App Lector OBD

https://github.com/Hug0Lz/Gestor-Coches-App.git

Nome Alumno/a: **Hugo López Carballo**

Curso: ***2º DAM***

Módulo: ***Proxecto Final Ciclo***

Contido

1. Introducción

Co avance das tecnoloxías, os vehículos fanse máis complexos cada xeración. Máis sensores que permiten facer mellores diagnósticos, máis sistemas de seguridade... Isto fixo necesario avanzar na creación de sistemas electrónicos para os vehículos que os melloraron en moitos aspectos. Dende a mecánica con avances coma a inxección electrónica, a incorporación de tecnoloxías aplicadas a seguridade coma os controles de tracción electrónicos(ESP). Isto fixo necesario a creación de unidades que procesaran toda a información e xestionaran os diferentes sensores e actuadores, tendo lugar a aparición de centralitas (ECU). Procesadores centralizados que controlan diferentes aspectos dos vehículos. A día de hoxe, nun mundo tan dixitalizado, fixéronse comúns as melloras a nivel de interface de usuario e infoentretemento no vehículo, con cada vez pantallas máis grandes e interfaces máis elaboradas. Tamen foi así no caso das centralitas. En coches máis antigos podemos atopar centralitas simples para o motor, a transmisión e sistemas de seguridade. Pero a día de hoxe podemos contar con centos de centralitas dedicadas a diferentes aspectos do vehículo: sistemas de seguridade activa, controles adaptativos de cruceiro, sistemas de infoentretemento...

Co aumento da complexidade destos sistemas nace a necesidade de novas interfaces de comunicación cos vehículos, tanto para o seu mantemento como para a súa xestión. Isto lévanos a ver a necesidade dunha aplicación que nos permita interactuar co noso vehículo, vendo en directo diferentes parámetros e datos do mesmo, ademáis de ter conta dos seus mantementos e levar un rexistro da manutención.

A partir do ano 1991 incorporouse OBD-I (On Board Diagnostics) en tódolos vehículos, sentando as bases dos primeiros sistemas de monitorización para algúns compoñentes dos vehículos coma os controladores de emisións. Non obstante, OBD-I non era un estándar universal, e os fabricantes facían implementacións propias e con diferentes nivel de exactitude e veracidade a hora de mostrar datos.

Máis adiante, coa chegada de novas normas de emisións e o aumento dos sistemas informáticos nos vehículos, nace a segunda xeración do sistema: OBD-II. Este novo estándar, introducido no 1996 en Estados Unidos permite detectar fallos eléctricos, químicos e mecánicos que poidan afectar as emisións do vehículo.

Os vehículos gardan nas ECU un rexistro destes fallos, permitindo a un mecánico revisar os valores dos sensores e facendo máis doado o diagnóstico de un erro no funcionamento normal dos mesmos.

Tamén permiten, coas interfaces adecuadas, comunicarse co vehículo de modo que se poidan editar valores. Isto permite, dende a codificación de novas chaves, ata o desbloqueo de características ocultas no vehículo ou reservadas a gamas superiores. Tamén se poden facer edicións no sistema motor, editando a mezcla de aire/combustible e personalizando o funcionamento dos inxectores. Claro que, por motivos de seguridade e intereses propios das marcas, necesítanse protocolos personalizados para a comunicación con certas centralitas, especialmente para a súa edición

Pero, a partir do ano 2000 para vehículos gasolina e 2003 para vehículos diésel, oblígase a introducción de portos OBD-II nos vehículos comercializados en Europa baixo o estándar EOBD (European On-Board Diagnostics) que será a versión europea do estándar OBD-II. Este estándar permitirá, con un sistema de diagnosis, comunicarse con calquera vehículo para leer erros e realizar diversas probas dos sistemas do vehículo.

2. Obxectivos

Aínda que se pode profundizar moito nos diferentes protocolos OBD e os estándares compatibles, o obxectivo da aplicación é dar acceso ao usuario a datos básicos de funcionamento do vehículo tales como a medición de temperatura ambiente, temperaturas de entrada do aire, temperatura do refrixerante e do aceite, porcentaxe de aceleración e frenada, velocidade do vehículo e revolucións por minuto, entre outras.

Coma os protocolos empregados en cada vehículo varían, a aplicación centraráse non tanto en poder mostrar moitos datos, senón nos máis cruciais.

Además, outro dos pilares da aplicación será proveer dunha interface na que apuntar os mantementos realizados no vehículo, así como crear recordatorios para futuros mantementos. Por exemplo, recordar o cambio do aceite unha vez o ano, ou a revisión dos frenos cada 6 meses.

3. Situación previa

No mercado xa existen máquinas de diagnose que leen portos OBD, pero a maioría son equipos profesionais de alto custo. Para un usuario normal, que só queira saber algo máis do estado do seu vehículo, podería bastar cunha interface máis sinxela. Por iso, óptase por empregar un sistema baseado no chip ELM327, un procesador de baixo custo e open source que nos permitirá comunicarnos co vehículo mantendo o presuposto axustado.

Tamén temos aplicacións de notas ou recordatorios que nos poderían permitir tomar conta dos mantementos, pero coa nosa aplicación buscamos centralizar todo nunha sola app

4. Tecnoloxías empregadas

O desenvolvemento da aplicación, xa que se centra nos dispositivos móbiles iOS, estará baseada no framework de SwiftUI desenvolto por Apple, presentado no seu WWDC de 2019. Trátase dunha alternativa máis sinxela a anterior ferramenta UIKit, introducindo melloras coma a sintaxe declarativa. Tamén dan facilidades para a adaptación das aplicacións a diferentes interfaces e dispositivos: iOS, macOS, watchOS e tvOS.

Para a conexión Bluetooth, empregaráse a librería *SwiftOBD2*, que emprega o framework de apple CoreBluetooth que permite a conexión con dispositivos BLE (Bluetooth Low Energy).

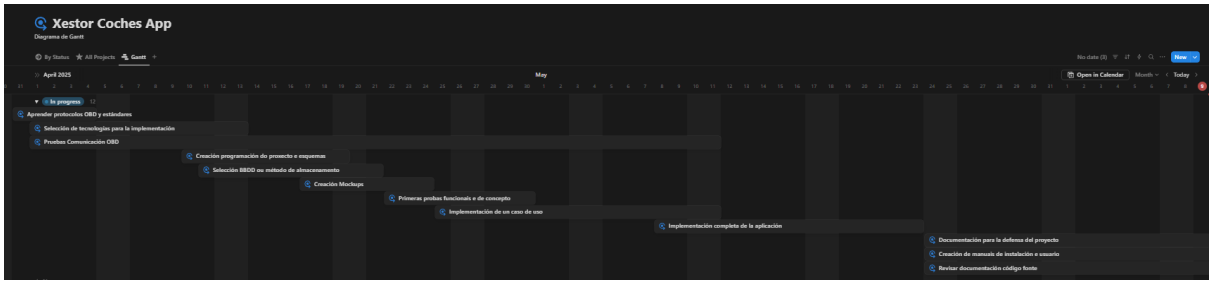
Para o almacenamento de datos empregaranse arquivos Json.

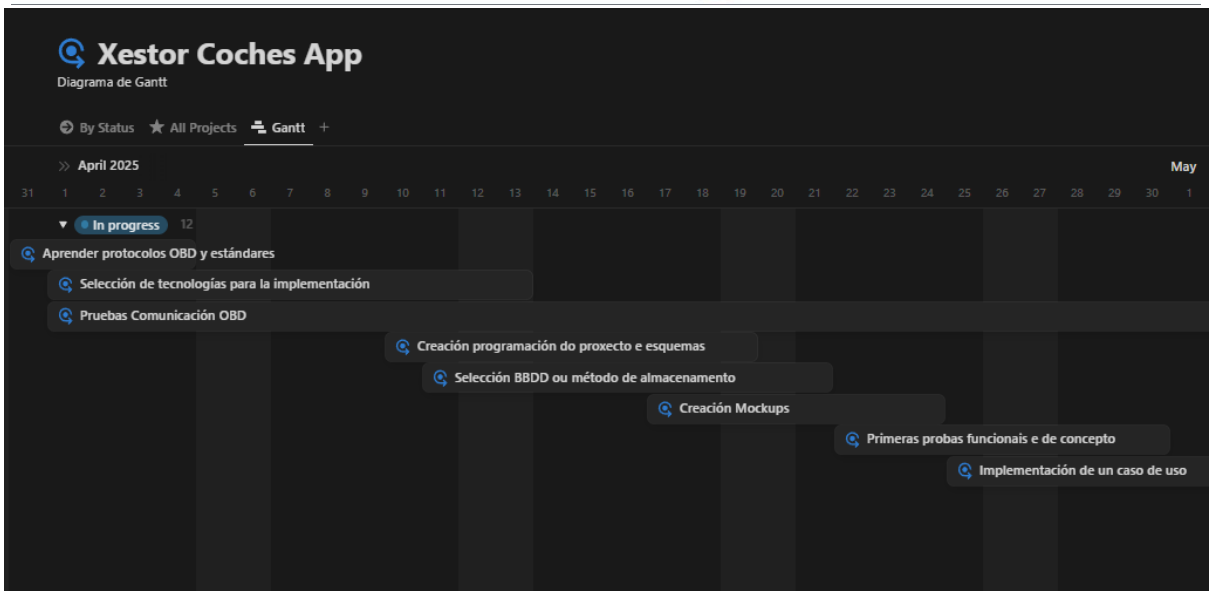
5. Solución proposta

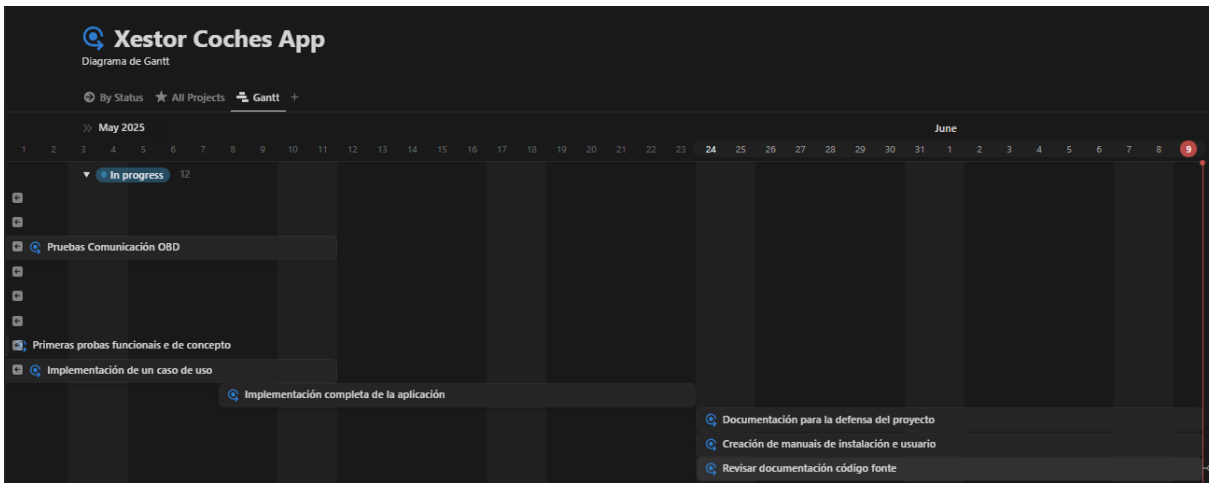
A proposta, en poucas palabas, trátase dunha aplicación sinxela na que revisar diferentes parámetros do coche e, además, levar conta dos mantementos do mesmo. Ao basearse nun sistema open source asequible, trátase dunha alternativa de baixo custo para o usuario e que pode ser moi útil, centralizando todo o relacionado co vehículo nunha mesma aplicación

6. Planificación do proxecto

Planificación do proxecto cun diagrama de Gantt







Descripción do modelo de desenvolvemento de software a implementar.

- O proxecto seguiu un método de desenvolvemento incremental. Este modelo permitiu ir engadindo paulatinamente diferentes funcionalidades de forma progresiva, realizando melloras en cada iteración

Análise do proxecto

Diagrama de casos de usos.

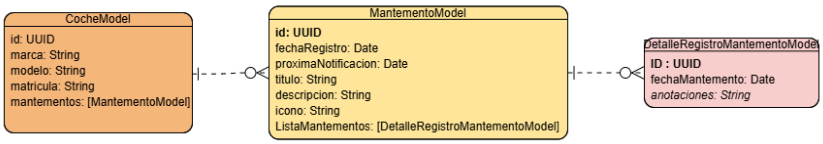
Aquí irá

Universo de discurso da base de datos.

O universo de discurso da base de datos da aplicación está composto polos seguintes elementos principais:

- Garaxe: almacena un conxunto de vehículos (coches) rexistrados polo usuario na aplicación
- Coche: contén información básica do vehículo (marca, modelo, combustible...) e garda os seus mantementos
- Mantemento: representa un tipo de operación realizada no vehículo (un cambio de aceite, revisión dos freos...)
- Detalle do mantemento: representa unha execución concreta do mantemento ao que se relaciona, podendo gardar información adicional da aplicación coma a kilometraxe a que se fixo ou calquera apunte que o usuario vexa necesario.

Diagrama de Entidad-Relación.



Deseño do proxecto:

Modelo relacional da base de datos.

Diagrama de clases.

Diagramas de fluxo de cada caso de uso.

Mockups da interface.

Presuposto completo (Hardware software e recursos humans)

Nome	Tipo	Prezo
Macbook Pro i7	Hardware	594,99€
iPhone XR	Hardware	174,00€
Lector OBD ELM327	Hardware	8,99€
Framework SwiftUI e XCode	Software	0€
Developer SwiftUI	Recursos humans	500€

7. Desenvolvemento e execución

Diagrama de despregamento e descripción xeral do funcionamento da aplicación.

Completar cos detalles técnicos que fose preciso resolver documentados ao longo do desenvolvemento.

8. Conclusións e reflexións

Valoración global do proxecto, resume do aprendido e das dificultades atopadas.

9. Bibliografía e Webgrafía

Formato APA

Wikipedia. (s.f.). *OBD*. Wikipedia, La enciclopedia libre. Recuperado el 9 de junio de 2025, de https://es.wikipedia.org/wiki/OBD

Elm Electronics. (2009). *ELM327DS: ELM327 – OBD to RS232 Interpreter* (versión 1.4b) [PDF]. SparkFun. https://cdn.sparkfun.com/assets/learn_tutorials/8/3/ELM327DS.pdf

Elm Electronics. (2005). *ELM327 AT Commands Summary Sheet* [PDF]. SparkFun. https://cdn.sparkfun.com/assets/4/e/5/0/2/ELM327_AT_Commands.pdf

Apple Inc. (s.f.). *Apple Developer*. https://developer.apple.com/

10. Anexo I – Manual técnico de instalación ou posta en marcha

Encóntrase no apartado docs de Git

11. Anexo II – Documentación de uso (manuais usuario)

Encóntrase no apartado docs de Git

12. Anexo III – Outra documentación

Calquera material adicional relevante (documentación da aplicación, manual de uso, capturas de pantalla, fragmentos de código, etc.).