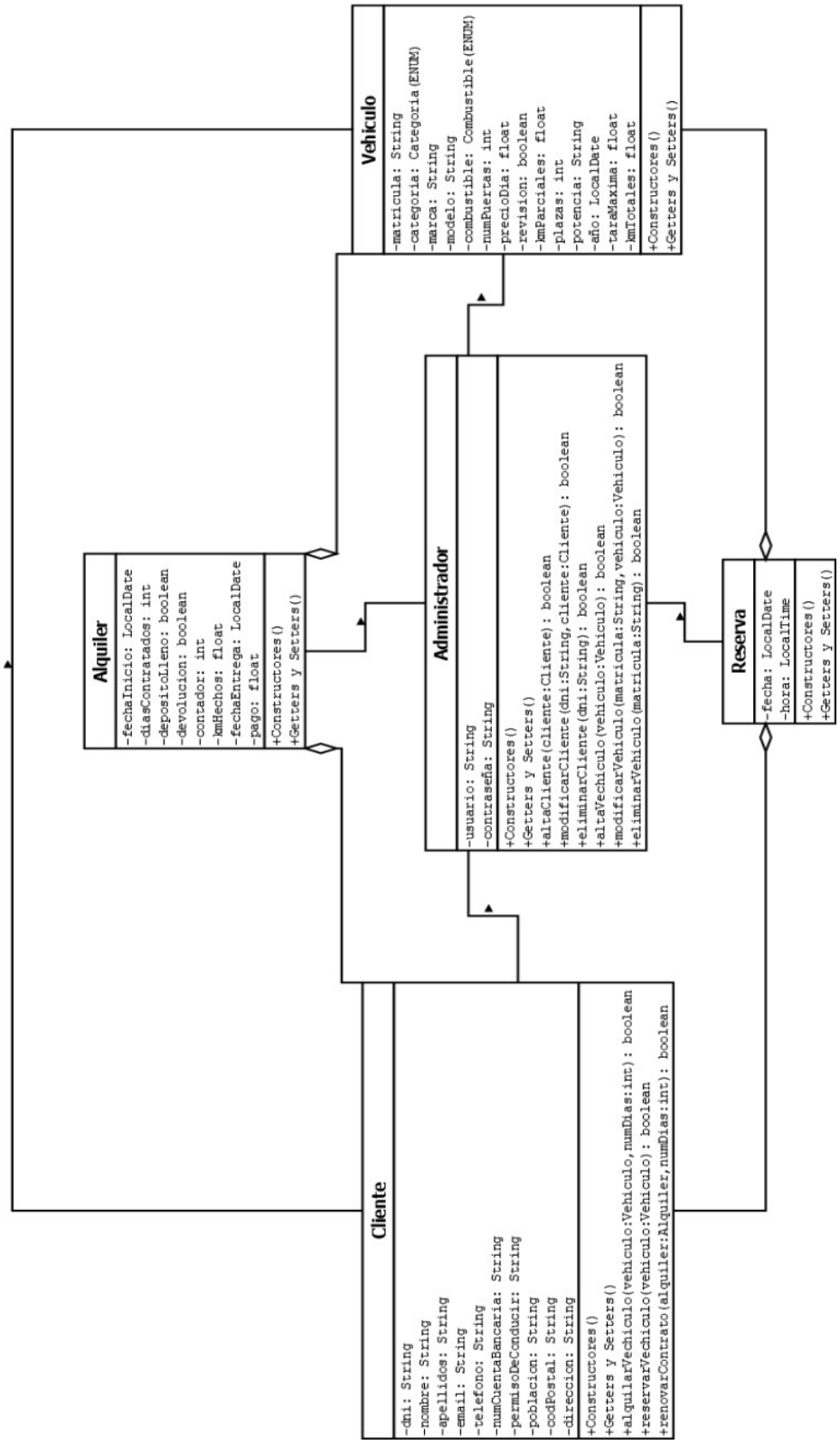
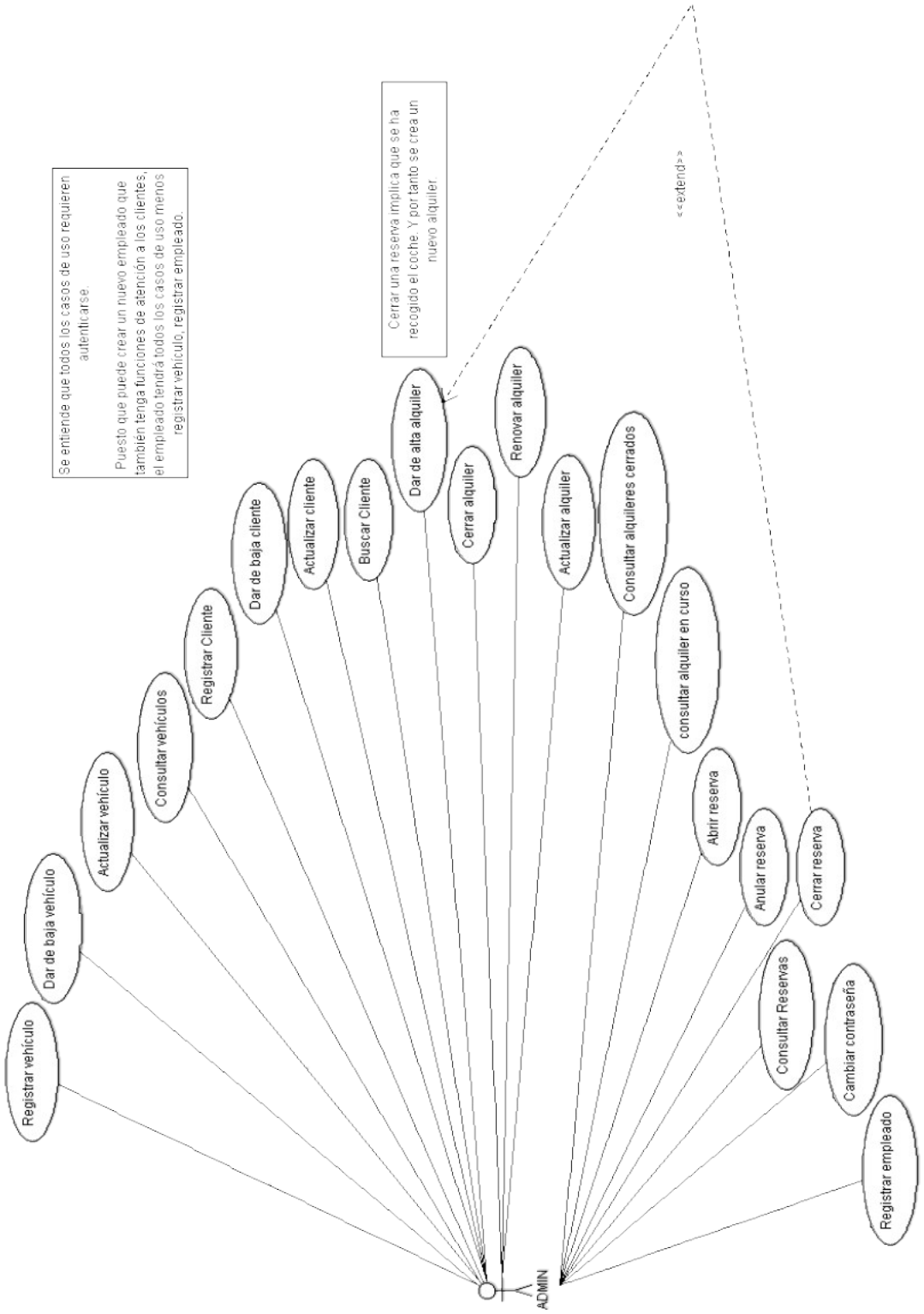


Diagrama de Casos de Uso.



Diario de Dificultades.

A lo largo del desarrollo del proyecto nos hemos encontrado con algunas dificultades, que hemos resuelto en la medida de nuestras posibilidades.

Primero consideramos necesario destacar que al principio, en los primeros sprints, creímos que la implementación de las ventanas con sus correspondientes controladores sería más sencillo. No hemos podido implementar características necesarias del proyecto por falta de tiempo, debido a un exceso de confianza en las primeras fases. La realidad de la integración del modelo vista-controlador nos ha supuesto un parón en el desarrollo.

Entre las principales dificultades se encuentra la imposibilidad de usar docker por todos los miembros del equipo, llevándonos al uso de mysql directamente y utilizando la base de datos de forma local, cada uno en nuestro terminal.

Durante dos semanas estuvimos “atascados” en la lógica de negocio en cuanto a lo que a la factura y contratos se refiere.

Como la teoría sobre polimorfismo e interfaces la hemos aprendido en clase muy tarde en comparación a cuando la necesitábamos, todos los miembros del equipo hemos “comprendido” cómo generar las vistas adecuadamente en apenas unos pocos días y justo antes de la entrega del proyecto.

Nos hubiera gustado poder probar las diferentes características de JSwing más profundamente. Precisamente , para que la aplicación tuviera una apariencia más agradable, y acorde con lo que nos imaginábamos.

En la carpeta Documentation, dentro de Equipo, se pueden encontrar los distintos ficheros informativos que hemos utilizado para mantener un trabajo común sin problema de versiones ni de decisiones(Como se puede observar en la relación de documentos de la página 3) .

En la carpeta Database , se encuentran los ficheros relacionados con la creación de la base de datos y la inserción de datos de prueba. El fichero sql correcto es el denominado CREATE_TABLES_ALTERNATIVO.sql que contiene también la tabla facturas.

División del trabajo:

Cada miembro del equipo tenia asignado a una sección del proyecto (cliente,vehículo,contrato,reservas,factura) en cuanto a código.

Rubén Núñez Valle:

- Modelo cliente
- ClienteDAO
- Interfaz Cliente
- Vista cliente
- Controlador Cliente
- Modelo de tablas cliente
- Dialogo cliente

Jose Luis Gómez López

- Modelo contrato
- ContratoDAO
- Interfaz contrato
- Vista contrato
- Controlador contrato
- Modelo de tablas contrato
- Dialogo contrato

Alejandro Cobo Yera

- Modelo Vehículo
- VehículoDAO
- Interfaz vehículo
- Vista vehículo
- Controlador vehículo
- Modelo de tablas vehículo
- Dialogo vehículo

Alejandra Izaguirre Tébar

- Modelo Reservas
- ReservasDAO
- Interfaz reservas
- Vista reservas
- Controlador reservas
- Modelo de tablas reservas
- Dialogo reservas

- Modelo Factura
- FacturaDAO
- Interfaz Factura

- Vista factura
- Controlador factura
- Modelo de tablas factura
- Dialogo factura

Además las tareas de análisis, creación de la base de datos y de integración del modelo MVC, se repartieron de la siguiente forma:

Análisis:

- Diagrama Entidad-Relación : Alejandro Cobo Yera, Alejandra Izaguirre Tébar, Rubén Núñez Valle.

-Diagrama de clases : Jose Luis Gómez López.

-Diagrama de casos de uso: Alejandra Izaguirre Tébar.

- Creación de la base de datos: Alejandra Izaguirre Tébar, Rubén Núñez Valle, Jose Luis Gómez López, Alejandro Cobo Yera, cada uno la tabla correspondiente a la secciones indicada anteriormente.

-Integración del MVC: Alejandra Izaguirre Tébar.

- Desarrollo de la búsqueda : Jose Luis Gómez López.

- Investigación sobre añadir color,cambio de fuente,incluir imágenes: Alejandro Cobo Yera.

- Manual: Alejandra Izaguirre Tébar .

- Informe: Alejandra Izaguirre Tébar,Alejandro Cobo Yera,Rubén Nuñez Valle.

- Power point:Rubén Nuñez Valle,Alejandro Cobo Yera.