

UNIVERSIDAD DE ALMERIA

ESCUELA SUPERIOR DE INGENIERÍA

**“DolphinOS, distribución
basada en Arch Linux
enfocada en desplegar
una Wii instalable en
sistemas x86_64”**

Curso 2022/2023

Alumno/a:

Diego Jesús Berenguel García

Director/es:

José Antonio Álvarez Bermejo

Agradecimientos

Querría agradecer en este apartado a todas las personas que me han estado apoyando tanto en el desarrollo de este proyecto, como a lo largo de toda la carrera.

Quisiera agradecer a toda mi familia, a mis padres, a mi hermana, a mis abuelos y a mi tía el apoyo incondicional que me han dado. Han sido comprensivos desde el primer momento que entré a la carrera y cuando he tenido momentos en los que me veía superado (que no han sido pocos), siempre han estado ahí para lo que fuese.

Sin su apoyo y cariño quizás no hubiera sido posible llegar a ser quien soy ahora mismo.

También agradecer a todos mis amigos y amigas, tanto de la carrera como de Discord, y antigu@s amig@s del instituto. En especial, también quisiera agradecer a mis amigos Alberto y Martín por todo el apoyo que me han estado dando (y por soportarme cada vez que la liaba) este año y durante toda la carrera, y a mi amiga Vicky, que aunque este año no haya estado aquí en España, desde que nos conocimos en primero de carrera, siempre ha estado apoyándome para lo que fuese y me ha hecho mejorar como persona.

Finalmente, me encantaría expresar mi agradecimiento a mi director de este Trabajo Fin de Grado, Jose Antonio Bermejo, por las herramientas y ayuda que me ha estado proporcionando a lo largo del desarrollo de este proyecto.

Índice

Agradecimientos.....	1
Índice de figuras.....	4
Resumen.....	5
Abstract.....	6
1 - Introducción.....	7
1.1 - Motivación del proyecto.....	7
1.2 - Objetivos del proyecto.....	8
1.3 - Planificación del proyecto.....	9
1.4 - Estado del arte.....	11
1.4.1 - Distribuciones Linux enfocadas en emulación de consolas.....	11
Batocera.linux.....	11
Lakka.....	12
Retropie.....	12
SteamOS.....	13
1.4.2 - Emuladores que permiten cargar el menú del sistema.....	14
RPCS3.....	14
JPCSP.....	15
Dolphin.....	16
2 - Tecnologías, herramientas y servicios utilizados.....	17
2.1 - Tecnologías.....	17
2.1.1 - Scripts de Bash.....	17
2.1.2 - Python y Python Flask.....	17
2.1.3 - Tecnologías web: HTML, CSS y JavaScript.....	18
2.2 - Herramientas.....	19
2.2.1 - Visual Studio Code.....	19
2.2.2 - Git.....	19
2.2.3 - VMWare Workstation.....	20
2.3 - Servicios.....	21
2.3.1 - GitHub.....	21
2.3.2 - SourceForge.....	22
3 - Desarrollo del proyecto.....	23
3.1 - Fase 1: Investigación.....	23
3.1.1 - Universal Blue (uBlue).....	23
3.1.2 - Arch Linux.....	24
3.1.3 - Software del sistema.....	25
3.2 - Fase 2: Infraestructura de desarrollo en GitHub.....	28
3.2.1 - Organización.....	28
3.2.2 - Repositorios.....	29
3.2.3 - GitHub Pages.....	31
3.2.4 - GitHub Actions.....	32
3.3 - Fase 3: Creación de la imagen ISO de instalación.....	34
3.3.1 - Archiso.....	34

3.3.2 - Calamares.....	36
3.4 - Fase 4: Configuración del sistema DolphinOS.....	40
3.4.1 - Configuración de DolphinOS.....	40
3.4.2 - Configuración del emulador Dolphin.....	42
3.5 - Fase 5: Desarrollo de la aplicación DolphinOS WebApp.....	44
4 - Conclusiones y trabajo futuro.....	48
4.1 - Conclusiones.....	48
4.2 - Trabajo futuro.....	48
5 - Enlaces de interés.....	50
6 - Guía de instalación y uso.....	51
Bibliografía.....	58

Índice de figuras

Figura 1: Cronograma Gantt de planificación del proyecto	10
Figura 2: Batocera.linux en Steam Deck	11
Figura 3: Lakka en una videoconsola Nintendo Switch	12
Figura 4: Máquina arcade a través de una Raspberry Pi usando Retropie	12
Figura 5: Emulación de Sega Dreamcast en SteamOS	13
Figura 6: XMB ejecutándose sobre RPCS3	14
Figura 7: Firmware del sistema ejecutándose en modo LLE en JPCSP	15
Figura 8: Menú del Sistema ejecutándose a través del emulador Dolphin	16
Figura 9: Navegador web Canal Internet (Opera 9)	18
Figura 10: Software de virtualización VMWare Workstation Pro	20
Figura 11: Portal del proyecto DolphinOS en SourceForge	22
Figura 12: Corrupción gráfica apreciable en aplicaciones XWayland en NVIDIA	26
Figura 13: Framework de instalación Calamares en DolphinOS	27
Figura 14: Organización DolphinOS-Development en GitHub	28
Figura 15: Repositorios del proyecto DolphinOS	30
Figura 16: Despliegue del repositorio de la distribución usando GitHub Pages	31
Figura 17: GitHub Action encargada de construir los paquetes de la distribución	33
Figura 18: GitHub Action encargada de construir la imagen ISO de instalación	33
Figura 19: Esquema de directorios del perfil Archiso de DolphinOS	34
Figura 20: Script de definición del perfil de Archiso	35
Figura 21: Gestor de arranque systemd-boot del medio de instalación de DolphinOS	36
Figura 22: Distintas instancias del módulo “shellprocess” de Calamares	37
Figura 23: Interfaz de bienvenida de Calamares	38
Figura 24: Interfaz de bienvenida de Calamares en la versión DolphinOS NVIDIA	39
Figura 25: Proceso de instalación de DolphinOS	39
Figura 26: Superposición MangoHud	41
Figura 27: Canal Homebrew Channel	43
Figura 28: Esquema de ficheros de DolphinOS WebApp	44
Figura 29: Mapa del sitio web DolphinOS WebApp	45
Figura 30: Aplicación web DolphinOS WebApp	45
Figura 31: Interfaz “Game Loader” de DolphinOS WebApp	46
Figura 32: Interfaz de conexión a red WiFi	48
Figura 33: Informe de conexión exitosa	48
Figura 34: Portal de descarga de DolphinOS	51
Figura 35: Interfaz de bienvenida de Calamares	52
Figura 36: Selección de región y huso horario	52
Figura 37: Selección de la distribución del teclado	53
Figura 38: Particionado de disco con Calamares	53
Figura 39: Interfaz de resumen previo a la instalación de DolphinOS	54
Figura 40: Instalación de DolphinOS	54
Figura 41: Configuración inicial de Wii	55
Figura 42: Menú del sistema Wii	55
Figura 43: Canal Internet	56
Figura 44: Marcador en favoritos de DolphinOS WebApp	56
Figura 45: Reinicio del sistema desde DolphinOS WebApp	57
Figura 46: Videojuego Planet 51 con la superposición MangoHud	57

Resumen

En el apasionante mundo de los videojuegos, las videoconsolas juegan un papel crucial ya que proporcionan un medio dedicado y medianamente asequible para poder disfrutar de los videojuegos. Como en la mayoría de ámbitos tecnológicos, todo está en continuo cambio, por lo que con el paso del tiempo las consolas se van quedando obsoletas tanto por obsolescencia programada o por meramente llegar a su vida máxima.

Sin embargo, a través de la emulación, es posible averiguar o simular cómo funcionan internamente estas consolas, y poder hacer compatible su software, específicamente videojuegos, en otros dispositivos hardware, con incluso, arquitecturas diferentes. Ya sea en un ordenador, un móvil, tablet, o incluso consolas más actuales, se podrá seguir disfrutando de los videojuegos de esas consolas más antiguas.

No obstante, existe un aspecto que la mayoría de emuladores actuales no consiguen lograr completamente, ya sea por ser muy complejo o por no hallarse en los planes de desarrollo de los mismos. Es el caso del firmware del sistema o sistema operativo, y algunas de las aplicaciones del sistema.

Esto suele ser un elemento de alta complejidad a la hora de ser emulado porque se necesita una emulación más a bajo nivel, llegando a emular partes del hardware de la consola, para que el firmware pueda ejecutarse como si fuese hardware nativo.

Por suerte, uno de los emuladores que permiten cargar el firmware del sistema de forma sencilla es Dolphin, un emulador de Wii, GameCube y Gameboy Advance.

A través de Dolphin se puede cargar la aplicación “Menú del Sistema”, la cual proporciona el propio menú principal del sistema Wii junto con todas las aplicaciones preinstaladas. De esta forma, se le proporciona al usuario la misma experiencia que ofrecería la videoconsola, pero en cualquier hardware en el cual se pueda ejecutar este emulador.

Por ello, se ha desarrollado un sistema instalable, basado en una distribución GNU/Linux, el cual permite disfrutar de la misma experiencia que ofrecería una videoconsola Wii, en cualquier ordenador con arquitectura x86_64.

A través de un instalador gráfico, un usuario podrá instalar el sistema fácilmente en su ordenador, pudiendo instalarlo conjuntamente con su sistema operativo actual si así lo desea, manteniendo todos sus datos intactos. Tras realizar la instalación, el sistema está configurado automáticamente para arrancar exclusivamente el emulador Dolphin, cargando el diálogo de configuración inicial de Wii. En los sucesivos inicios, el sistema cargará automáticamente al menú de Wii.

Además, mediante una herramienta web que el usuario podrá acceder a través del navegador web, el usuario podrá gestionar el sistema Linux base, cargar juegos almacenados en una memoria USB o abrir la configuración del emulador.

Palabras clave: Emulación; Preservación del software; Dolphin; Wii; Linux; Arch Linux; Python Flask

Abstract

In the thrilling world of video games, video game consoles play an important role given that they provide a dedicated and affordable medium for allowing you to play video games.

As in almost every IT environment, everything is changing constantly, and this affects video game consoles as well, as they become deprecated either due to planned obsolescence or simply by reaching its maximum lifespan.

However, it is possible to find out or simulate how these video game consoles work internally through emulation, as well as making their software (specially video games) compatible with other devices that could even be running on different architectures. Through emulation it's possible to keep playing and enjoying older consoles games on a modern computer, phone, tablet or even on more recent consoles.

Nevertheless, there is an aspect that most current emulators don't manage to develop. This is the case of video games consoles' operating systems or system firmware.

The reasons may vary, but it is clear that this is a very complex thing to carry out, given that a lower level emulation is needed. Sometimes some hardware of the actual console needs to be emulated in order to run the system firmware.

Luckily, one of the few emulators that allows the system firmware to boot easily is Dolphin, a Wii, GameCube and GameBoy Advance emulator.

Through Dolphin it's possible to boot the "System Menu" application, that provides the actual Wii menu, including all the preinstalled applications. This way, the user is provided with the same experience that the video game console would provide, but on every hardware that this emulator could run on.

Because of this, this project provides an installable system, based on a GNU/Linux distribution, which allows an user to enjoy the same experience that the Wii video game console could offer, on every x86_64 architecture computer.

Through a graphical installer, an user will be able to easily install the system on their computer, being able to install it alongside their current operating system if desired, while keeping all their data intact. After system installation, the system is automatically configured to boot directly to Dolphin emulator, loading the Wii's initial configuration procedure.

In subsequent startups, the system will automatically load the Wii menu.

This project provides, as well, a web application accessible through the Wii's web browser, that allows the user to manage the base Linux system, load games stored on a USB stick or even enter the emulator's configuration interface.

Keywords: Emulation; Software Preservation; Dolphin; Wii; Linux; Arch Linux; Python Flask

1 - Introducción

Este Trabajo Fin de Grado se enfoca en la creación de un sistema operativo instalable, basado en una distribución Linux, el cual servirá de base para el emulador de la videoconsola Nintendo Wii.

De esta forma, una vez instalado el sistema, le brindará al usuario la misma experiencia que tendría con una auténtica Wii, pero sin necesidad de disponer de dicha videoconsola.

El proyecto consta de varias partes. Por un lado, se ofrece una imagen de disco ISO, la cual se puede grabar en un DVD o en una memoria USB.

Esta imagen de disco incluye un sistema “live” con el programa de instalación Calamares. Al arrancar desde la unidad de DVD o desde la memoria USB, automáticamente se cargará el instalador Calamares, a través del cual el usuario podrá instalar el sistema DolphinOS en su ordenador.

Por otro lado, tras instalarse, el sistema viene preconfigurado y optimizado para arrancar automáticamente en el menú de la videoconsola Wii mediante el emulador Dolphin.

Por último, se ofrece una interfaz web de configuración, accesible a través del navegador de internet de Wii. Desde esta interfaz el usuario podrá realizar varias acciones de configuración del sistema Arch Linux base, abrir la configuración del emulador Dolphin, o cargar juegos, almacenados en una unidad USB externa, en el sistema Wii emulado.

1.1 - Motivación del proyecto

Todos los equipos electrónicos disponen de una vida útil máxima, en ocasiones llegando a acortarse debido a las prácticas de obsolescencia programada realizadas por algunos fabricantes. En ciertos tipos de dispositivos la obsolescencia del hardware también puede conllevar a la desaparición del software que se ejecuta en ellos. Esto es bastante común en dispositivos dedicados, en los cuales se diseña el hardware y el software para una tarea en concreto.

Este es el caso de las videoconsolas. En ellas su hardware y software se centra en proveer una plataforma en la cual se ejecuten videojuegos. El software normalmente se diseña para poder ejecutarse exclusivamente sobre ese hardware, sacando el máximo partido del hardware.

Por ello, se plantea una solución que pretende mantener la experiencia que ofrece una videoconsola Wii, pero sin depender de su hardware propio. Gracias al emulador Dolphin, es posible ejecutar el software de Wii, en un hardware totalmente distinto. Actualmente Dolphin soporta oficialmente las arquitecturas x86_64 y ARM64.

En cuanto a sistemas operativos soporta todos los sistemas operativos principales como Windows, Linux, macOS y Android.

En este caso, el sistema operativo elegido ha sido Arch Linux debido a su gran capacidad de adaptarlo hacia una tarea específica, manteniendo un bajo impacto en el rendimiento del hardware comparado con otros sistemas operativos.

Además, en proyectos de preservación del software de este tipo es una buena elección el uso de sistemas operativos de código libre, puesto que gracias a la comunidad se puede realizar una colaboración conjunta y mantener el software a lo largo del tiempo.

1.2 - Objetivos del proyecto

Este proyecto tiene como objetivo principal la construcción de una distribución Linux customizada, la cual pueda simular la experiencia de usuario de una videoconsola Nintendo Wii proporcionando una base para el emulador Dolphin.

Dentro de este proyecto se han definido tres objetivos principales:

- Creación de un medio de instalación para DolphinOS. A través de una imagen ISO customizada de Arch Linux se carga el software necesario para la instalación de DolphinOS.
Gracias al framework de instalación Calamares, se le presenta al usuario una interfaz gráfica en la que sencillamente en un par de pasos puede instalar el sistema DolphinOS en su equipo.
- Diseño y configuración de un sistema basado en Arch Linux enfocado a proveer una base para el emulador Dolphin. A partir de una instalación base de Arch Linux, se instalan todos los paquetes necesarios y se definen las configuraciones esenciales para que el sistema pueda funcionar correctamente. Toda la configuración se distribuye en paquetes compatibles con Arch Linux, de forma que, si se introduce una nueva característica o se realiza alguna corrección de errores, los usuarios puedan recibirlas a través de una actualización del sistema, sin necesidad de reinstalar DolphinOS.
- Creación de una interfaz web utilizando Python Flask para proporcionar al usuario un medio gráfico en el cual pueda seleccionar los juegos a cargar en el emulador, cambiar la configuración del emulador o administrar el sistema.
La idea principal es que el usuario no tenga que salir del emulador Dolphin, que todas las acciones se realicen dentro de él para poder ofrecer una experiencia similar a la que ofrecería una Wii auténtica. Puesto que Dolphin es solamente un emulador, se necesita una interfaz que sirva de comunicación entre el sistema emulado y el sistema anfitrión.

1.3 - Planificación del proyecto

El planteamiento del proyecto comenzó la semana del 24 de abril de 2023, pocos días después de la entrega del anteproyecto.

La planificación del desarrollo del proyecto se ha llevado a cabo en 6 fases:

- **Fase 1:**

Durante esta primera fase se estuvo realizando una investigación exhaustiva sobre cómo se podría llevar a cabo este proyecto, tanto de las distribuciones Linux candidatas como de las tecnologías, software necesario e infraestructura de desarrollo del proyecto.

Esta investigación tuvo una duración de aproximadamente unas 40 horas.

- **Fase 2:**

Tras realizar la fase de investigación, comenzó la fase de definición de la infraestructura de desarrollo. Se decidió apostar por desarrollo basado en repositorios de control de versiones de código fuente Git, en la plataforma GitHub. En esta fase se fueron configurando los repositorios necesarios para cada una de las partes del proyecto, y el sistema de integración continua mediante GitHub Actions, para construir los paquetes de la distribución Linux, albergar el repositorio de la distribución y el sistema de construcción de la imagen ISO de instalación. Tuvo una duración estimada de unas 20 horas.

- **Fase 3:**

A partir de la tercera fase, comienza el desarrollo como tal de proyecto. Durante esta fase se configuró el medio de instalación de DolphinOS.

Aquí se incluye la configuración del sistema live usando una versión customizada de las imágenes de instalación de Arch Linux (ArchISO), y la configuración del framework de instalación de distribuciones Linux, Calamares. Esta fase tuvo una duración de unas 80 horas.

- **Fase 4:**

En el transcurso de esta fase se llevó a cabo la configuración del sistema DolphinOS. Todas las configuraciones han sido divididas en diferentes repositorios, a partir de los cuales se construyen los paquetes instalables en el sistema. Para la construcción de cada paquete se definieron una serie de scripts de creación de paquetes de Arch Linux (PKGBUILD).

Es la fase en la que más tiempo fue invertido, alrededor de unas 110 horas.

- **Fase 5:**

Última fase de desarrollo del proyecto DolphinOS.

En esta fase se desarrolló la aplicación web para poder interactuar con el sistema anfitrión desde el sistema emulado.

Su desarrollo tuvo una duración de aproximadamente 40 horas.

- **Fase 6:**

En esta fase se halla la documentación de la memoria y la presentación para defensa del trabajo. Aproximadamente ha tenido una duración de unas 30 horas.

Este Trabajo Fin de Grado ha tenido una duración total de 320 horas de trabajo totalmente autónomo. A continuación se muestra un diagrama de Gantt con la planificación del proyecto:



Figura 1: Cronograma Gantt de planificación del proyecto

1.4 - Estado del arte

Actualmente DolphinOS es el único sistema enfocado en proveer una base para el emulador Dolphin, pudiendo disfrutar de una experiencia muy cercana a la que ofrecería una videoconsola Wii auténtica.

Sin embargo, existen varios proyectos de distribuciones Linux enfocados en ofrecer un sistema preconfigurado para albergar múltiples emuladores de videoconsolas.

Por otro lado, Dolphin no es el único emulador que permite cargar el firmware del sistema de la videoconsola emulada, simulando su experiencia de usuario.

1.4.1 - Distribuciones Linux enfocadas en emulación de consolas

La versatilidad que ofrecen los sistemas operativos GNU/Linux hace que estos sistemas sean una muy buena elección para el desarrollo de tareas específicas.

Una de estas tareas específicas puede ser el caso de la creación de una máquina dedicada a la emulación de videoconsolas, como la creación de un sistema arcade o un sistema en el cual se puedan disfrutar de los juegos de múltiples consolas antiguas.

A continuación se listan una serie de distribuciones enfocadas en la emulación de videoconsolas:

Batocera.linux

Batocera [3] es una sistema basado en la distribución Buildroot Linux [53].

Actualmente soporta más de 70 sistemas mediante diversos emuladores, incluyendo desde máquinas arcade, consolas de 8 bits, videoconsolas portátiles y sobremesa hasta octava generación, ordenadores retro como las máquinas Apple II, Commodore o Atari. Adicionalmente, incluye varios portes de motores de juegos de código abierto como prboom para el Doom original o xash3d_fwgs para Half Life.

Cuenta con la interfaz gráfica EmulationStation para cargar los juegos y gestionar los emuladores o el sistema.

Batocera soporta las arquitecturas x86, x86_64 de ordenador, y múltiples sistemas ARM incluyendo videoconsolas portátiles enfocadas en emulación y placas de desarrollo como las Raspberry Pi o las Odroid.



Figura 2: Batocera.linux en Steam Deck

Lakka

Lakka [4] es una distribución enfocada a la emulación de videoconsolas, basada en el sistema LibreELEC, reemplazando el sistema de entretenimiento multimedia Kodi por RetroArch, una suite de emulación de consolas que incluye un gran número de emuladores, llamados “cores” en RetroArch.

Al contrario que Batocera, Lakka utiliza la propia interfaz gráfica de RetroArch para gestionar juegos, emuladores y el sistema. Lakka actualmente soporta ordenadores x86_64 y sistemas ARM como Odroid, Raspberry Pi, o incluso videoconsolas portátiles como Nintendo Switch.



Figura 3: Lakka en una videoconsola Nintendo Switch [56]

Retropie

Retropie [5] es un sistema enfocado en la emulación de consolas a través de la suite de emulación RetroArch, usando EmulationStation como interfaz gráfica principal.

Al igual que en Lakka, Retropie soporta casi la mayoría de sistemas de emulación de RetroArch. Retropie está basado en Raspbian [6] (actualmente conocido como Raspberry Pi OS), una versión de Debian exclusiva para las placas Raspberry Pi.

Por este motivo, Retropie sólamente soporta la arquitectura ARM, en particular las placas Raspberry Pi y similares. Sin embargo, Retropie ofrece también una forma de poder ser instalado sobre un sistema Debian en un ordenador x86_64.



Figura 4: Máquina arcade a través de una Raspberry Pi usando Retropie [55]

SteamOS

SteamOS [7] es una distribución Linux desarrollada principalmente por Valve con el objetivo de ser el sistema operativo enfocado en los videojuegos tanto para PC como para las propias videoconsolas de Valve. Su interfaz gráfica es el nuevo modo Big Picture de Steam. Inicialmente estaba basada en Debian hasta su versión 2, aunque posteriormente, con la llegada de la videoconsola portátil Steam Deck en febrero de 2022, Valve presenta SteamOS 3.0, usando como base el sistema Arch Linux.

Uno de los puntos a favor de SteamOS es que además de los juegos y software que se distribuyen en la propia tienda de Steam, también soporta Flatpak como su gestor de paquetes oficial. Flatpak [8] es un gestor de paquetes multi distribución basado en contenedores. Gracias a Flatpak se pueden instalar aplicaciones en el sistema inmutable que utiliza SteamOS, al instalarse las aplicaciones en el entorno del usuario, sin interferir con los programas y dependencias del sistema.

Pese a que SteamOS no esté principalmente enfocado a emulación, la gran variedad de emuladores disponibles a través de Flatpak y la experiencia de usuario enfocada a videoconsolas, hacen que SteamOS sea una distribución excelente para la emulación de videoconsolas, sobre todo en el Steam Deck.



Figura 5: Emulación de Sega Dreamcast en SteamOS [57]

1.4.2 - Emuladores que permiten cargar el menú del sistema

El mayor componente para poder preservar la experiencia de usuario de una videoconsola es a través de la emulación de su menú del sistema. La gran mayoría de emuladores necesitan del sistema operativo de la consola (también conocido como firmware del sistema).

Sin embargo, no todos los emuladores de consolas de quinta generación en adelante soportan cargar el menú del sistema de la videoconsola (la mayoría de consolas de cuarta generación y anteriores no disponían de ningún menú del sistema, cargaban el juego desde el disco o cartucho directamente al arranque). Esto es debido a que hacer que se emule correctamente el menú del sistema, junto con la mayoría de componentes del firmware es una tarea bastante compleja que se suma al objetivo principal de los emuladores, que es poder jugar juegos de dicha videoconsola en un hardware diferente. Algunos emuladores establecen esta funcionalidad como un objetivo secundario, a lograr a medio/largo plazo, y otros no consideran que sea una característica esencial y no está en sus planes la implementación de dicha característica. A continuación se exponen algunos de los emuladores que soportan la emulación del menú del sistema.

RPCS3

RPCS3 [9] es un emulador de código libre (GPLv2) escrito en C++ de la videoconsola Sony PlayStation 3. Su desarrollo comenzó en 2011, y actualmente es el único emulador de PS3 que sigue en desarrollo, siendo a su vez bastante funcional. En 2018 apareció la primera build funcional que permitía cargar el menú del sistema, conocido como XMB o VSH (internamente). Esta primera build tenía una funcionalidad bastante reducida, no podía cargar juegos ni aplicaciones desde el XMB, no tenía soporte de audio (faltaba implementar la emulación de la tarjeta de sonido de la tarjeta gráfica (RSXAudio)), no existía soporte de HDMI emulado y contaba con varios problemas gráficos.

Con el paso de los años, algunos problemas se han ido solucionando, como es el caso del audio, los problemas gráficos y el HDMI, pero a día de hoy todavía no se pueden ejecutar juegos ni aplicaciones desde el XMB. Esto es debido a que es necesario implementar el soporte multiproceso dentro del emulador, puesto que cuando se ejecuta un juego o una aplicación, estos se ejecutan en un proceso y los juegos en otro diferente.

Por este motivo, principalmente, no se seleccionó RPCS3 para el desarrollo de este proyecto.

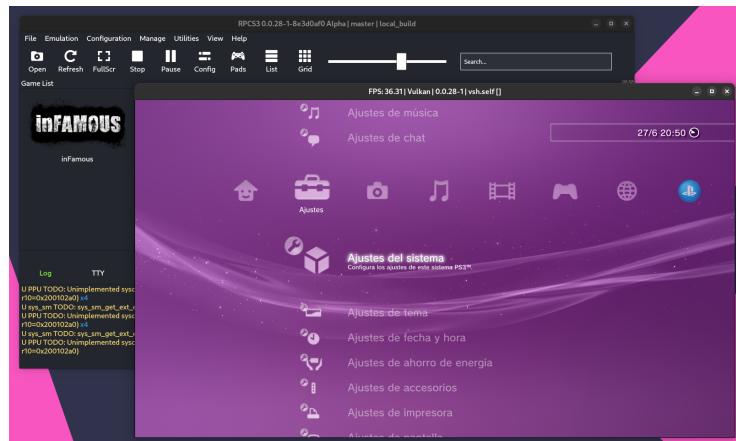


Figura 6: XMB ejecutándose sobre RPCS3

JPCSP

JPCSP [10] es un emulador de la videoconsola Sony PSP escrito en Java, que inicialmente se desarrolló como un depurador para investigación del funcionamiento interno de PSP, aunque posteriormente logró tomar tracción y consiguió poder ejecutar juegos comerciales. El proyecto comenzó en 2008, y actualmente sigue en desarrollo. Tanto JPCSP son los únicos emuladores funcionales de PSP que siguen en desarrollo.

Ambos proyectos se basan en la emulación de PSP, aunque cada uno tiene unos objetivos diferentes.

Por un lado, PPSSPP se centra sobre todo en la parte de la emulación de juegos, tomando un enfoque de emulación de más alto nivel.

Por otro lado, JPCSP toma un enfoque de emulación más a bajo nivel, centrándose más en la emulación del sistema, y no tanto en la emulación de juegos.

A través de JPCSP es posible tanto cargar sólamente el menú del sistema (XMB) en modo de emulación de alto nivel (HLE), como emular el comportamiento real de una PSP mediante la emulación de bajo nivel (LLE). Sin embargo, cada enfoque tiene sus ventajas y desventajas. La emulación a alto nivel ofrece un rendimiento superior, puesto que sólamente necesita emular lo esencial, y para poder cargar el XMB es necesario obtener los componentes del XMB desencriptados, lo cual sólamente es posible realizarlo desde una PSP o PS Vita real, ambas con el firmware modificado.

Por otro lado, la emulación a alto nivel permite emular el comportamiento real de una PSP con su secuencia de arranque, por lo que únicamente es necesario proporcionar el fichero de actualización del sistema, el cual incluye el firmware del sistema. Por contrapunto, el rendimiento se ve gravemente afectado, y en determinadas partes, pueden ocurrir algunos bugs. Debido a estos problemas, a la falta de soporte en Java para el sistema de ventanas Wayland, y a ciertos errores que se producían al establecer el emulador en pantalla completa en sistemas Linux, no se seleccionó este emulador para este proyecto.

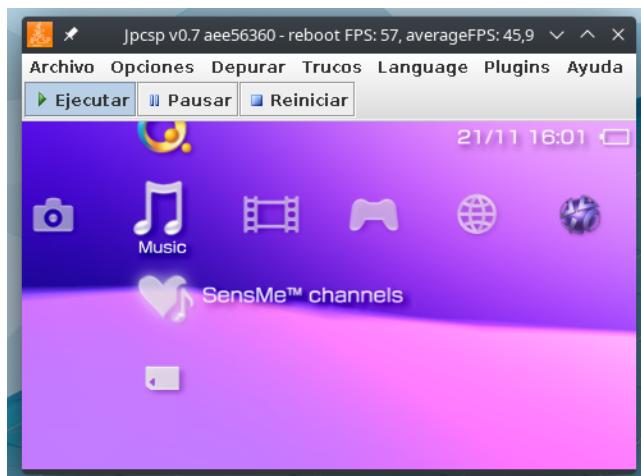


Figura 7: Firmware del sistema ejecutándose en modo LLE en JPCSP

Dolphin

Dolphin [13] es un emulador de las videoconsolas Nintendo GameCube y Nintendo Wii. Su desarrollo comenzó en 2003 como un emulador propietario de GameCube. En mayo de 2007 se publicó como un proyecto de código libre (GPLv2), y con soporte inicial de emulación de Wii. Actualmente continúa en desarrollo y es único el emulador totalmente funcional de Wii. Dolphin oficialmente soporta múltiples plataformas, incluyendo sistemas Windows, Linux, macOS y FreeBSD en arquitectura x86_64, y Android y macOS en arquitectura ARM64.

Dolphin soporta múltiples tipos de mandos, incluyendo nativamente mandos de GameCube y mandos de Wii. En el caso de los mandos de Wii, soporta dos métodos de conexión de los mandos. Por un lado el método “Bluetooth Passthrough”, en el cual el adaptador Bluetooth pasa a ser controlado directamente por el sistema de Wii emulado. Como inconveniente, el sistema operativo deja de tener acceso al adaptador bluetooth, lo cual hace que otros dispositivos Bluetooth conectados, como auriculares o altavoces dejen de funcionar con el sistema base. Este método es recomendable sólo en ciertos sistemas, en los cuales la pila Bluetooth del sistema no funcione correctamente. Por otro lado, Dolphin hace uso de un controlador emulado, el cual permite emparejar y conectar mandos de Wii oficiales sin necesidad de ceder el control del adaptador Bluetooth al sistema Wii emulado. Este es el método más recomendado.

A partir de febrero de 2009, se consiguió emular correctamente el menú del sistema en su versión 1.0. A partir de ese momento, todas las versiones de Dolphin han sido capaces de emular correctamente prácticamente todas las versiones del firmware de Wii.

Además, recientemente los desarrolladores de Dolphin han estado colaborando con los desarrolladores de RiiConnect24, para hacer que sea compatible con Dolphin.

RiiConnect24 [14] una proyecto por parte de la comunidad que pretende reimplementar y volver a hacer funcional el servicio online de Wii, WiiConnect24, a través de los canales “Canal Noticias”, “Canal Tiempo” y “Canal Nintendo”, entre otros.

Por otro lado, Dolphin también es compatible con Wiimmfi [15], la reimplementación de la comunidad de los servicios online de ciertos juegos (Conexión WiFi de Nintendo), pudiendo hacer posible el juego online en determinados juegos, como por ejemplo, en Mario Kart.



Figura 8: Menú del Sistema ejecutándose a través del emulador Dolphin

2 - Tecnologías, herramientas y servicios utilizados

2.1 - Tecnologías

En este apartado, se enumeran los lenguajes de programación, scripting y de desarrollo web utilizados en el desarrollo del proyecto.

2.1.1 - Scripts de Bash

En sistemas UNIX y UNIX-like, los comandos introducidos a través de una terminal son interpretados por un software llamado intérprete de comandos. Existen diversos intérpretes de comandos, siendo Bash [16] (Bourne Again SHell) el más utilizado en sistemas UNIX y UNIX-like. Bash permite tanto ejecución de comandos de forma interactiva, como en lote mediante un guión de comandos. Esto es lo que se conoce como scripts.

Bash presenta un lenguaje de scripting, el cual permite automatizar tareas repetitivas o completas. Una de las características más poderosas de los scripts Bash es su capacidad para utilizar variables, estructuras de control de flujo (como bucles y condicionales), funciones y herramientas de redirección de entrada/salida. Esto permite a los desarrolladores escribir scripts flexibles y sofisticados que pueden interactuar con el sistema operativo, realizar cálculos, manipular archivos, leer y escribir en la entrada/salida estándar y mucho más.

En este caso, se han utilizado scripts de Bash de varias formas. Por un lado, de forma directa, para automatizar tareas tanto en el sistema DolphinOS como en los scripts de las GitHub Actions. Por otro lado, también se han usado de forma indirecta, a través de ciertos ficheros de configuración (ficheros de autoarranque del gestor de ventanas, fichero .bash_profile para la carga de programas al inicio de sesión), los cuales se interpretan usando Bash.

2.1.2 - Python y Python Flask

Python [54] es un lenguaje de programación y scripting de uso general. Python es un lenguaje interpretado, y no compilado, lo que hace que los módulos de Python sean portables allá donde exista un intérprete de Python. Soporta múltiples paradigmas de programación, incluyendo programación orientada a procedimientos, programación orientada a objetos y programación funcional. Uno de sus puntos fuertes es la sencilla legibilidad del código escrito, lo cual hace que sea una opción excelente para tareas que no requieran un lenguaje de programación específico y se deseen realizar más sencillamente.

Python cuenta además con un repositorio y gestor de paquetes, contando con un amplio catálogo de librerías y módulos, con casi medio millón de proyectos.

Por otro lado, Python Flask [17] es un framework web ligero y flexible que permite crear aplicaciones web rápidas y sencillas usando el lenguaje de programación Python.

Entre sus ventajas comparado con otros frameworks web, se destaca su facilidad de uso y curva de aprendizaje baja, y su enfoque de “micro-framework”, el cual se centra en proporcionar las funcionalidades básicas necesarias (aunque más avanzado que el propio módulo HTTP de Python) para construir aplicaciones web, como enruteamiento de URLs y manejo de solicitudes y respuestas HTTP.

Gracias a Python y al framework Flask, se ha podido desarrollar en este proyecto la aplicación web DolphinOS WebApp, la cual se encarga de proporcionar una interfaz al usuario para gestionar ciertas configuraciones del emulador y del sistema operativo Arch Linux anfitrión, todo dentro del propio emulador, a través del navegador web de Wii.

En DolphinOS WebApp, Flask proporciona un backend, un sistema de enrutamiento de URLs y gestión de peticiones HTML, mientras que la interacción con el sistema operativo se realiza a través de scripts de Python. Finalmente, gracias a las tecnologías web HTML4, CSS 2.1 y JavaScript, se muestra una interfaz gráfica al usuario a través del navegador.

2.1.3 - Tecnologías web: HTML, CSS y JavaScript

El lenguaje HTML [18] (HyperText Markup Language) es el lenguaje de marcado estándar para la elaboración de documentos de páginas web.

Este estándar es gestionado por la World Wide Web Consortium (W3C), una organización dedicada a la estandarización de casi todas las tecnologías dedicadas a la web.

HTML fue una pieza clave en el desarrollo y expansión de la World Wide Web (WWW), siendo el estándar que se impuso para la visualización de páginas web, y es el que, hasta la actualidad, han adoptado todos los navegadores.

Por otro lado, CSS [19] (Cascading Style Sheets) es un lenguaje de diseño gráfico para crear y definir la presentación de un documento estructurado escrito en un lenguaje de marcado. En el desarrollo web, se utiliza para establecer el diseño visual de los documentos web HTML.

En cuanto a la programación web como tal, se encuentra el lenguaje de programación JavaScript [20]. Es un lenguaje de programación interpretado y orientado a objetos.

Se utiliza principalmente en el lado del cliente en ámbito del desarrollo web, aunque en los últimos años, gracias a diversos frameworks basados en JavaScript, también se implementa para la lógica en el lado del servidor.

En este proyecto se utilizan las tecnologías web anteriormente mencionadas para el desarrollo del lado del cliente de la aplicación web DolphinOS WebApp.

Cualquier interacción que tiene el usuario con la aplicación web se realiza a través de la interfaz gráfica escrita en HTML4, CSS 2.1 y JavaScript.

El motivo por el cual se utilicen versiones obsoletas de HTML, CSS y JavaScript viene dado por la compatibilidad web que soporta el navegador de Wii, también conocido como “Canal Internet”, el cual está basado en una versión muy antigua de Opera (Opera 9), la cual data de 2007 [21].



Figura 9: Navegador web Canal Internet (Opera 9)

2.2 - Herramientas

Durante el desarrollo de este proyecto se han utilizado una serie de diversas herramientas, como pueden ser editores de código, sistemas de control de versiones de código, sistemas de virtualización, etc.

2.2.1 - Visual Studio Code

Visual Studio Code [22] es un editor de código fuente desarrollado por Microsoft, y de código abierto, publicado bajo la licencia MIT. Visual Studio Code es un software multiplataforma, soportando los sistemas operativos Windows, Linux y macOS. También cuenta con una versión online, puesto que es una aplicación web.

En Windows, Linux y macOS, esta aplicación web se integra con el sistema usando el framework Electron, el cual es un navegador web Chromium embebido, con el objetivo de crear aplicaciones de escritorio a partir de aplicaciones web.

En el desarrollo de este proyecto se ha utilizado una versión de Visual Studio Code mantenida por la comunidad denominada Code - OSS [55].

A través de Code - OSS se ha llevado a cabo el desarrollo la aplicación web DolphinOS WebApp, programado los scripts de Python y Bash, editado los ficheros de configuración del sistema DolphinOS y gestionado el software de gestión de versiones de código Git, a través de la interfaz gráfica que ofrece Visual Studio Code.

2.2.2 - Git

Git es un software y protocolo de control de versiones de software, desarrollado por el propio desarrollador del Kernel Linux, Linus Torvalds, y publicado como software libre bajo la licencia GNU GPLv2.

Git se encarga de llevar un registro de todos los cambios producidos en el código, que previamente han de haber sido marcados para su registro por el usuario.

Es una herramienta sumamente potente a la hora de realizar trabajo colaborativo entre varias personas o entidades. El concepto de las ramas permite que puedan haber varias vías de desarrollo simultáneamente sin interferir una con otra dentro de un mismo repositorio, y que en un momento una se pueda beneficiar de la otra o viceversa.

En este proyecto, Git se ha utilizado para llevar un registro de todos los cambios producidos en los distintos repositorios, y además, poder publicar el código y los repositorios en la red a través de la plataforma GitHub.

2.2.3 - VMWare Workstation

VMWare Workstation es un software de virtualización comercial para equipos de escritorio desarrollado por la empresa VMWare [24]. VMWare Workstation dispone de dos versiones con distintas licencias, por un lado la versión gratuita VMWare Workstation Player y por otro lado, la versión de pago VMWare Workstation Pro. La versión Player es una versión más simple, mientras que la versión Pro ofrece un número mayor de características, así como una opción de selección de compatibilidad con versiones anteriores de VMWare Workstation. Es compatible con los sistemas operativos Windows, Linux, FreeBSD y Solaris. Alternativamente a VMWare Workstation, existe la versión VMWare Fusion, que ofrece un sistema de virtualización a dispositivos Apple Mac, tanto en arquitecturas x86_64 (Intel) y ARM64 (Apple M Series).

En este proyecto VMWare Workstation ha servido de gran utilidad para virtualizar e ir realizando pruebas del sistema DolphinOS en máquinas virtuales. De esta forma, se puede testear sencillamente la configuración de DolphinOS sin necesidad de estar instalándolo constantemente en un ordenador físicamente.

El motivo principal por el cual se ha seleccionado VMWare Workstation como software de virtualización comparado con otros sistemas de virtualización como VirtualBox o QEMU/KVM, es debido a que VMWare proporciona un sistema de gráficos paravirtualizados, de forma que se puedan ejecutar aplicaciones 3D con un rendimiento aceptable sin necesidad de realizar passthrough a la tarjeta gráfica para hacer que se conecte al sistema virtualizado.

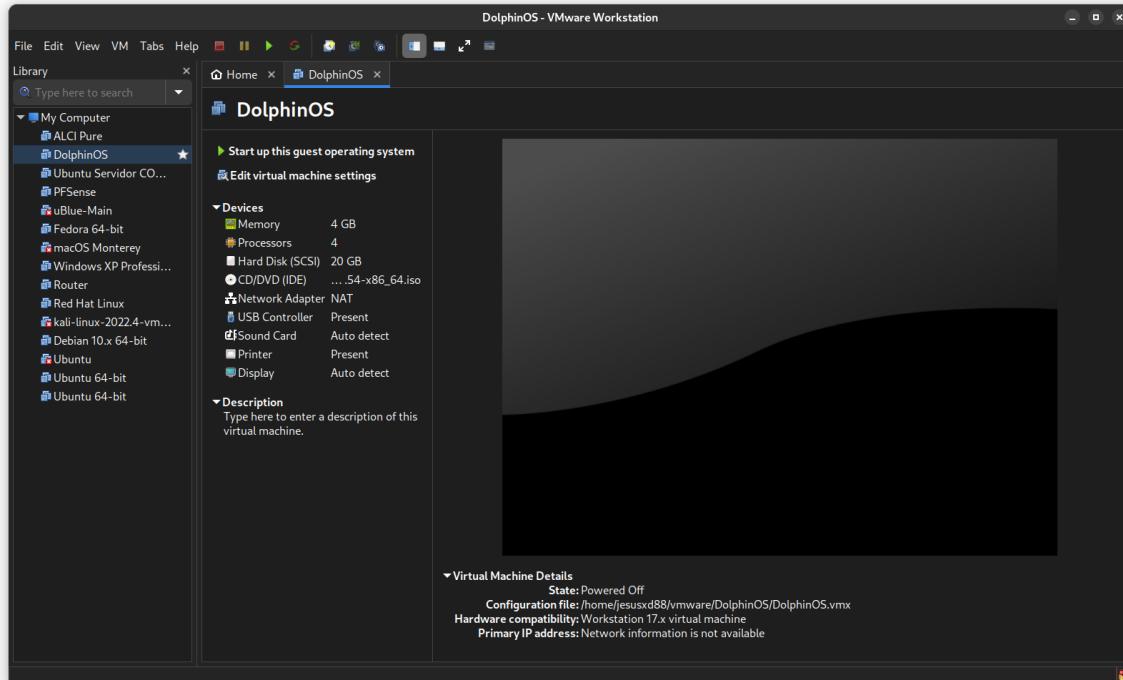


Figura 10: Software de virtualización VMWare Workstation Pro

2.3 - Servicios

Para el desarrollo de este proyecto, se han usado varios servicios online, tanto para publicación del código online, alojamiento del repositorio de la distribución y de las imágenes ISO, como para la construcción del sistema.

2.3.1 - GitHub

GitHub [25] es una plataforma de desarrollo colaborativo, enfocada originalmente para el alojamiento de proyectos utilizando el sistema de control de versiones Git.

Fue fundada en 2008 y hasta la fecha, es la plataforma de desarrollo colaborativo más grande y usada que existe.

Además del control de versiones, GitHub ofrece una amplia gama de características y herramientas que facilitan el desarrollo de software colaborativo, como pueden ser el seguimiento de problemas e incidencias, gestión de ramas y pull requests, sistemas de integración continua (CI) y despliegue continuo (CD), y herramientas de colaboración social.

En DolphinOS se han hecho uso de las siguientes características de GitHub:

- **Organizaciones:** Una organización [26] es una entidad dentro de GitHub que permite a los usuarios, equipos o empresas colaborar en proyectos de manera centralizada. Permite así gestionar los repositorios de una forma más eficaz entre los miembros de un equipo. En el caso de DolphinOS, se ha tomado la decisión de que sus repositorios formen parte de una organización debido a que si en un futuro este proyecto empieza a tomar tracción y se convierte en un proyecto mantenido con una comunidad detrás, la mejor forma de gestionar los repositorios es a través de una organización.

- **Repositorios:** A través de los repositorios online de GitHub, se puede mantener una fuente remota para los repositorios locales Git. De esta forma todo el código y el desarrollo queda reflejado abiertamente en la red.

Todos los repositorios de DolphinOS son repositorios públicos de código libre.

- **GitHub Actions:** GitHub Actions [27] es un sistema de integración continua y despliegue continuo (CI/CD). Permite la automatización de flujos de trabajo dentro de los repositorios, proporcionando un entorno flexible y personalizable.

A través de flujos de trabajo (workflows), se pueden definir las acciones que se deben realizar en respuesta a determinados eventos específicos.

Los GitHub Actions toman un papel fundamental en el desarrollo de DolphinOS, puesto que se han desarrollado varios sistemas de despliegue continuo (CD) para poder, por un lado, compilar y empaquetar los paquetes de la distribución y publicar el repositorio de la distribución en GitHub Pages, y por otro lado, permiten construir el sistema base y la imagen ISO de instalación de DolphinOS.

De esta forma, todo el proceso de desarrollo queda publicado y es transparente a cualquier usuario.

- **GitHub Pages:** GitHub Pages [28] es un servicio proporcionado por GitHub que permite a los usuarios publicar sitios web estáticos directamente desde sus repositorios de GitHub. En DolphinOS se hace uso de GitHub pages para poder publicar y alojar el repositorio de la distribución. Cuando el gestor de paquetes pacman realiza una petición a la dirección URL del repositorio online, este le devuelve un fichero de base de datos con los datos y URL de descarga de los paquetes contenidos en dicho repositorio.

2.3.2 - SourceForge

SourceForge [29] es una plataforma de desarrollo de software y alojamiento de proyectos que ha estado en funcionamiento desde 1999. Fue una de las primeras plataformas en ofrecer servicios de alojamiento de código fuente, seguimiento de problemas y colaboración en proyectos de código abierto.

En el caso de DolphinOS, SourceForge se ha utilizado para almacenar las imágenes ISO de instalación del sistema, utilizando su función del servicio de portal de descargas y distribución de archivos. Tras construirse las imágenes ISO desde las GitHub actions, se envían mediante el protocolo rsync al sistema de archivos virtual del portal de descargas y distribución de archivos de SourceForge, automatizando todo el proceso de publicación de las imágenes. De esta forma, un usuario puede dirigirse al apartado de archivos de proyecto DolphinOS de SourceForge, y descargarse la última imagen ISO compatible con su sistema.

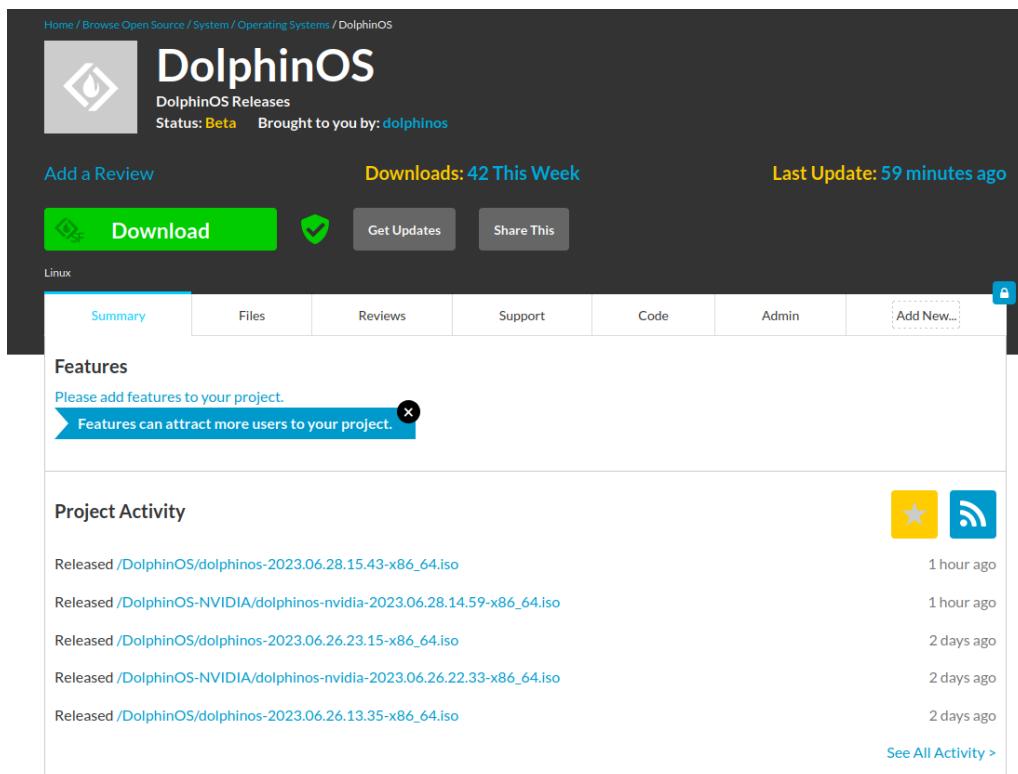


Figura 11: Portal del proyecto DolphinOS en SourceForge

3 - Desarrollo del proyecto

Tal y como se estipula en la planificación en el apartado 1.3 de esta memoria, el proyecto se ha dividido en seis fases diferentes.

En este apartado se hará un desglose de cada una de las fases del proyecto, explicando cómo se ha ido realizando cada acción.

3.1 - Fase 1: Investigación

Durante esta fase se realizó una investigación sobre cómo plantear y enfocar el proyecto.

La idea que se tenía clara era que el proyecto tenía que ser la construcción de una distribución Linux que sirviese como base de un emulador de una videoconsola que soportase funcionalmente iniciar el menú del sistema, pudiendo cargar juegos y aplicaciones de la consola emulada, todo ello sin salir del propio emulador.

De esta forma, se lograría ofrecer al usuario una experiencia muy cercana a la que ofrece la videoconsola original.

Como se explicó en el apartado 2.2, los emuladores RPCS3 y JPCSP contaban con una serie de problemas, ya fuesen por imposibilidad de cargar juegos desde el menú del sistema, problemas técnicos o de rendimiento, que no los hacían viables para este proyecto. Por tanto, se seleccionó la videoconsola Wii y su emulador Dolphin como el objetivo de este proyecto.

Una vez seleccionado la videoconsola y emulador, se necesitaba encontrar una distribución Linux ideal para este proyecto.

Tras una investigación, se encontraron varias distribuciones que podrían ser posibles candidatas. Entre ellas destacaban Universal Blue y Arch Linux. Cada una tenía una serie de ventajas e inconvenientes, las cuales se listan a continuación:

3.1.1 - Universal Blue (uBlue)

Universal Blue [30] (también llamado uBlue) es una distribución Linux basada en Fedora Silverblue [31]. Tanto Fedora Silverblue como uBlue toman un enfoque diferente a los sistemas operativos tradicionales. Uno de los puntos clave de estas distribuciones es que son sistemas operativos inmutables.

Los sistemas operativos inmutables son una forma de diseñar sistemas operativos de forma en la que el sistema raíz se mantiene en un estado fijo y no puede (o no debe) ser modificado durante la ejecución. En lugar de realizarse las actualizaciones e instalaciones de paquetes en tiempo real, los sistemas se distribuyen como imágenes estáticas en las que contienen todo el software y configuraciones necesarias.

En este caso, la partición raíz y los directorios del sistema formarían parte del sistema inmutable, mientras que el directorio de los usuarios (por ejemplo, /var/home/<nombre_usuario>) quedaría de forma mutable.

Fedora Silverblue y derivados utilizan ostree [32] como componente para gestionar el sistema inmutable.

Tal y como se describe en su página oficial, ostree puede ser resumido como “un sistema Git para los binarios de un sistema operativo”. Esto significa que ostree toma un enfoque de un repositorio Git para gestionar las configuraciones y paquetes de un sistema operativo inmutable.

Gracias a al gestor de paquetes rpm-ostree, se pueden realizar actualizaciones atómicas, es decir, las actualizaciones se engloban en una única imagen, que al instalarla, se despliega la imagen creando una nueva capa sobre el sistema de archivos.

La mayor ventaja de este sistema es que si se produce cualquier error mientras se actualiza el sistema, se puede volver rápidamente a un estado anterior, simplemente cargando al arranque la última capa funcional de ostree.

En comparación con Fedora Silverblue, Universal Blue va un paso más allá. Universal Blue incluye características adicionales, como por ejemplo, llevar los drivers propietarios de las tarjetas gráficas de NVIDIA preinstalados o que las imágenes son customizables.

uBlue ofrece un enfoque similar a las imágenes que se despliegan en máquinas virtuales en sistemas de cómputo en la nube, lo cual permite que dichas imágenes sean customizables.

uBlue hubiera sido una opción ideal para este proyecto, puesto que permite desplegar un sistema customizado y preconfigurado para el objetivo de este proyecto, teniendo las ventajas de seguridad y estabilidad que ofrecen los sistemas inmutables.

Sin embargo, es una distribución que está todavía en una fase temprana de desarrollo, y su instalador, Anaconda, todavía no es customizable, por lo cual sería tedioso adaptar el sistema al objetivo de DolphinOS. Por este motivo, se descartó esta distribución

3.1.2 - Arch Linux

Arch Linux [33] es una distribución Linux independiente, ligera y flexible.

Entre sus ventajas se encuentra que es una distribución muy simple por defecto.

La instalación normalmente se realiza mediante terminal, pero gracias a su gran documentación, la cual es otra de sus ventajas, se puede realizar sencillamente siguiendo los pasos indicados en la documentación. Esta documentación, denominada ArchWiki, es considerada una de las mayores fuentes de conocimiento e información, no sólo de Arch Linux, si no que también es de gran utilidad para la configuración y administración de otras distribuciones. Su modelo de actualización sigue un enfoque conocido como “rolling release”, es decir, la distribución está constantemente actualizándose, no existen versiones de la distribución como tal. Este modelo de distribución no es adecuado para todos los usos, como puede ser en servidores, donde se requieren sistemas más estables (estable en cuanto a invariación a lo largo del tiempo, que no necesariamente ha de significar que está más exenta de bugs y fallos). Cuenta adicionalmente con un amplio catálogo de software en sus repositorios, y a través de las PKGBUILDS se pueden definir scripts de empaquetado e instalación para poder crear paquetes e instalarlos en el sistema.

La mayor desventaja frente a Universal Blue es que es un sistema operativo que sigue el enfoque tradicional. Sin embargo, existen métodos para hacer que un sistema sea resistente a errores en actualizaciones / instalación de paquetes, los cuales pueden corromper el sistema, sin necesidad de tener que hacer uso de un sistema operativo inmutable.

Teniendo en cuenta lo anterior, y gracias a las ventajas en ligereza y customización, Arch Linux se seleccionó como distribución candidata para servir de base de DolphinOS.

3.1.3 - Software del sistema

Una vez elegido el emulador y la videoconsola a emular, y el sistema operativo sobre el cual se asentará el emulador, se procede a elegir el software necesario para el sistema.

Se comienza con la elección del sistema de ventanas. Actualmente en el entorno Linux existen dos protocolos / sistemas de ventanas, X11 y Wayland:

- **X11** [34]: El sistema de ventanas X es un sistema de ventanas comúnmente usado en sistemas UNIX, que fue desarrollado a mediados de la década de 1980. La versión actual es X11. X11 sigue el enfoque cliente servidor. El servidor provee servicios para acceder a la pantalla, teclado y ratón, mientras que los clientes son las aplicaciones que utilizan estos recursos para interacción con el usuario. La comunicación entre el cliente X y el servidor se realiza por medio de un protocolo conocido como XProtocol.

A lo largo de su historia X ha tenido distintas implementaciones, siendo X.Org la implementación canónica de X. X.Org nació como un proyecto con el objetivo de unificar las diversas implementaciones y funcionalidades del sistema de ventanas X. Sin embargo, tanto X11 como X.Org llevan casi 10 años en estado de mantenimiento. Esto es debido a que las características de X y X.Org no se ajustan a los requisitos de los sistemas actuales, y presentan ciertas carencias en cuestiones de seguridad. La deuda técnica que acarrean ambos proyectos, hace que sea prácticamente imposible implementar estas nuevas características.

Por este motivo, los propios desarrolladores decidieron desarrollar un nuevo sistema de ventanas, Wayland.

- **Wayland** [35]: Es el sucesor canónico al sistema de ventanas X. Tal y como se explicó anteriormente, Wayland pretende ser un reimplementación limpia del sistema de ventanas X, corrigiendo sus carencias de seguridad y añadiendo características las cuales no podían ser agregadas a X.Org por motivos técnicos.

Al contrario que en X, en vez de existir un único servidor X.Org y el resto de aplicaciones son clientes, incluidos gestores de ventanas y compositores de efectos de escritorio, en Wayland cada gestor de ventanas es un servidor gráfico independiente, llamado “compositor”.

Todos los esfuerzos en la comunidad de desarrolladores de Linux de escritorio están actualmente enfocados a migrar tanto gestores de ventanas como las aplicaciones a Wayland. Wayland es el futuro del sistema de gestión de ventanas en sistemas Linux.

Para que funcione correctamente el protocolo, los drivers de las tarjetas gráficas han de dar soporte al protocolo adecuadamente. Sin embargo, cuando las bases del protocolo empezaron a definirse en conjunción con los principales fabricantes de tarjetas gráficas a principios de 2010, el fabricante de tarjetas gráficas NVIDIA se negó inicialmente a dar soporte a Wayland. Sólo comenzó a dar soporte inicial a finales de la década de los 2010, y a través de sus propios protocolos propietarios (EGLStreams) en vez de adoptar los estándares de Wayland (GBM). Finalmente en 2021, NVIDIA comenzó a adoptar los estándares de Wayland y a dar soporte a la aceleración gráfica XWayland.

XWayland [36] es un servidor X.Org anidado que permite poder ejecutar las aplicaciones que todavía no han sido portadas para tener soporte en Wayland.

El mayor problema que existe actualmente es que no existe un método efectivo de sincronización entre el driver propietario de NVIDIA y XWayland. Esto resulta en que los fotogramas cuando se muestran tras ser renderizados se muestran en un orden arbitrario, produciendo corrupción gráfica. Existe bastante discrepancia [37] por parte de los desarrolladores de tanto XWayland como de NVIDIA. Las soluciones consisten en implementar la sincronización implícita en el driver propietario de NVIDIA (lo cual se niegan) o implementar sincronización explícita en stack gráfico de Linux (XWayland, KMS y Linux Kernel), lo cual tardaría bastantes años en implementarse.

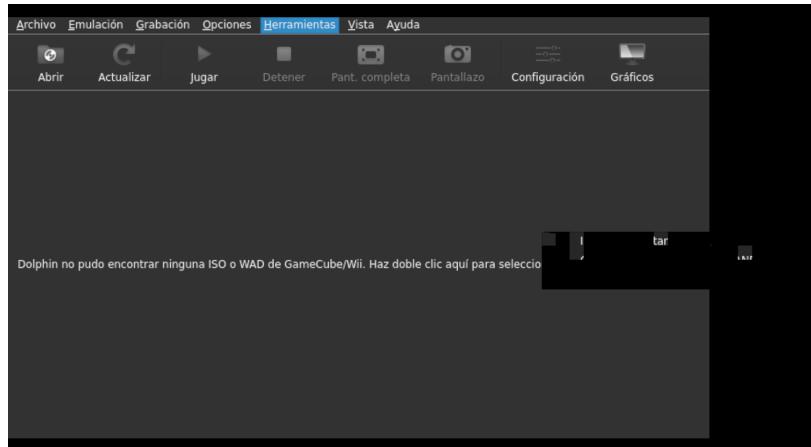


Figura 12: Corrupción gráfica apreciable en aplicaciones XWayland en NVIDIA

Por este motivo, pese a que la idea inicial era utilizar el compositor de Wayland, Gamescope, el cual está pensado para albergar ventanas relacionadas con los videojuegos, se decidió por utilizar el sistema de ventanas X11 con el servidor X.Org y el gestor de ventanas OpenBox, para todos los sistemas.

En cuanto al sistema de audio, se decidió utilizar el servidor de audio PipeWire, el cual es una implementación superior comparado con el antiguo sistema de audio, PulseAudio.

Para evitar que, en caso de producirse un error al actualizar o instalar un programa se corrompa el sistema, se ha hecho uso del sistema de archivos BTRFS [38].

BTRFS ofrece una función denominada instantánea [40]. Las instantáneas, o snapshots, son una copia puntual del estado del sistema de archivos en un momento específico. Una snapshot captura el estado de todos los datos, metadatos y estructuras del sistema de archivos en un punto determinado y permite mantener ese estado de forma independiente de las modificaciones que ocurran posteriormente. Occupan muy poco espacio, ya que sólamente conservan los cambios producidos desde la instantánea anterior. Además, gracias a otra característica de BTRFS, llamados subvolúmenes [39], se pueden almacenar dentro del propio sistema de archivos. Los subvolúmenes son entidades lógicas que actúan como una unidad independiente dentro del sistema de archivos. Puede considerarse como una especie de directorio virtual que se comporta como un sistema de archivos independiente, con su propio conjunto de archivos, directorios y atributos.

Para poder gestionar instantáneas, se hace uso del software Snapper [41], el cual se puede configurar para que antes y después de realizar una actualización, instalación o desinstalación desde el gestor de paquetes pacman se tome una instantánea del sistema de archivos. Si la actualización fallase, se podría arrancar desde una instantánea anterior desde el gestor de arranque GRUB, y restaurar esa instantánea, sobreescritiendo el subvolumen objetivo con la instantánea, evitando así una posible corrupción del sistema.

Por lo que respecta al sistema de instalación del sistema DolphinOS en un ordenador, se ha hecho uso del framework de instalación Calamares [42].

Calamares es un framework de instalación multidistribución, el cual permite a los desarrolladores de las distribuciones crear un instalador para su distribución.

Gracias a su diseño modular, las distribuciones pueden ajustarlo a sus necesidades sin tener que modificar la forma en la que funciona.

De cara al usuario, Calamares proporciona una interfaz sencilla para que un usuario pueda instalar el sistema en su equipo con tan solo un par de clicks.

Además, incluye un sistema de particionado bastante ameno para el usuario.

En DolphinOS, un usuario puede elegir si desea eliminar todos los datos del dispositivo de almacenamiento e instalar el sistema en dicho dispositivo. También se le ofrece la opción de redimensionar o reemplazar una partición del dispositivo de almacenamiento, dejando intactas las particiones que no se han modificado. En todo momento se va informando y avisando al usuario de las acciones que se van a realizar en el sistema.

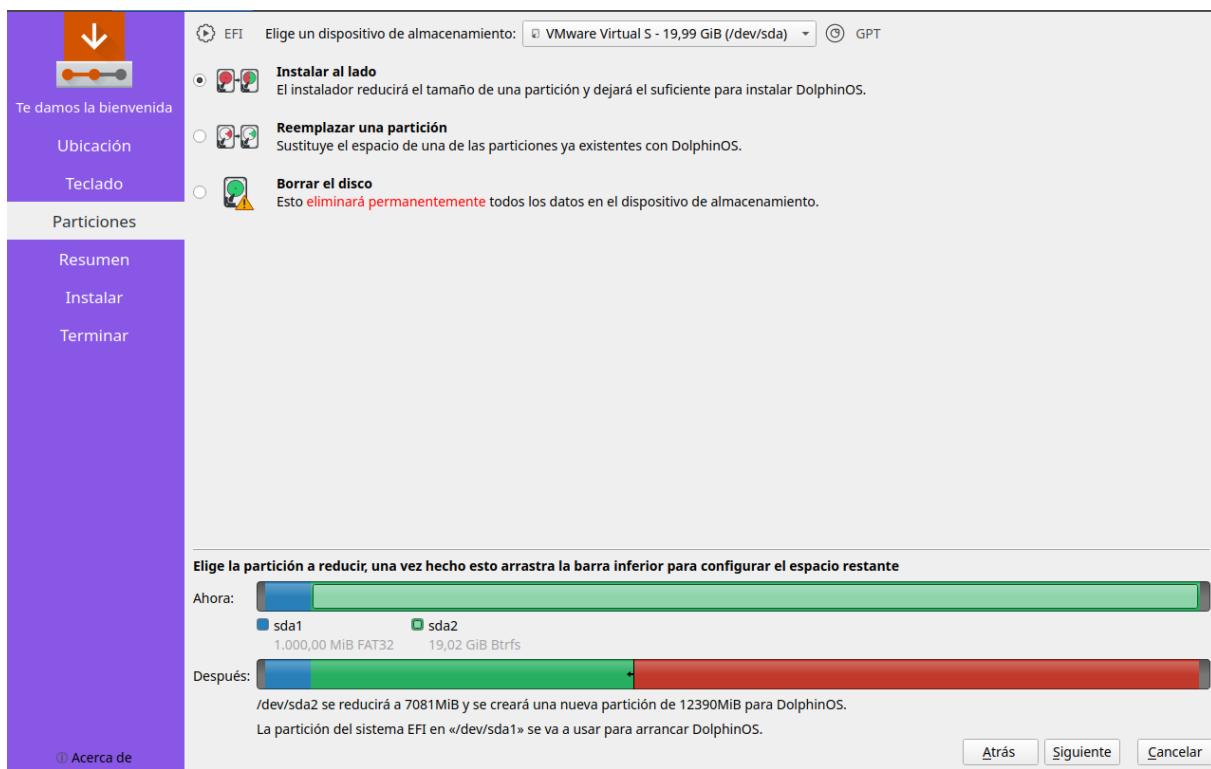


Figura 13: Framework de instalación Calamares en DolphinOS

Finalmente, se tomó la decisión de crear dos versiones de DolphinOS.

Una versión principal, compatible con las tarjetas gráficas Intel, AMD y los gráficos virtualizados de VMWare, y otra versión compatible solamente con las tarjetas gráficas de NVIDIA. Esto es debido a la naturaleza de los drivers propietarios de NVIDIA, los cuales necesitan una serie de configuraciones diferentes en el sistema comparado con la versión principal. En ciertos repositorios del proyecto, existe una rama principal y una rama nvidia, la cual incluye las configuraciones específicas para estas tarjetas gráficas.

A la hora de la creación de los paquetes del sistema, se crean paquetes específicos para NVIDIA, basándose en dicha rama de su repositorio correspondiente.

3.2 - Fase 2: Infraestructura de desarrollo en GitHub

Para el desarrollo de DolphinOS, se han hecho uso de los servicios de DolphinOS. Esta fase comienza con la creación de una organización de GitHub. Posteriormente, se procede a crear los repositorios necesarios para la publicación y la gestión de las versiones del código. Finalmente, se configuran los sistemas de despliegue continuo (CD) mediante GitHub Actions, y la publicación del repositorio de la distribución mediante GitHub Pages.

3.2.1 - Organización

La creación de una organización se realiza por si en un futuro el proyecto tomase tracción y el desarrollo fuese llevado a cabo por varios desarrolladores, se pudieran gestionar los repositorios de una manera más sencilla por los miembros de la organización. Para ello, se crea una nueva organización, y por el momento, se establece el plan gratuito, el cual es suficiente para el alcance de este proyecto.

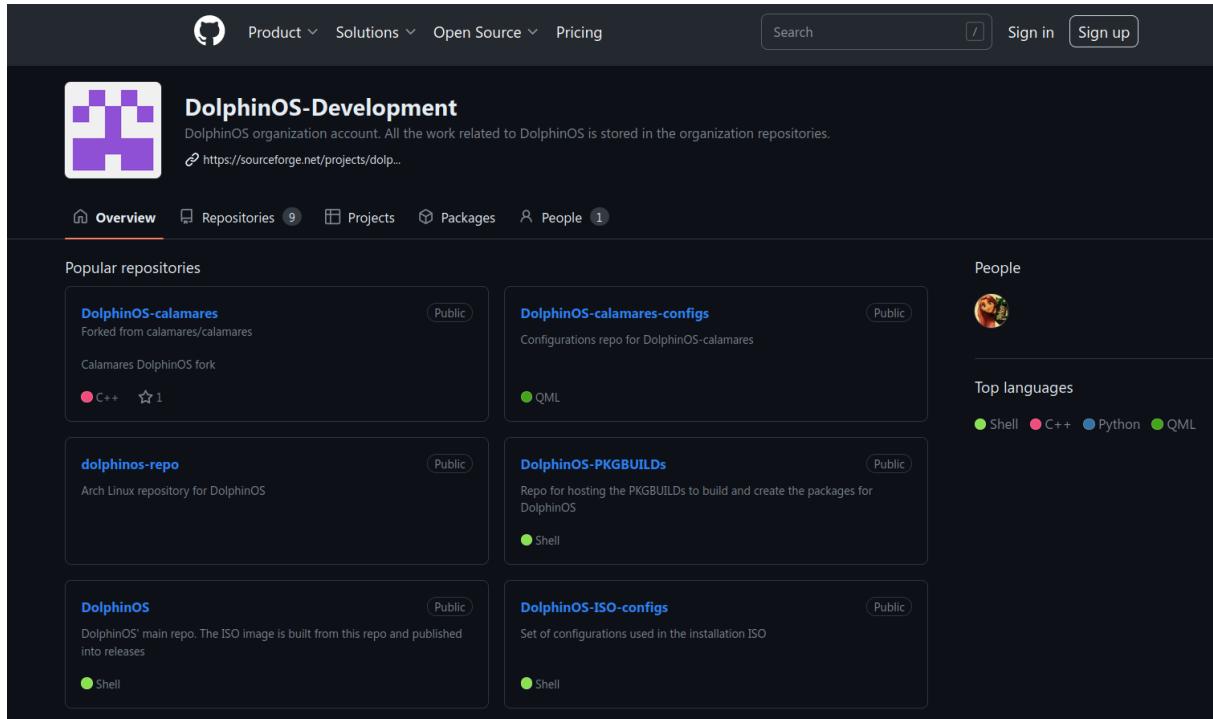


Figura 14: Organización DolphinOS-Development en GitHub

3.2.2 - Repositorios

Tras establecer la organización para el proyecto, el siguiente paso es crear los repositorios necesarios para el desarrollo del proyecto. En ellos se va a ir alojando todas las configuraciones y código necesarios para el funcionamiento de DolphinOS.

A continuación, se desglosan los repositorios que forman parte de DolphinOS:

- **DolphinOS:** Es el repositorio principal del proyecto. En él se alojan los ficheros necesarios para la construcción del sistema base (fichero squashfs) y la imagen ISO de instalación del sistema, los cuales son construidos a través del flujo de trabajo de las GitHub Actions. Cuenta con dos ramas, la rama principal, de la cual se crea la ISO principal, y la rama nvidia, de la cual se crea la ISO con las configuraciones específicas para sistemas con gráficas NVIDIA.
- **DolphinOS-calamares:** Este repositorio es un fork del repositorio oficial de Calamares. La creación del fork es debido a la futura necesidad de modificar o introducir módulos nuevos al framework de instalación.
- **DolphinOS-calamares-configs:** En este repositorio se alojan los ficheros de configuración necesarios para Calamares, tanto para su diseño del estilo visual de la aplicación, como lo que se va a presentar al usuario en la interfaz, y las acciones a realizar por el instalador para la instalación del sistema. Cuenta con dos ramas, la principal y la nvidia, cada una con sus configuraciones específicas.
- **DolphinOS-configs:** Aquí se ubican todas las configuraciones del sistema DolphinOS. En vez de establecerse sobre la imagen de instalación o copiarse al sistema individualmente tras finalizar la instalación, se crea un paquete de Arch Linux a partir de este repositorio. De esta forma, si en un futuro se realiza alguna modificación en la configuración del sistema, esta le pueda llegar a todos los usuarios, sin tener que reinstalar el sistema. Al igual que otros repositorios, incluye la rama nvidia con sus configuraciones específicas para los sistemas NVIDIA.
- **DolphinOS-DolphinData:** En este repositorio se alojan tanto los ficheros de configuración del emulador Dolphin, como el firmware del sistema Wii emulado y las aplicaciones caseras (homebrews) almacenadas en la tarjeta de memoria SD virtual.
- **DolphinOS-ISO-configs:** Al igual que en DolphinOS-configs, se alojan las configuraciones del sistema. Sin embargo, estas configuraciones son específicas únicamente de la imagen ISO de instalación. Tras realizar la instalación, este paquete se desinstala del sistema, limpiándolo de los ficheros de instalación.
- **DolphinOS-PKGBUILDs:** Este repositorio sirve para alojar los scripts de creación de paquetes e instalación de Arch Linux. En cada subdirectorio se encuentra un fichero PKGBUILD, que incluye las instrucciones necesarias para la construcción / empaquetado / instalación del paquete a crear.

- **dolphinos-repo:** Este es repositorio de la distribución. En él se alojan en un subdirectorio (cuyo nombre indica la arquitectura del sistema) los paquetes contenidos en el repositorio dolphinos-repo del gestor de paquetes pacman, junto con el fichero dolphinos-repo.db. Este fichero permite al gestor de paquetes verificar si existen actualizaciones de los paquetes, y proporcionar las URLs de los paquetes al gestor de paquetes para poder descargárselos. Este repositorio se publica online mediante las GitHub Pages. Además, gracias a las GitHub actions, en este repositorio se crean los paquetes que formarán parte del repositorio, usando las PKGBUILDs del repositorio DolphinOS-PKGBUILDs.
- **DolphinOS-WebApp:** En este repositorio se aloja el código de la aplicación web DolphinOS-WebApp.

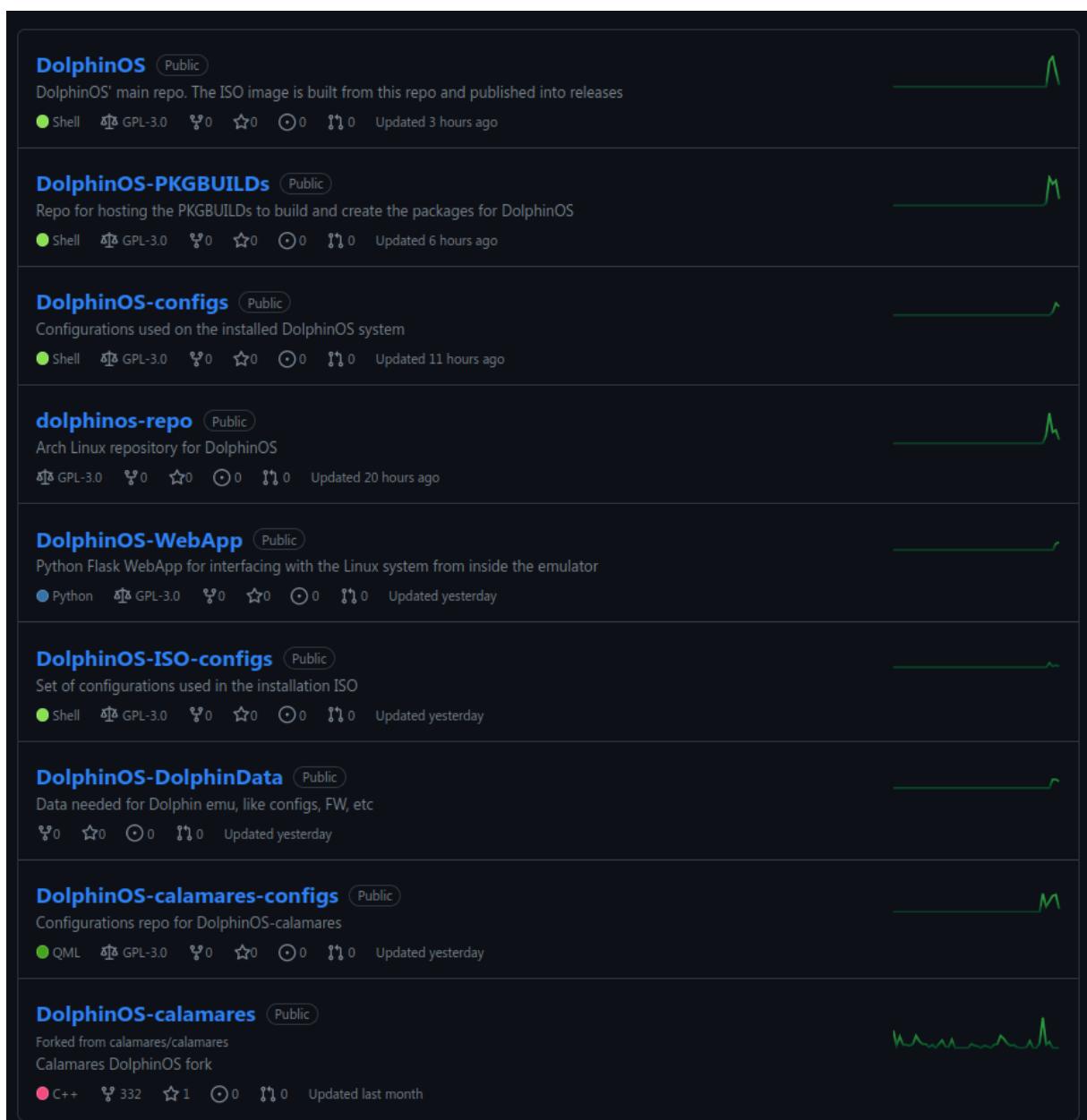


Figura 15: Repositorios del proyecto DolphinOS

3.2.3 - GitHub Pages

Gracias a GitHub Pages se pueden construir sitios web estáticos para cualquier tipo de uso. Puesto que los repositorios del gestor de paquetes pacman de Arch Linux utilizan el protocolo HTTP o HTTPS, se puede crear un repositorio de ArchLinux [43] manteniendo la siguiente estructura:

```
/home/archie/customrepo/
└── arch
    ├── customrepo.db -> customrepo.db.tar.xz
    ├── customrepo.db.tar.xz
    ├── customrepo.files -> customrepo.files.tar.xz
    ├── customrepo.files.tar.xz
    └── personal-website-git-b99cce0-1-arch.pkg.tar.zst
```

Donde “arch” representa la arquitectura del sistema objetivo, en este caso x86_64. Para publicar en línea el repositorio de la distribución, solamente es necesario seguir esa organización de archivos, y en la configuración del repositorio de GitHub, en el apartado de Pages, seleccionar la fuente desplegar desde una rama, y en la selección de rama seleccionar la rama principal. Tras presionar en guardar, automáticamente ejecutará una GitHub Action que desplegará los contenidos del repositorio en una página web estática usando Jekyll.

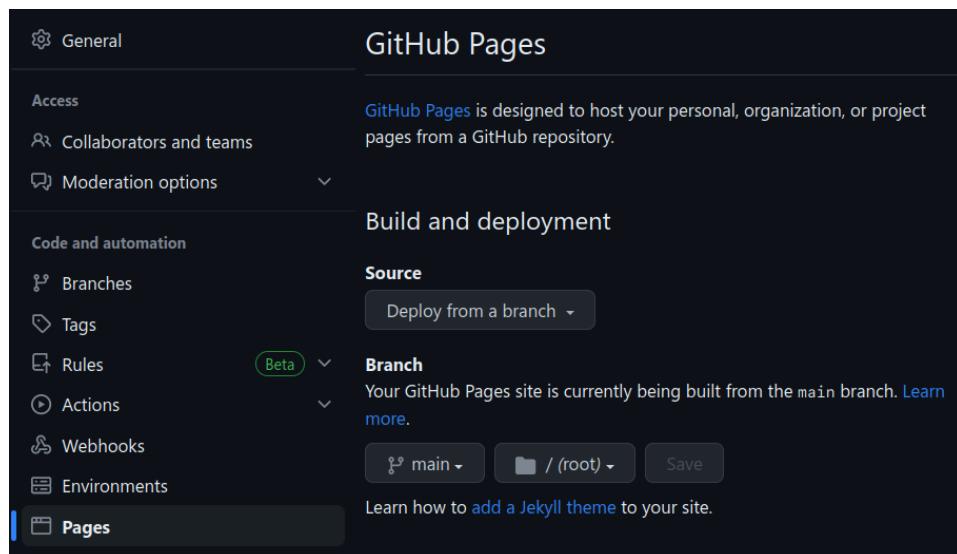


Figura 16: Despliegue del repositorio de la distribución usando GitHub Pages

3.2.4 - GitHub Actions

En DolphinOS se hace uso de las GitHub Actions con la finalidad de lograr dos objetivos. Por un lado, se utilizan para poder construir los paquetes del repositorio de la distribución, y compilar los programas que requieran de compilación previa antes de su empaquetado. Por otro lado, se hace uso de un flujo de trabajo para poder construir las imágenes ISO de instalación del sistema y su sistema base.

En cualquiera de las GitHub Actions, si en alguno de los pasos del flujo de trabajo se produjese un error, la ejecución de dicha Action fallaría. Los ficheros de flujo de trabajo se pueden encontrar bajo el directorio .github/workflows de cada repositorio.

A continuación, se detalla más en profundidad cada una, en función al repositorio al que pertenecen.

- **dolphinos-repo:** Esta GitHub Action se encarga primeramente de clonar el repositorio DolphinOS-PKGBUILDs, con el fin de obtener las PKGBUILDs. Una vez clonado el repositorio, a través de un bucle for, se va iterando sobre cada uno de los directorios, en los que se halla el script PKGBUILD de construcción de paquetes, para así poder construir los paquetes del repositorio de la distribución, y compilar los programas que requieran de compilación previa antes de su empaquetado. Previamente a la construcción se realiza una configuración inicial del contenedor Arch Linux sobre el cual se ejecuta el flujo de trabajo de la GitHub action, y se eliminan los paquetes construidos anteriormente. Tras construir los paquetes, se añade la información de los paquetes al fichero del repositorio, se crea un commit nuevo con los paquetes construidos y se suben al repositorio de GitHub.
- **DolphinOS:** El flujo de trabajo de esta GitHub action comienza estableciendo la configuración necesaria en el contenedor Arch Linux para poder crear las imágenes ISO de instalación, como la creación de un usuario no privilegiado para realizar acciones que necesitan ser ejecutadas por un usuario no privilegiado, o la instalación de dependencias. A continuación se define una variable del entorno con la fecha y hora correspondiente al momento de ejecución, la cual se utilizará para crear un nuevo tag de release en el apartado de releases del repositorio de GitHub. Posteriormente, obtienen los datos de la rama nvidia del repositorio, y se descargan en un directorio de dentro del sistema base (airootfs) los paquetes que incluyen las configuraciones del sistema y del emulador Dolphin, pero sin llegar a instalarlos. Esto se realiza de esta forma para que no se produzca un conflicto con las configuraciones del sistema ISO de instalación. Tras instalarse el nuevo sistema en un ordenador, Calamares instalará estos paquetes en él.
Tras tener todo listo, se procede a construir la ISO de instalación utilizando el script mkarchiso proporcionado por Arch Linux. Este script se encargará de descargar e instalar los paquetes, construir la imagen del sistema base (airootfs.sfs) y la imagen ISO de instalación.

Una vez finalizada la construcción de la imagen ISO, esta se sube al portal de almacenamiento de archivos de SourceForge, desde donde podrán descargarla los usuarios. Para evitar que la clave privada SSH para poder enviar los datos a SourceForge sea publicada públicamente, esta se oculta usando la funcionalidad Action secrets en los ajustes del repositorio de GitHub. En los ficheros de registro aparece censurada y no se publica en el repositorio de ninguna forma.

Estas acciones se realizan de igual forma para la imagen ISO principal.

A continuación, se muestran unas capturas de pantalla de ambas Actions:

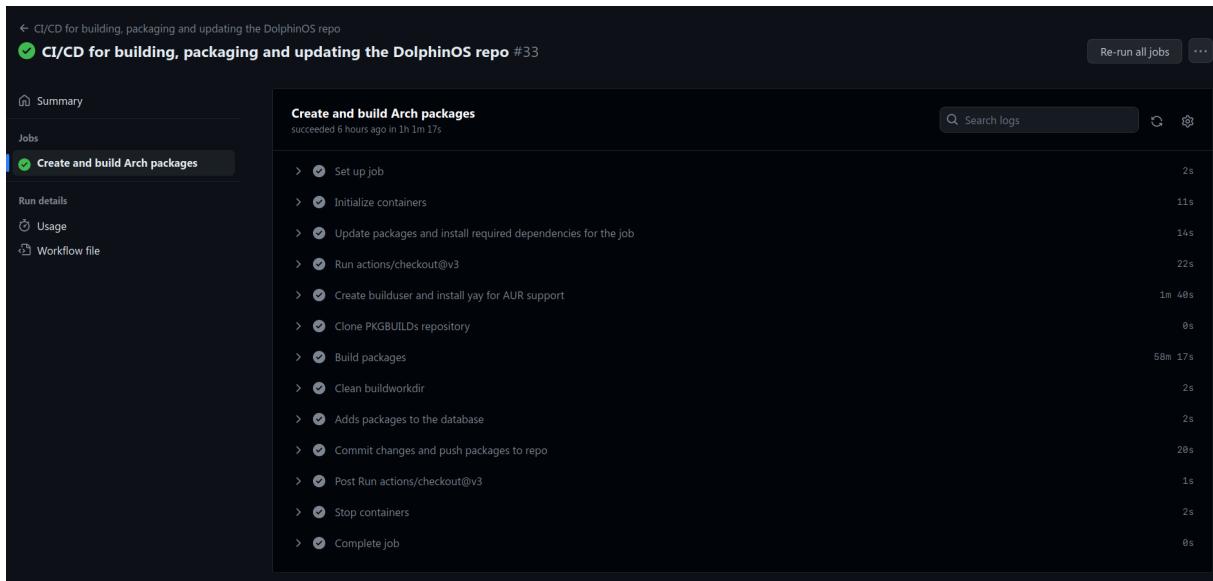


Figura 17: GitHub Action encargada de construir los paquetes de la distribución

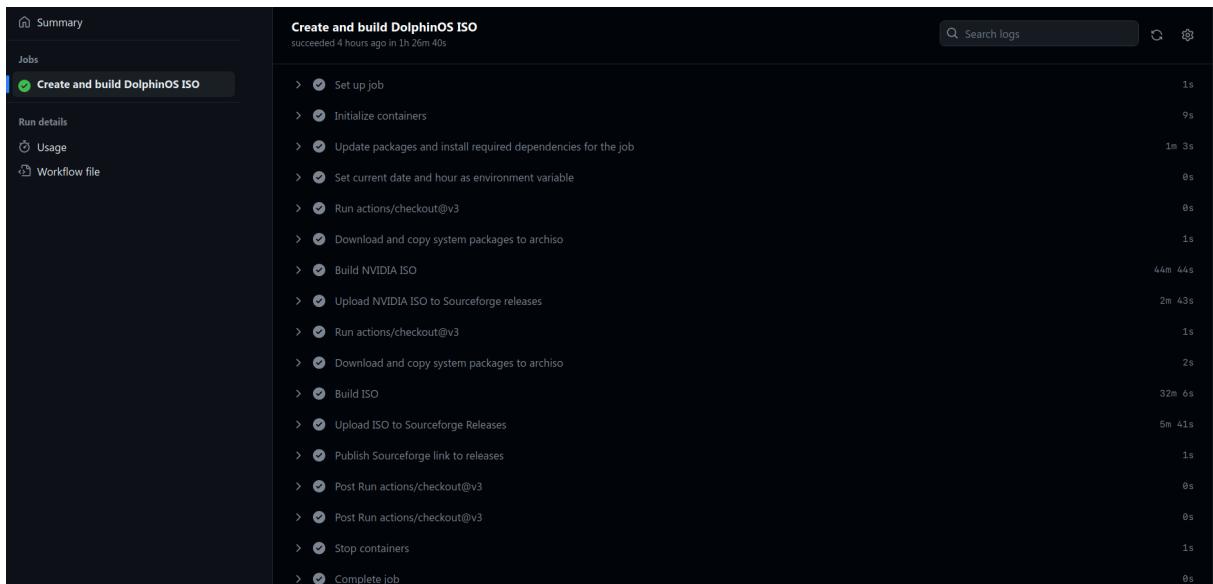


Figura 18: GitHub Action encargada de construir la imagen ISO de instalación

3.3 - Fase 3: Creación de la imagen ISO de instalación

Una vez definida la infraestructura de desarrollo y despliegue continuo (CD) en GitHub, se procede a crear el medio de instalación de DolphinOS.

Los ficheros resultantes de esta fase son dos imágenes ISO arrancables (una versión que soporta las GPUs de Intel, AMD y VMWare, y otra versión con soporte único para NVIDIA). Estas imágenes ISO se pueden grabar en un disco DVD o en una memoria flash (mediante Balena Etcher, dd, Rufus, ...). Una vez grabadas, se pueden usar dichos dispositivos de almacenamiento para arrancar el ordenador desde ellos, y poder así cargar el software de instalación de DolphinOS.

3.3.1 - Archiso

En cuanto al proceso de creación de la imagen ISO de instalación es necesaria la herramienta Archiso [44] de Arch Linux. Archiso incluye dos perfiles para la creación de la ISO, releng y baseline. Releng es el perfil utilizado por Arch Linux para crear la imagen ISO de instalación oficial, mientras que baseline es una versión mínima de Archiso en la cual sólamente se proporcionan las configuraciones necesarias para arrancar el sistema Live desde el medio de instalación. En el caso de DolphinOS, se hace uso del perfil releng, el cual se customiza para adaptarse a la configuración de DolphinOS.

El esquema de la distribución de ficheros del perfil Archiso customizado de DolphinOS es el siguiente:

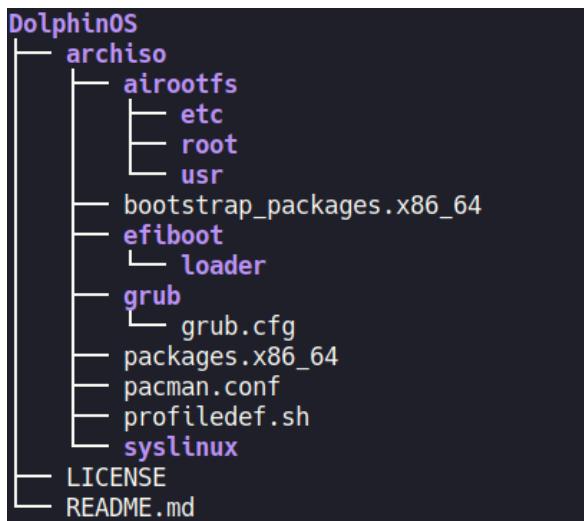


Figura 19: Esquema de directorios del perfil Archiso de DolphinOS

Dentro del directorio archiso se encuentran todos los ficheros de configuración para crear una imagen ISO dada dicha configuración.

El script profiledef.sh es el fichero de configuración principal. En él se establece el nombre de la imagen ISO, la etiqueta, la entidad publicadora, la versión, arquitectura, modos de arranque y los permisos iniciales del sistema, entre otros.

```
#!/usr/bin/env bash
# shellcheck disable=SC2034

iso_name="dolphinos"
iso_label="DOLPHINOS_$(date --date="@${SOURCE_DATE_EPOCH}-$(date +%s)}"
+%Y%m%d%H%M)"
iso_publisher="DolphinOS <https://github.com/DolphinOS-Development>"
iso_application="DolphinOS installation ISO"
iso_version="$(date --date="@${SOURCE_DATE_EPOCH}-$(date +%s)}"
+%Y.%m.%d.%H.%M)"
install_dir=dolphinos
buildmodes=('iso')
bootmodes=('bios.syslinux.mbr' 'bios.syslinux.eltorito'
'uefi-x64.systemd-boot.esp' 'uefi-x64.systemd-boot.eltorito')
arch=x86_64
pacman_conf=pacman.conf
airootfs_image_type=squashfs
airootfs_image_tool_options=(-comp xz -Xbcj x86 -b 1M
-Xdict-size 1M)
file_permissions=(
  [/etc/shadow]=0:0:400
  [/root]=0:0:750
  [/root/.automated_script.sh]=0:0:755
  [/usr/local/bin/choose-mirror]=0:0:755
  [/usr/local/bin/Installation_guide]=0:0:755
  [/usr/local/bin/livecd-sound]=0:0:755
)
```

Figura 20: Script de definición del perfil de Archiso

Por otro lado, en el fichero packages.x86_64, se agregan todos los paquetes necesarios tanto para la imagen de instalación como para el sistema instalado, a excepción de los paquetes con las configuraciones de DolphinOS y de Dolphin. Tras realizarse la instalación del sistema, los paquetes que sólamente sean necesarios para el sistema de instalación serán desinstalados por Calamares.

En el fichero pacman.conf se establece la configuración del gestor de paquetes pacman. Para que el repositorio de la distribución de DolphinOS funcione tanto a la hora de instalar los paquetes en la imagen ISO de instalación como luego en el sistema instalado, se ha de añadir la siguiente línea al comienzo del apartado de los ficheros pacman.conf tanto del directorio archiso como del directorio archiso/airootfs/etc:

```
[dolphinos-repo]
SigLevel = Optional TrustedOnly
Server = https://jesusxd88.github.io/$repo/$arch
```

Bajo el directorio airootfs se ubican las configuraciones de Archiso para poder arrancar el sistema Live, las cuales serán copiadas al sistema de archivos virtual airootfs.sfs.

En los directorios efiboot y syslinux se establecen las configuraciones para el gestor de arranque.

Si un usuario decide arrancar desde el medio de instalación DolphinOS, primeramente se cargará el gestor de arranque desde la UEFI del ordenador.



Figura 21: Gestor de arranque systemd-boot del medio de instalación de DolphinOS

A partir de ahí, el cargador de arranque carga la imagen del Kernel del sistema y el sistema initramfs. En un sistema tradicional, una vez cargado el sistema init desde el initramfs, se montarían los sistemas de archivos y se pivotaría el directorio raíz (/) al sistema de archivos principal. En caso de los sistemas Live, se descomprime el sistema de archivos principal desde una imagen (airootfs.sfs) en un sistema de archivos alojado en la memoria RAM. De esta forma, se puede hacer uso del sistema sin modificar ninguno de los sistemas de almacenamiento del equipo.

Las configuraciones específicas de DolphinOS para el sistema Live se almacenan en el repositorio DolphinOS-ISO-configs. De este repositorio se crea un paquete el cual se instala en la fase de creación de la ISO de instalación.

Aquí se incluyen ciertas configuraciones, como por ejemplo, que se cargue el servidor X.Org tras realizarse el inicio de sesión automático, y que a su vez arranque el instalador Calamares en una sesión del gestor de ventanas OpenBox.

3.3.2 - Calamares

Tal y como se explicó en el apartado 4.1.3, Calamares es un framework de instalación modular multidistribución. En DolphinOS se utiliza como herramienta para poder instalar el sistema en un equipo. En la imagen ISO se incluyen dos paquetes de Calamares, por un lado dolphinos-calamares el cual es el software Calamares como tal, y por otro lado, dolphinos-calamares-configs. Ambos se encuentran disponibles en sus correspondientes repositorios de la organización DolphinOS-Development en GitHub.

Tanto las interfaces que se le presentan al usuario como las acciones a realizar para poder instalar el sistema correctamente se ubican en una serie de módulos.

Desde el fichero settings.conf se puede definir la secuencia en la que aparecerán dichas interfaces, y el orden de los pasos a seguir por el propio instalados para poder instalar el sistema. Se divide en dos partes, “show” para las interfaces que se presentan al usuario y “exec” para la secuencia de instalación.

Si múltiples acciones requieren el uso de un único módulo, se definen distintas instancias de dicho módulo. En cada instancia se establece el nombre de la instancia, módulo que utilizan y fichero de configuración de dicho módulo. En este caso, se hacen uso de distintas instancias del módulo “shellprocess”, el cual permite ejecutar un script de comandos en el sistema instalado:

```
instances:  
- id:      create_user  
  module:  shellprocess  
  config:  shellprocess-create_user.conf  
- id:      final  
  module:  shellprocess  
  config:  shellprocess-final.conf  
- id:      snapper  
  module:  shellprocess  
  config:  shellprocess-snapper.conf
```

Figura 22: Distintas instancias del módulo “shellprocess” de Calamares

Todos los ficheros de configuración de Calamares siguen el formato de archivo YAML (YAML Ain't Markup Language).

En la secuencia de instalación, la primera acción a realizar es definir el esquema de particionado y la creación de los sistemas de archivos.

En cuanto la estructura de particionado del sistema instalado, en DolphinOS se hace uso de dos particiones:

- **Sistema EFI:** Puesto que DolphinOS solamente es compatible con sistemas con firmware UEFI y no en con sistemas BIOS antiguos, es necesaria una partición EFI (ESP, EFI System Partition) en la cual se alojarán los ficheros EFI arrancables. Puesto que la mayoría de sistemas UEFI disponen de partición ESP, se instalarán los ficheros del cargador de arranque en dicha partición sin modificar ningún otro archivo contenido en ella. Si el dispositivo de almacenamiento no presentase partición ESP, Calamares crearía dicha partición.
- **Sistema de archivos BTRFS:** Se crea la partición principal del sistema bajo un sistema de archivos BTRFS. BTRFS ofrece el mismo grado de confiabilidad que ofrecería un sistema de archivos EXT4, pero con características añadidas, como la compresión por defecto, el sistema Copy-on-Write (CoW), deduplicación de archivos y soporte de instantáneas en el mismo dispositivo de almacenamiento. Se configura un único subvolumen (/@) en el cual irá ubicado el directorio raíz del sistema. Calamares permite definir una serie de opciones de montaje de los sistemas de archivos, dependiendo de si el dispositivo de almacenamiento es un disco duro (HDD) o una unidad de estado sólido (SSD). En el caso de que se instale sobre un SSD, se agregan opciones de compresión y optimización de la unidad flash, para reducir su desgaste. Por el contrario, si se instala sobre un disco duro, se habilita la desfragmentación automática.

Tras realizar el particionado y crear los sistemas de archivos, Calamares procede a desempaquetar la imagen del sistema a través del módulo “unpackfs”. En el caso de DolphinOS, se desempaquetan la imagen del sistema base (airootfs.sfs), creada en el proceso de creación de la imagen ISO, en el sistema de archivos raíz del nuevo sistema, y posteriormente, se copian bajo el directorio “boot” las imágenes del Kernel Linux.

A continuación, Calamares monta el nuevo sistema de archivos y procede a establecer la configuración esencial del sistema, como el esquema de montaje (fichero fstab), localización internacional y distribución del teclado. También genera las imágenes del initramfs en el nuevo sistema.

En el caso de DolphinOS, al ser un sistema preconfigurado para una tarea en específico, el usuario lo crea el sistema y no el usuario, por lo que se hace uso de una instancia de “shellprocess” para crear el usuario, asignarle contraseña, y permitir que pueda ejecutar comandos como superusuario (sudo). Se establece la configuración de red y de fecha y hora, y se habilitan los servicios de systemd.

Antes de finalizar la instalación, se desinstalan los paquetes y se eliminan las configuraciones que forman parte del medio de instalación, con el objetivo de limpiar el sistema instalado.

Finalmente, se instalan los paquetes con las configuraciones de DolphinOS y del emulador Dolphin, se instala y configura el gestor de arranque GRUB en el nuevo sistema para que pueda ser arrancable, y se configura el sistema de gestión de instantáneas Snapper.

A continuación, se muestran unas capturas de pantallas del instalador Calamares de DolphinOS:



Figura 23: Interfaz de bienvenida de Calamares



Figura 24: Interfaz de bienvenida de Calamares en la versión DolphinOS NVIDIA

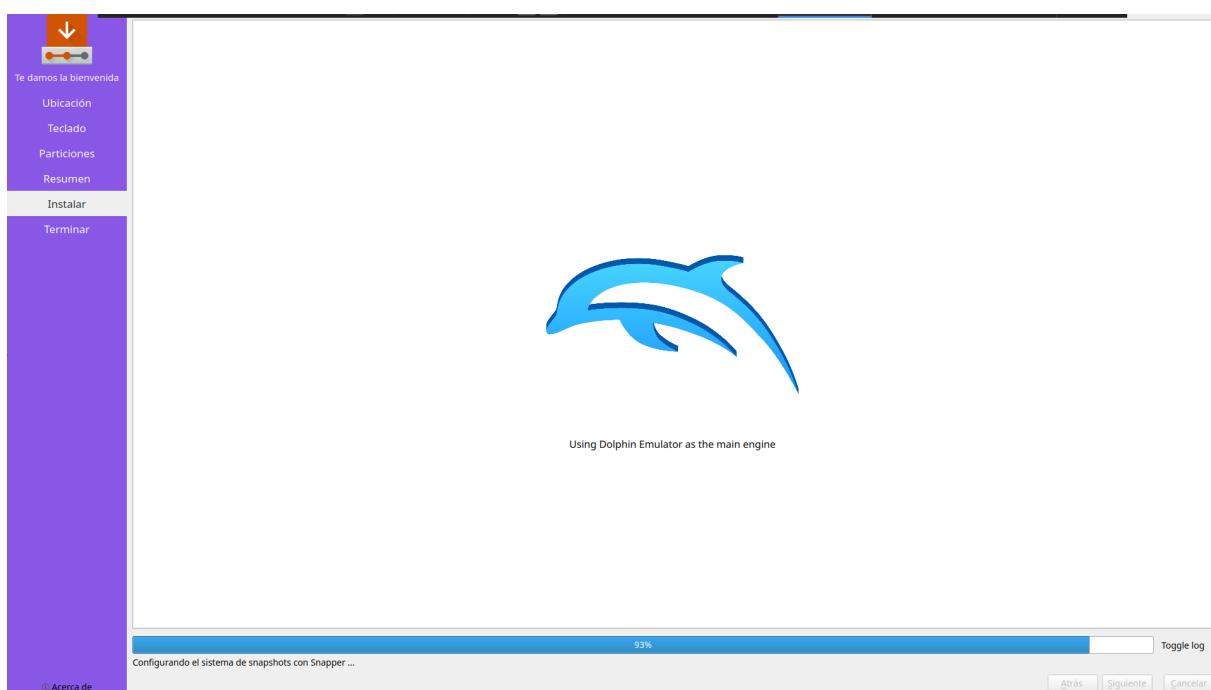


Figura 25: Proceso de instalación de DolphinOS

3.4 - Fase 4: Configuración del sistema DolphinOS

En esta fase se han ido estableciendo las configuraciones del sistema DolphinOS.

El desarrollo de las configuraciones se divide en dos partes. Por un lado se encuentran las configuraciones propias del sistema DolphinOS una vez instalado, y por otro, las configuraciones del emulador Dolphin, y del sistema Wii emulado.

Para poder probar que las configuraciones sean correctas, se han estado realizando instalaciones constantemente con las nuevas configuraciones, tanto en equipos físicos como en máquinas virtuales VMWare.

3.4.1 - Configuración de DolphinOS

En este apartado se realizará un desglose de las configuraciones aplicadas al sistema DolphinOS. Dichas configuraciones se encuentran alojadas en el repositorio de GitHub [DolphinOS-configs](#). Dado que DolphinOS ofrece dos versiones, una para gráficas Intel / AMD / VMWare y otra para gráficas NVIDIA, las configuraciones específicas de cada sistema se encuentran en la rama main y nvidia, correspondientemente.

De este repositorio se generan dos paquetes de Arch Linux, `dolphinos-configs` y `dolphinos-configs-nvidia`, los cuales son instalados por el instalador Calamares según la versión del sistema elegida.

En general, las configuraciones comunes a ambos sistemas son las siguientes:

- Instalación de un script que permitirá regenerar la configuración del gestor de arranque grub tras una instalación, actualización o eliminación de paquetes. Esto se usa en conjunción con Snapper para generar nuevas instantáneas del sistema, y poder así listarlas en el menú Snapshots de GRUB.
- Configuración tanto del cargador de arranque GRUB, como de Linux, SystemD, Bash y xinit para proporcionar un arranque silencioso [45]. Es decir, gracias a esta configuración, sólamente se mostrará una pantalla en negro desde que el usuario enciende el equipo hasta que carga el menú de Wii, ofreciendo una experiencia de usuario similar a un sistema Wii original.
- Configuración para hacer que el sistema reconozca y monte automáticamente una unidad de almacenamiento, estableciendo sus opciones de montaje. De esta forma, permite al usuario conectar una unidad de almacenamiento USB e instantáneamente poder cargar un juego desde la unidad USB a través de la aplicación web DolphinOS WebApp.
- En el directorio del usuario se definen una serie de configuraciones para poder arrancar el servidor gráfico X.Org tras realizarse el inicio de sesión automático. El servidor X.Org se inicia a través del programa xinit/startx. Xinit [46] permite sencillamente iniciar una aplicación o sesión de un gestor de ventanas sobre el servidor gráfico X.Org. De esta forma se inicia el gestor de ventanas OpenBox automáticamente en DolphinOS.

- En el fichero .bash_profile, previamente a la ejecución de startx para iniciar X.Org, se inicializa udiskie [47], un programa de automontaje de dispositivos de almacenamiento y se ejecuta la aplicación web DolphinOS WebApp.
- Para cargar el emulador Dolphin, haciendo que arranque directamente al menú del sistema tras iniciar el gestor de ventanas OpenBox, se agrega el comando de lanzamiento de Dolphin al fichero en la ruta .config/openbox/autostart. En este fichero se introducen adicionalmente una serie de parámetros para X.Org y se inicia el servidor de audio PipeWire.

En cuanto a las configuraciones específicas de la versión para gráficas NVIDIA:

- Creación de un hook [48] para pacman, el cual permitirá regenerar la imagen del initramfs cada vez que se realice una actualización de los kernels Linux instalados en el sistema. De esta forma los módulos del kernel del driver de NVIDIA se agregan a la imagen RAM inicial, pudiendo cargar los módulos en las etapas tempranas del arranque del sistema. Si los drivers del kernel de NVIDIA (nvidia-open) estuvieran integrados en el kernel Linux, este paso no sería necesario puesto que ya se cargarían desde el propio kernel.
- Configuración para equipos con gráficos duales, ya sean Intel - NVIDIA o AMD - NVIDIA. Este es el caso de la mayoría de portátiles que cuentan con tarjetas gráficas dedicadas de NVIDIA, adicionalmente de la tarjeta gráfica integrada en el procesador del equipo. Se configura el servidor X.Org para que en estos sistemas se utilice la tarjeta gráfica dedicada de NVIDIA como tarjeta gráfica principal para el renderizado de aplicaciones 3D, pudiendo disfrutar del rendimiento aumentado que ofrece la tarjeta gráfica dedicada.

Uno de los puntos fuertes de DolphinOS es que está optimizado para un entorno gaming. Por un lado, se hace uso del kernel customizado Linux-Zen, el cual incluye una serie de parches y modificaciones sobre Linux que ofrecen un mayor rendimiento del sistema. Por otro lado, cuando se lanza el emulador Dolphin se utiliza Gamemode [49] para optimizar el sistema durante su ejecución. Adicionalmente, durante la ejecución del emulador Dolphin, el usuario puede monitorizar el uso de los recursos y temperatura del hardware, la tasa de fotogramas y determinada información en tiempo real a través de la superposición en pantalla MangoHud [50].

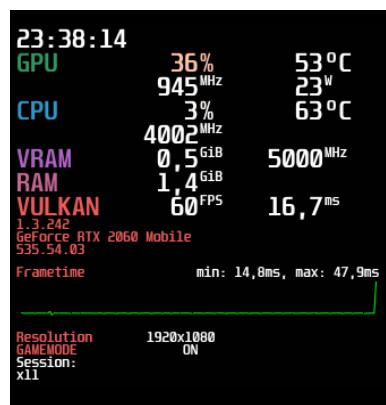


Figura 26: Superposición MangoHud

3.4.2 - Configuración del emulador Dolphin

Para el correcto funcionamiento y optimización del emulador Dolphin se han definido una serie de configuraciones.

Estas configuraciones se alojan en el repositorio DolphinOS-DolphinData, del cual se genera el paquete dolphinos-dolphin-data.

En cuanto a la configuración del propio emulador se establece lo siguiente:

- Para la API gráfica se utiliza Vulkan en todos los sistemas, menos en VMWare, que se hace uso de OpenGL puesto que VMWare carece de drivers paravirtualizados para Vulkan.
- Por defecto, se hace uso del modo PAL60, que permite ejecutar los juegos que no estén limitados internamente a 60 FPS (Fotogramas Por Segundo).
- El emulador utilizará una memoria SD virtual que se sincronizará con una carpeta.
- El cursor del ratón permanecerá invisible.
- El sistema emulado se ejecutará a pantalla completa.
- Dolphin determinará la resolución del sistema emulado en base a la resolución de pantalla en el momento del inicio del emulador.
- Se establece una anisotropía máxima de 2 y un antialias MSAA 2x.
- Se hace uso de los “Ubershaders híbridos” y de la precompilación de shaders para obtener el máximo rendimiento, evitando el efecto stuttering (parones en la ejecución mientras que se compilan los shaders).
- Se han agregado varios perfiles de controles al emulador para poder manejar el sistema emulado, por ejemplo, usando un mando de PS3.
- Se establecen una serie de atajos de teclado que permitirán al usuario cambiar entre los perfiles anteriormente mencionados, pudiendo así utilizar el mando que desee.

Por otro lado, en el sistema Wii emulado se ha agregado lo siguiente:

- **Canal Homebrew Channel** [51]: A través de este canal un usuario podrá ejecutar aplicaciones caseras (homebrews), como emuladores, utilidades, aplicaciones, etc). Se incluye además la aplicación Homebrew Browser, la cual permitirá que un usuario pueda descargar e instalar homebrews desde la propia aplicación.
- **Servicios de RiiConnect24**: RiiConnect24 [14] es una reimplementación de los servicios online de Nintendo (WiiConnect24), como el “Canal Tiempo” o el “Canal Nintendo”.



Figura 27: Canal Homebrew Channel

3.5 - Fase 5: Desarrollo de la aplicación DolphinOS WebApp

DolphinOS WebApp es una aplicación web diseñada para poder proporcionar a los usuarios de DolphinOS una interfaz gráfica, en la cual puedan gestionar tanto el emulador como el sistema Linux base desde dentro del sistema emulado.

DolphinOS WebApp se ha inspirado en la aplicación casera webMAN-MOD [52] de PS3, pero enfocándose en el caso de uso de esta aplicación.

A lo largo de esta fase se ha ido desarrollando la aplicación web, añadiendo determinadas características.

La aplicación web se ha desarrollado usando el lenguaje de programación Python y el framework web Python Flask. La estructura de ficheros es la siguiente:

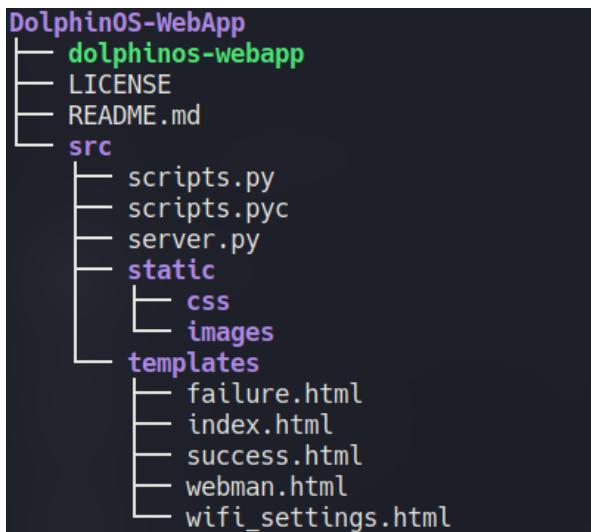


Figura 28: Esquema de ficheros de DolphinOS WebApp

El archivo `dolphinos-webapp` es un script que se encarga de ejecutar la aplicación web. Su objetivo es proporcionar un ejecutable, el cual se ubica en el directorio `/usr/bin` del sistema instalado. Al ubicarse dentro de un directorio que está contenido en la variable del entorno `$PATH`, se podrá ejecutar simplemente escribiendo “`dolphinos-webapp`” en una terminal sin necesidad de indicar la ruta completa.

Por lo que respecta a la aplicación web, el fichero principal es el módulo `server.py`.

Al ejecutar el programa, se inicia un servidor web de Python Flask. Por defecto, se abre en la dirección del host local `http://127.0.0.1` utilizando el puerto 5000.

Para definir cada uno de los sitios que ofrece el servidor web se hace uso de la función `route()` de Flask. A continuación de la llamada a la función `route()` para crear una nueva ruta del sitio web, se define una función de Python. En dicha función, se establecen las acciones que debe realizar el servidor, como enviar una página web HTML al cliente o realizar una acción en el propio servidor.

A continuación se muestra un mapa del sitio web y la página principal de la aplicación:

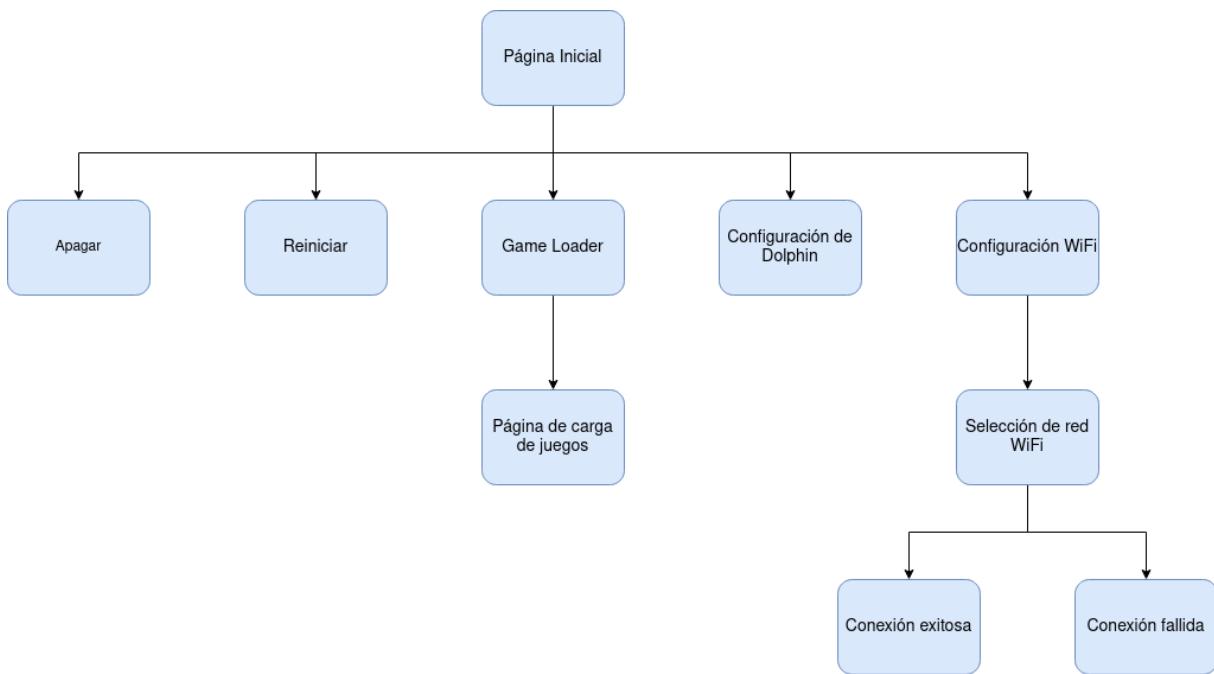


Figura 29: Mapa del sitio web DolphinOS WebApp

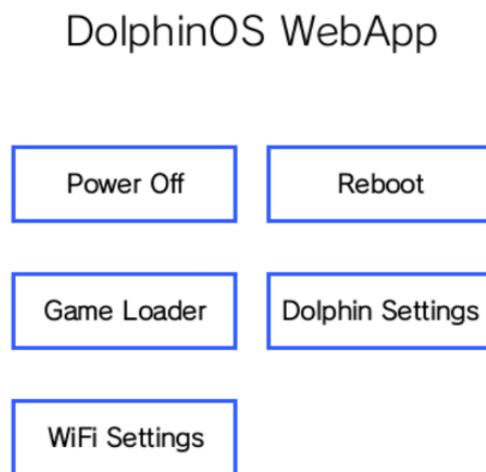


Figura 30: Aplicación web DolphinOS WebApp

Si un usuario selecciona los botones “Power Off” o “Reboot”, el sistema se apagará o reiniciará, correspondientemente. Antes de apagar el sistema, se detiene primeramente el emulador Dolphin de forma segura, evitando que se corrompan los datos.

Por otro lado, si selecciona la opción Dolphin Settings, se detendrá de forma segura el emulador, y se abrirá una ventana de Dolphin. Aquí el usuario podrá gestionar la configuración del emulador, como ajustes generales, ajustes de gráficos o ajustes de mandos. Sin embargo, no se recomienda utilizar esta ventana para cargar el menú del sistema o cargar un juego, puesto que no se está haciendo uso de todas las optimizaciones del sistema. Esta ventana es solo para configuración.

Tras finalizar la configuración el usuario deberá cerrar la ventana e instantáneamente se volverá a abrir el emulador en el menú del sistema Wii.

DolphinOS WebApp presenta al usuario una interfaz, denominada “Game Loader”, donde podrá cargar sus juegos en el emulador desde una memoria flash USB, y posteriormente ejecutarlos desde el menú de Wii.

DolphinOS soporta los formatos FAT32, exFAT, NTFS y BTRFS.

Para poder cargar los juegos desde un USB, el usuario deberá crear previamente desde otro ordenador una carpeta denominada “Games” en la raíz de la memoria, y colocar los juegos en dicho directorio. Se aceptan los formatos de juego .iso, .wbfs y .rvz.

A continuación el usuario deberá conectar la memoria USB al sistema y esperar unos segundos a que se monte. Posteriormente deberá seleccionar “Game Loader” y se mostrarán los juegos que se encuentran actualmente en la memoria USB.

Si el usuario selecciona un juego, se envía al servidor el nombre del juego seleccionado, y en base la ruta del directorio Games de la memoria USB y el nombre de archivo, se obtiene la ruta completa del archivo del juego. Posteriormente, se cambia el valor de la variable “DefaultISO” a la ruta anteriormente obtenida, en el fichero de configuración de Dolphin.

Tras establecerse la nueva imagen de juego por defecto, se reinicia el emulador.

Al regresar al menú de Wii, el juego aparecerá en el “Canal Disco” desde donde se podrá ejecutar.

Si no se detectase ninguna unidad USB o la carpeta “Games” no existiese, se enviaría una alerta de JavaScript al usuario informando del incidente.

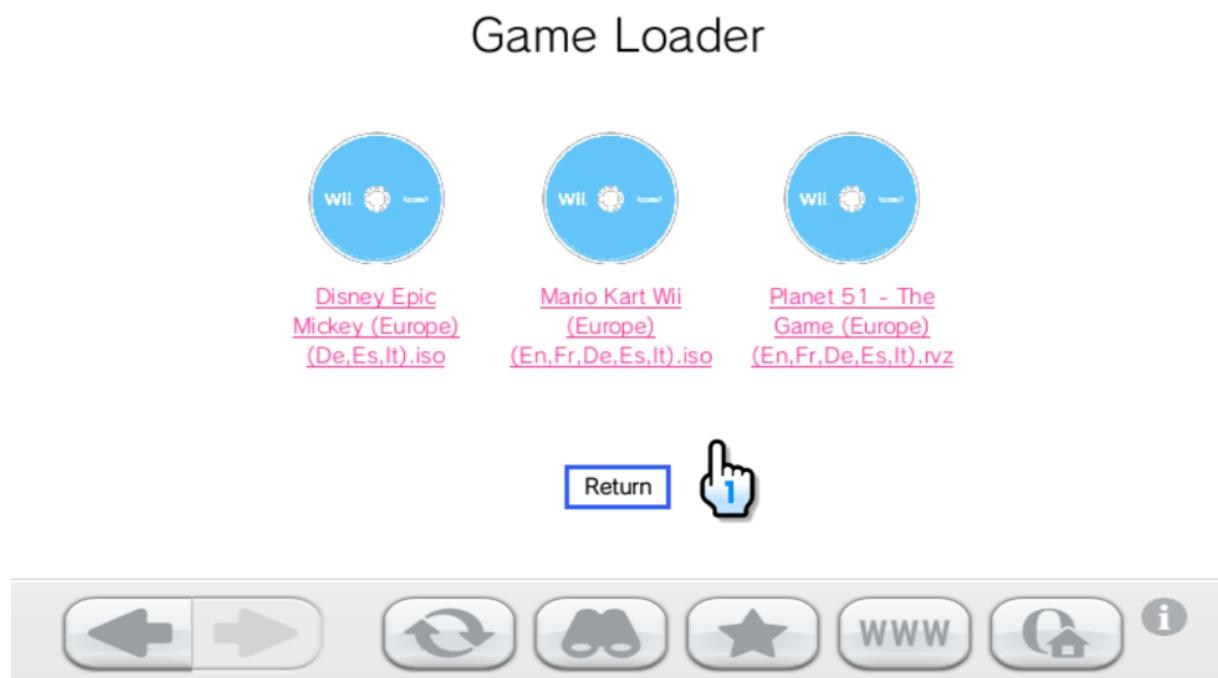


Figura 31: Interfaz “Game Loader” de DolphinOS WebApp

Finalmente, DolphinOS WebApp ofrece un gestor de conexiones WiFi. En la sección “WiFi Settings”, si el equipo del usuario dispone de tarjeta inalámbrica, se listarán las redes WiFi detectadas en el entorno. El usuario deberá seleccionar una red de la lista e introducir la contraseña de la red. Al presionar el botón connect, el sistema tratará de conectarse a la red mediante los datos proporcionados. Si la conexión se realiza con éxito, se redirige a una página en la que se muestra el nombre de la red actualmente conectada y la dirección IPv4 privada del dispositivo. Si por algún motivo la conexión fallase, ya sea por contraseña incorrecta o por fallo en la conectividad, se redirigiría al usuario a otra página informándole del error.

WiFi Network Settings



Figura 32: Interfaz de conexión a red WiFi

Connection successful

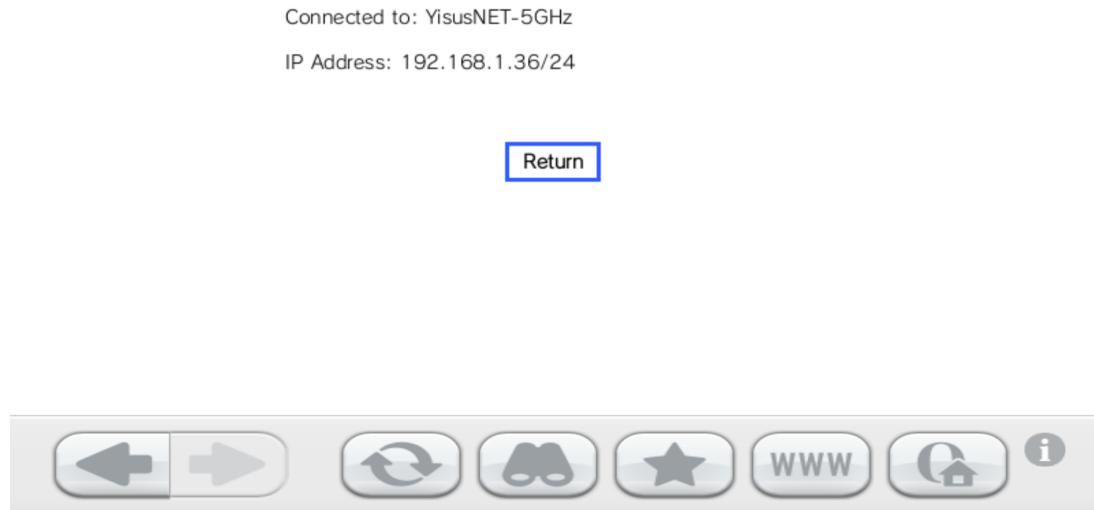


Figura 33: Informe de conexión exitosa

4 - Conclusiones y trabajo futuro

4.1 - Conclusiones

Gracias a DolphinOS, un usuario puede disfrutar de la misma experiencia que ofrecería una consola Wii original, sin depender de ella. Llegado a un punto en un futuro, el hardware de las consolas alcanzará su tiempo de vida útil máximo, por lo que sistemas operativos como DolphinOS serán la única forma de preservar el software y la experiencia de dichas consolas.

A través del desarrollo de DolphinOS he aprendido tanto a crear mi propia distribución basada en Arch Linux, como a la configuración de sistemas instalables, y a diseñar una pequeña aplicación web usando Python Flask.

Además, este proyecto me ha servido para reforzar y ampliar mi conocimiento en sistemas Linux y en cómo trabajar con GitHub y los servicios que ofrece.

Para finalizar, decir que, personalmente, me haría muchísima ilusión que en un futuro no muy lejano este proyecto comience a tomar tracción y pueda crear una gran comunidad detrás, que ayude a mantener el proyecto y a proponer nuevas ideas y soluciones.

4.2 - Trabajo futuro

El trabajo realizado durante este proyecto es sólamente la punta del iceberg.

DolphinOS es un proyecto con un gran potencial si consigue tomar tracción, sobre todo si consigue aunar a una comunidad de desarrolladores y usuarios.

Para un trabajo futuro, a corto y medio plazo, se establecen los siguientes objetivos:

- Implementar un método que permita realizar actualizaciones del sistema de forma segura a través de la aplicación web DolphinOS WebApp.
- Añadir más configuraciones del sistema a DolphinOS WebApp, como puede ser la gestión de pantallas, configuración de sonido, ajustes de red más avanzados, gestión de energía, etc.
- Incorporar en DolphinOS WebApp los ajustes esenciales del emulador Dolphin (resolución, ciertos ajustes gráficos, configuración de mandos, etc.), sin la necesidad de tener que salir de la aplicación web para establecer dichos ajustes.
- Dar la posibilidad al usuario de copiar los juegos de la unidad USB al almacenamiento interno del sistema, ofreciendo un sistema para evitar que se sature el espacio del almacenamiento interno.
- Separar el firmware del sistema de DolphinOS para evitar posibles problemas que se puedan producir con Nintendo. En este caso, un usuario deberá proporcionar, durante la instalación de DolphinOS, un archivo comprimido con el firmware del sistema, ya sea obtenido de una videoconsola Wii real o desde otra instancia de Dolphin en otro sistema.

En cuanto a los objetivos a largo plazo:

- Migrar el proyecto al sistema de ventanas Wayland y así poder abandonar el antiguo sistema X11. Usando el compositor Gamescope, se podrá renderizar el emulador Dolphin, con las ventajas que ofrece Gamescope para entornos gaming.
Sin embargo, esto no llegará a ser posible hasta que exista una solución viable a los problemas de sincronización de los drivers propietarios de NVIDIA con XWayland, o hasta que tanto Gamescope soporte clientes Wayland y Dolphin sea portado a Wayland..
- Servir de inspiración proyectos similares. Tomando la base del funcionamiento de DolphinOS, se podría realizar un proyecto similar, pero usando otros emuladores, como podrían ser RPCS3, JPCSP, CEMU, etc.

5 - Enlaces de interés

El desarrollo de DolphinOS se ha llevado a cabo bajo una infraestructura de repositorios de control de versiones de código fuente, alojados en la plataforma GitHub.

El enlace a la organización de desarrollo donde se pueden encontrar los repositorios se adjunta a continuación:

- <https://github.com/DolphinOS-Development>

A fecha de elaboración de la memoria, el repositorio de la distribución se aloja en el siguiente repositorio:

- <https://github.com/JesusXD88/dolphinos-repo>

Si se desea probar DolphinOS, se pueden descargar las imágenes ISO de instalación desde la página del proyecto en SourceForge:

- <https://sourceforge.net/projects/dolphinos/>

6 - Guía de instalación y uso

Para la instalación de DolphinOS es necesario descargarse la imagen ISO de instalación desde la página del proyecto en SourceForge [59].

En cuanto a los requisitos técnicos se establecen los mismos requisitos del emulador Dolphin [60]. En el caso de instalarlo en una máquina virtual VMWare, se recomienda establecer mínimo 4 GB de RAM, y 1 GB de memoria de vídeo. La memoria de vídeo virtual forma parte de la memoria RAM asignada al sistema, por tanto, quedarían 3 GB disponibles para DolphinOS.

En DolphinOS aparecen dos carpetas, correspondientes a las dos versiones de DolphinOS:

- **DolphinOS:** Tarjetas gráficas de Intel, AMD y máquinas virtuales VMWare.
- **DolphinOS-NVIDIA:** Tarjetas gráficas de NVIDIA. También es recomendable para los equipos con gráficas duales, ya sea Intel-NVIDIA o AMD-NVIDIA, que deseen utilizar Dolphin con la tarjeta gráfica dedicada (normalmente son la mayoría de portátiles gaming).

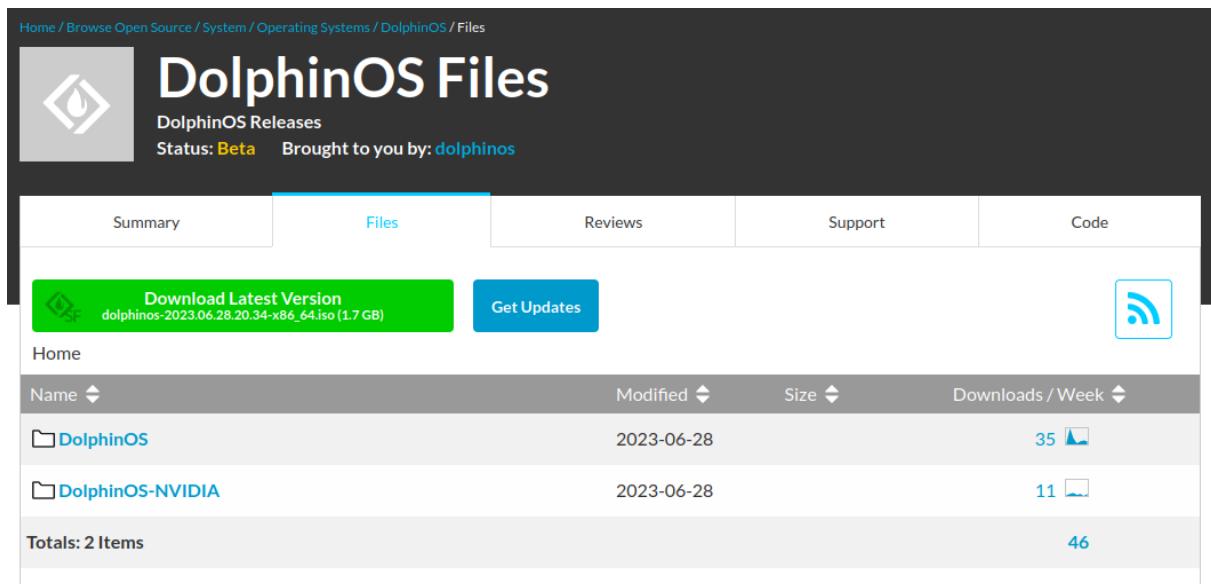


Figura 34: Portal de descarga de DolphinOS

Tras descargar la imagen ISO deseada, se procede a grabar la imagen en un DVD o una memoria USB. En el caso de usar una memoria USB, se recomienda usar el software de creación de USBs arrancables Balena Etcher [61], o si se desea utilizar una unidad USB con varias imágenes ISO de instalación de diferentes sistemas, se recomienda la herramienta Ventoy [62].

Una vez grabada la ISO en un DVD o en una memoria USB, se reinicia el ordenador y se accede al cargador de arranque del sistema UEFI. Se selecciona el medio de instalación de DolphinOS y se arranca el sistema.

Cuando inicie el sistema, aparecerá automáticamente el software de instalación Calamares:



Figura 35: Interfaz de bienvenida de Calamares

Se selecciona el idioma, y en la siguiente interfaz se establece la región y huso horario:

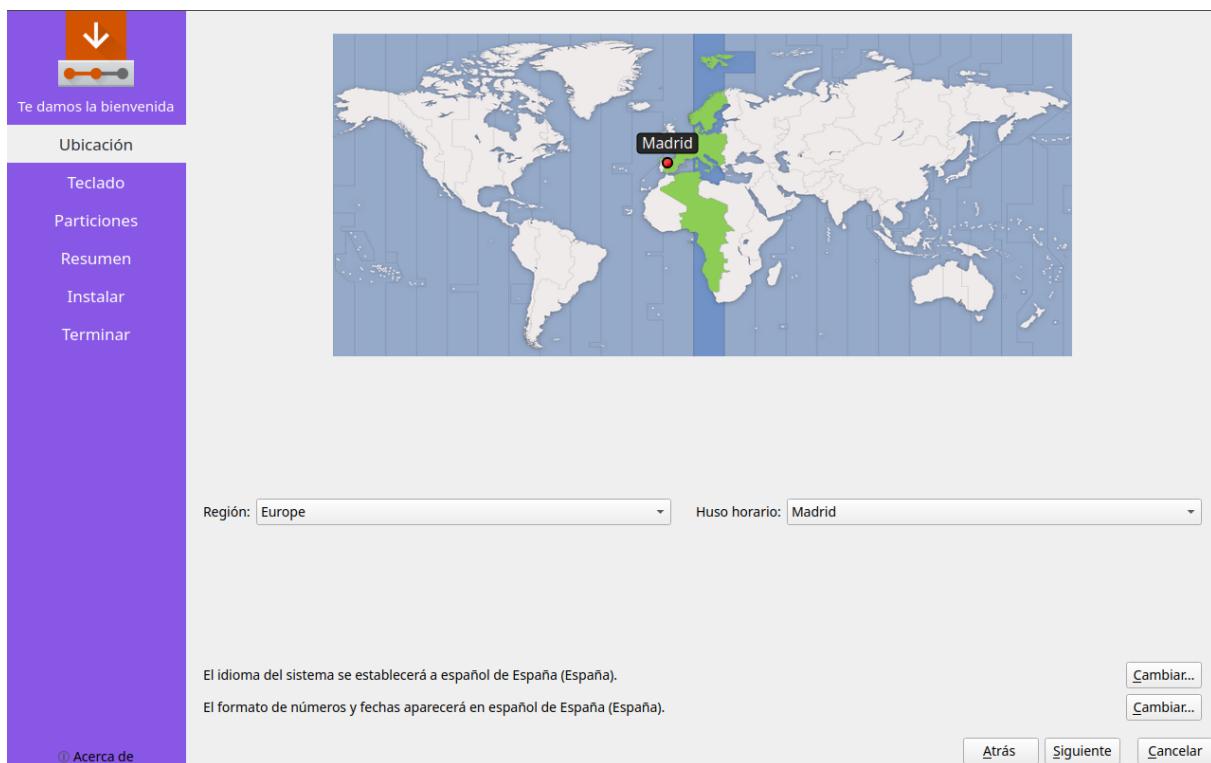


Figura 36: Selección de región y huso horario

Posteriormente se elige la distribución del teclado en base al teclado que se vaya a usar:

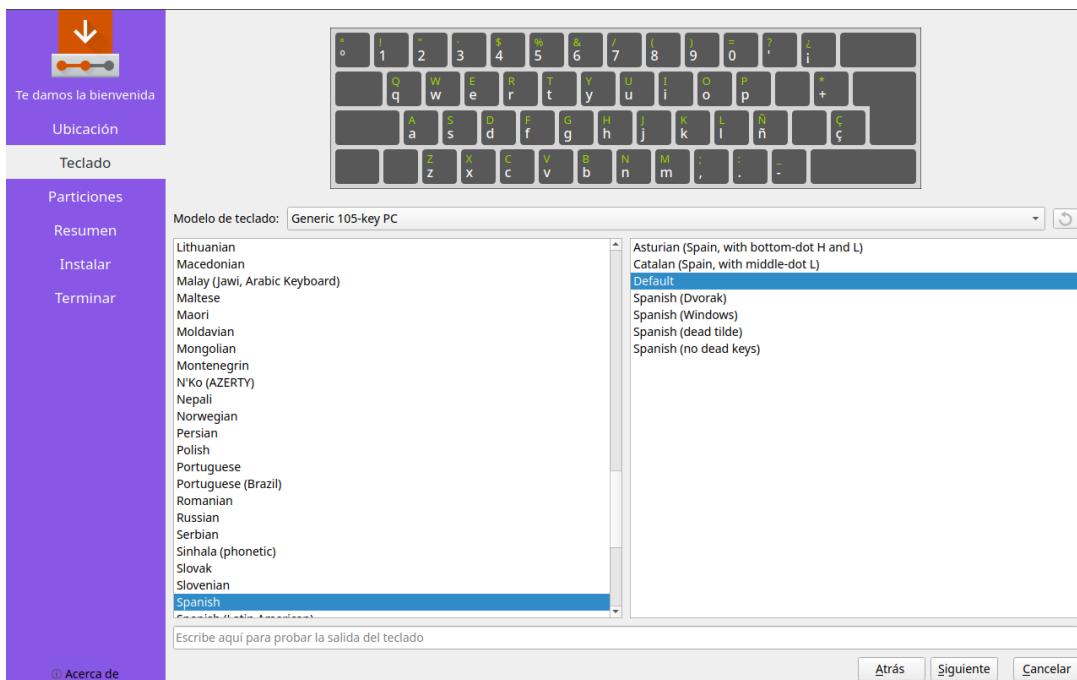


Figura 37: Selección de la distribución del teclado

A continuación se muestra la herramienta de particionado de disco. En la parte superior de la pantalla se puede seleccionar el dispositivo de almacenamiento objetivo.

Se proporcionan tres métodos de particionado: Instalar al lado, reemplazar una partición o borrar el disco completo y crear las particiones de DolphinOS desde cero.

Se selecciona la opción más adecuada para cada caso de uso.

La opción instalar al lado permite redimensionar una partición y poder instalar en el nuevo espacio el sistema DolphinOS. Se requieren 10 GB de espacio libre en disco como mínimo.

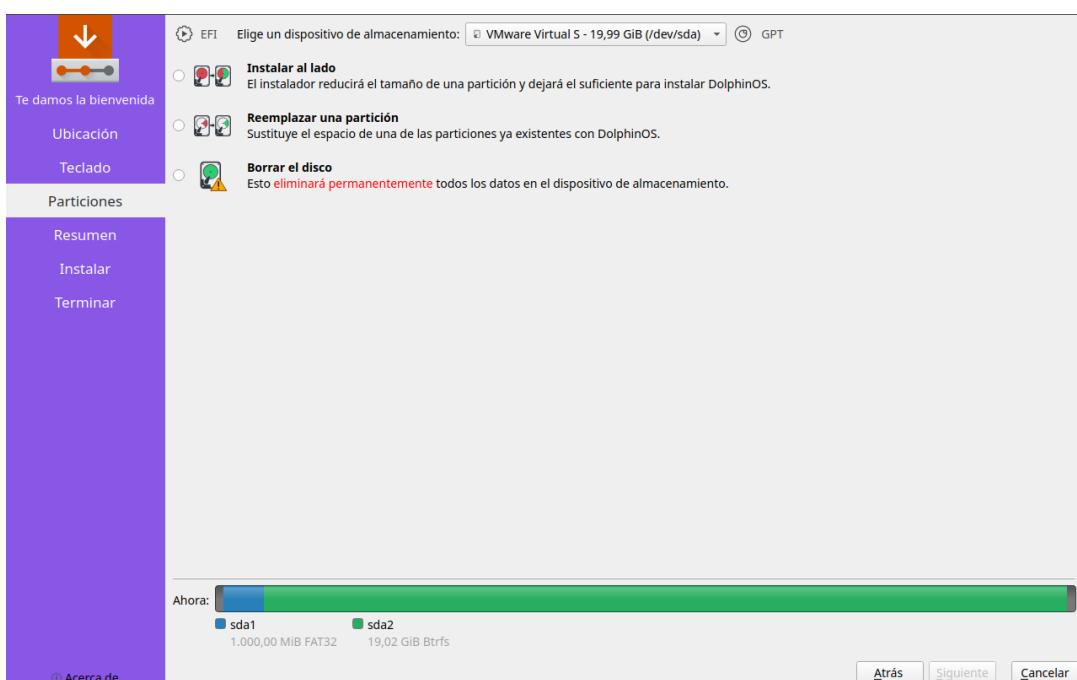


Figura 38: Particionado de disco con Calamares

Finalmente, se muestra un resumen de las acciones a realizar en el sistema para la instalación y si todo es correcto, se presiona el botón “Instalar”.

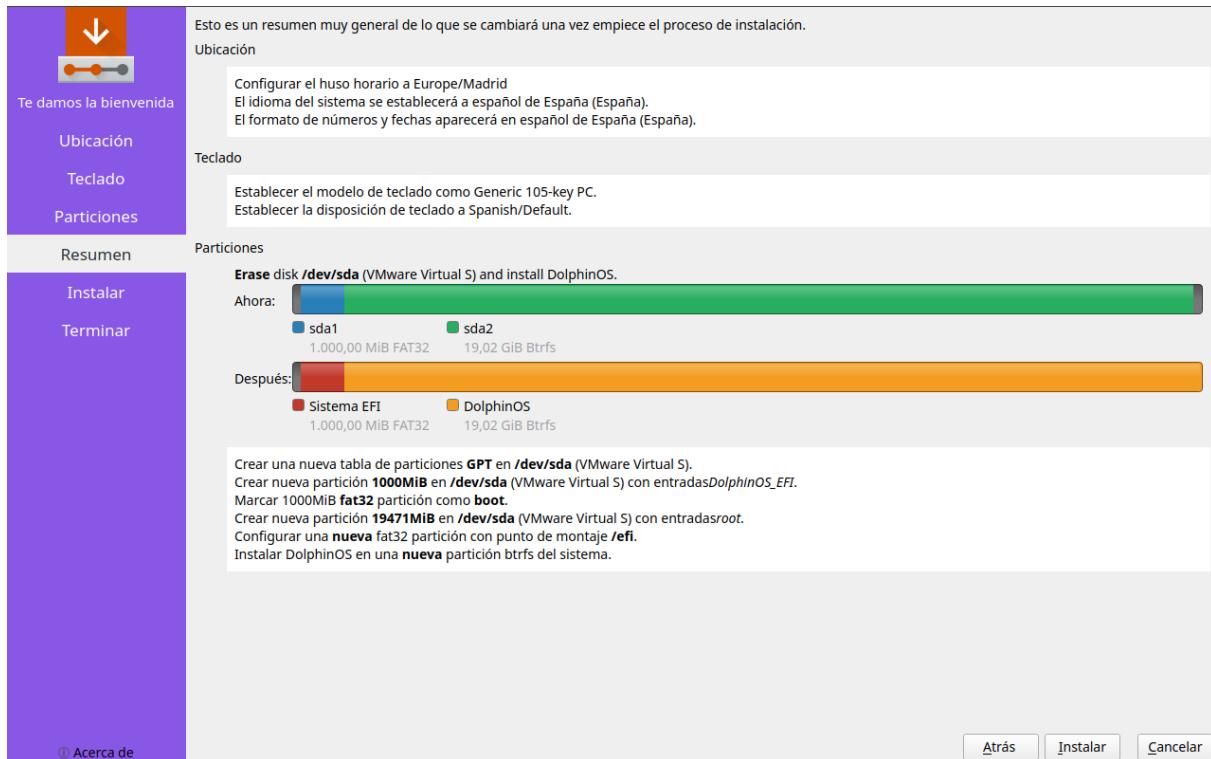


Figura 39: Interfaz de resumen previo a la instalación de DolphinOS

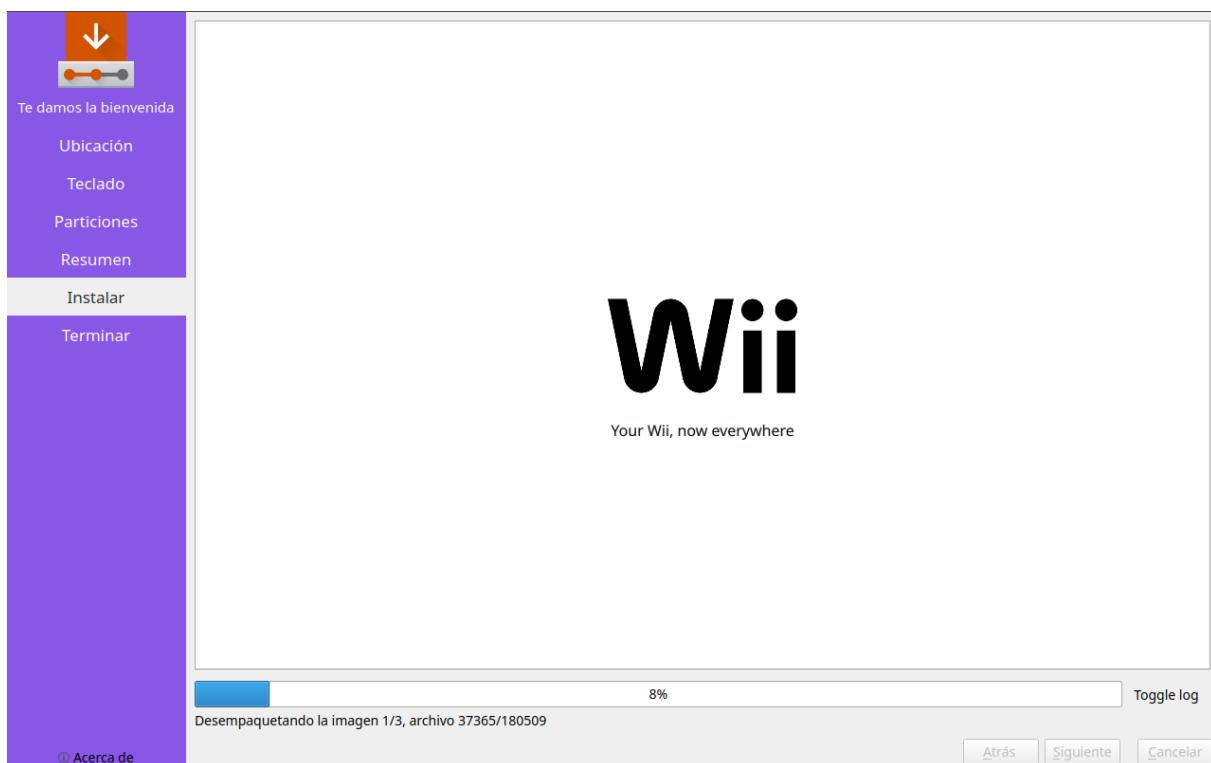


Figura 40: Instalación de DolphinOS

Tras instalar el sistema, se procede a reiniciar el equipo, y automáticamente el ordenador arrancará desde el gestor de arranque GRUB de DolphinOS.

Al finalizar el proceso de arranque, se mostrará una interfaz de configuración inicial de Wii. Por defecto, para controlar el sistema Wii, se hace uso de ratón y teclado, aunque si se conecta un mando DualShock 3 o 8BitDo, y se presiona la combinación de teclas “Control” y “+”, se puede hacer uso de estos cualquiera de estos mandos.

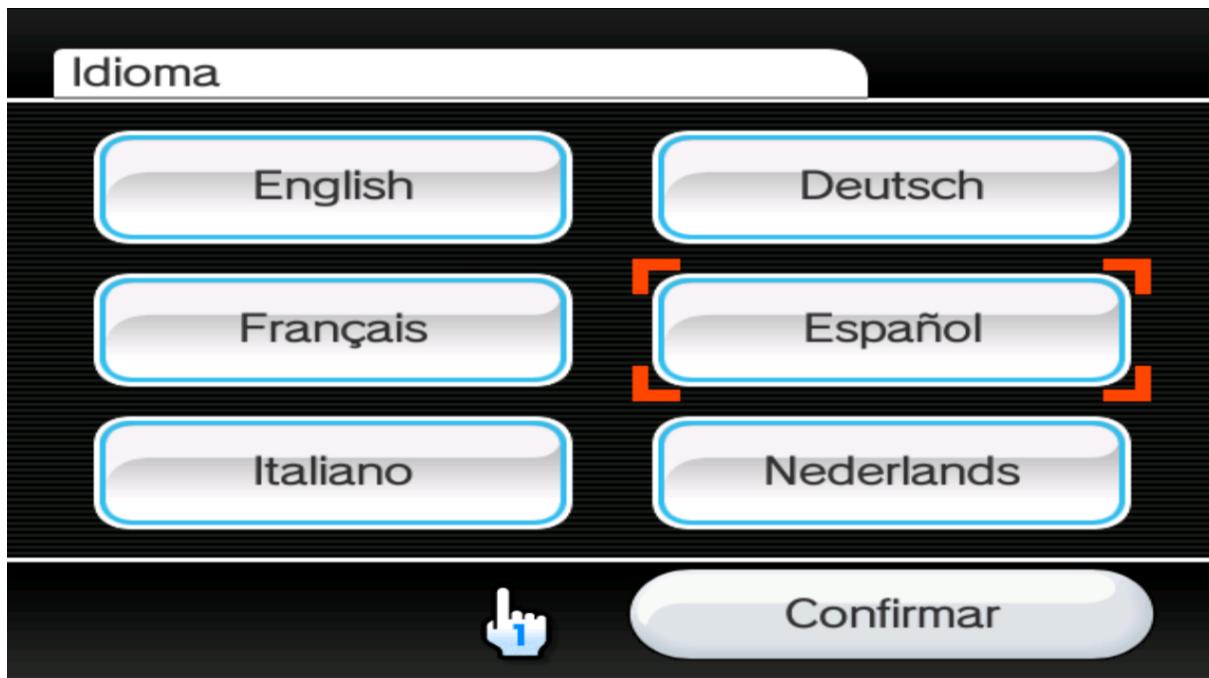


Figura 41: Configuración inicial de Wii

Una vez configurado el sistema, se accede automáticamente al menú de Wii. Ya está el sistema listo para usarse.



Figura 42: Menú del sistema Wii

Sin embargo, se recomienda reiniciar el sistema para que se termine de configurar internamente. Para ello, se accede al “Canal Internet” y se inicia el navegador web.



Figura 43: Canal Internet

Una vez dentro, en la pantalla de inicio, se selecciona el botón “Favoritos”. En la lista de favoritos aparecerá un elemento llamado DolphinOS. Se selecciona para acceder a la aplicación web DolphinOS WebApp.



Figura 44: Marcador en favoritos de DolphinOS WebApp

Finalmente, dentro de la aplicación web DolphinOS WebApp, se selecciona la opción “Reboot” para poder reiniciar el sistema seguramente:



Figura 45: Reinicio del sistema desde DolphinOS WebApp

Tras el reinicio, ya se puede hacer uso de DolphinOS de forma similar a una Wii. Si se desea abrir la superposición en tiempo real MangoHud, se puede realizar a través de la combinación de teclas “Shift derecho” y “F12”.



Figura 46: Videojuego Planet 51 con la superposición MangoHud

Bibliografía

A continuación, se adjuntan las referencias bibliográficas que se han hecho uso durante el desarrollo de esta memoria:

[1] Wiki de Arch Linux, [consulta 29-06-2023], disponible en:
<https://wiki.archlinux.org/>

[2] Linux Distros That Turn Your PC into Retro Gaming Console, [consulta 26-06-2023], disponible en:
<https://itsfoss.com/retro-gaming-console-linux-distros/>

[3] Batocera Linux , [consulta 26-06-2023], disponible en:
<https://batocera.org/>

[4] Lakka, [consulta 26-06-2023], disponible en:
<https://www.lakka.tv/>

[5] Retropie, [consulta 26-06-2023], disponible en:
<https://retropie.org.uk/>

[6] Raspbian, [consulta 26-06-2023], disponible en:
<https://www.raspberrypi.com/software/>

[7] SteamOS, [consulta 26-06-2023], disponible en:
<https://en.wikipedia.org/wiki/SteamOS>

[8] Flatpak, [consulta 26-06-2023], disponible en:
<https://www.flatpak.org/>

[9] RPCS3, [consulta 26-06-2023], disponible en:
<https://rpcs3.net/>

[10] JPCSP, [consulta 26-06-2023], disponible en:
<https://github.com/jpcsp/jpcsp>

[11] PlayStation Portable emulators, [consulta 26-06-2023], disponible en:
https://emulation.gametechwiki.com/index.php/PlayStation_Portable_emulators

[12] Wii emulators, [consulta 26-06-2023], disponible en:
https://emulation.gametechwiki.com/index.php/Wii_emulators

[13] Dolphin Emulator, [consulta 26-06-2023], disponible en:
<https://dolphin-emu.org/>

[14] RiiConnect24, [consulta 26-06-2023], disponible en:
<https://rc24.xyz/>

[15] Wiimmfi, [consulta 26-06-2023], disponible en:
<https://wiimmfi.de/>

[16] Bash, [consulta 27-06-2023], disponible en:
<https://es.wikipedia.org/wiki/Bash>

[17] Python Flask, [consulta 27-06-2023], disponible en:
<https://flask.palletsprojects.com>

[18] HTML, [consulta 27-06-2023], disponible en:
<https://es.wikipedia.org/wiki/HTML>

[19] CSS, [consulta 27-06-2023], disponible en:
<https://es.wikipedia.org/wiki/CSS>

[20] JavaScript, [consulta 27-06-2023], disponible en:
<https://es.wikipedia.org/wiki/JavaScript>

[21] Navegador Web Opera, [consulta 27-06-2023], disponible en:
[https://es.wikipedia.org/wiki/Opera_\(navegador\)#Cambio_a_licencia_freeware](https://es.wikipedia.org/wiki/Opera_(navegador)#Cambio_a_licencia_freeware)

[22] Visual Studio Code, [consulta 27-06-2023], disponible en:
<https://code.visualstudio.com/>

[23] Git, [consulta 27-06-2023], disponible en:
<https://git-scm.com/>

[24] VMWare, [consulta 27-06-2023], disponible en:
<https://www.vmware.com/>

[25] GitHub, [consulta 27-06-2023], disponible en:
<https://en.wikipedia.org/wiki/GitHub>

[26] Organizaciones de GitHub, [consulta 27-06-2023], disponible en:
<https://docs.github.com/es/organizations/collaborating-with-groups-in-organizations/about-organizations>

[27] GitHub Actions, [consulta 27-06-2023], disponible en:
<https://docs.github.com/en/actions>

[28] GitHub Pages, [consulta 27-06-2023], disponible en:
<https://pages.github.com/>

[29] SourceForge, [consulta 27-06-2023], disponible en:
<https://sourceforge.net/>

[30] Universal Blue, [consulta 28-06-2023], disponible en:
<https://universal-blue.org/>

[31] Fedora Silverblue, [consulta 28-06-2023], disponible en:
<https://fedoraproject.org/es/silverblue/>

[32] OSTree, [consulta 28-06-2023], disponible en:
<https://ostreedev.github.io/ostree/>

[33] Arch Linux, [consulta 28-06-2023], disponible en:
<https://archlinux.org/>

[34] X Window System, [consulta 28-06-2023], disponible en:
https://en.wikipedia.org/wiki/X_Window_System

[35] Wayland, [consulta 28-06-2023], disponible en:
[https://en.wikipedia.org/wiki/Wayland_\(protocol\)](https://en.wikipedia.org/wiki/Wayland_(protocol))

[36] XWayland, [consulta 28-06-2023], disponible en:
<https://man.archlinux.org/man/extra/xorg-xwayland/Xwayland.1.en>

[37] XWayland glamor renders incorrectly on nvidia, [consulta 28-06-2023], disponible en:
<https://gitlab.freedesktop.org/xorg/xserver/-/issues/1317>

[38] Sistema de archivos BTRFS, [consulta 28-06-2023], disponible en:
<https://en.wikipedia.org/wiki/Btrfs>

[39] BTRFS Subvolumes, [consulta 28-06-2023], disponible en:
<https://wiki.archlinux.org/title/Btrfs#Subvolumes>

[40] BTRFS Snapshots, [consulta 28-06-2023], disponible en:
<https://wiki.archlinux.org/title/Btrfs#Snapshots>

[41] Snapper, [consulta 28-06-2023], disponible en:
<https://wiki.archlinux.org/title/Snapper>

[42] Calamares, [consulta 28-06-2023], disponible en:
<https://github.com/calamares/calamares>

[43] Creación de un repositorio de Arch Linux, [consulta 28-06-2023], disponible en:
https://wiki.archlinux.org/title/Pacman/Tips_and_tricks#Custom_local_repository

[44] Archiso, [consulta 29-06-2023], disponible en:
<https://wiki.archlinux.org/title/Archiso>

[45] Arranque silencioso en Arch Linux, [consulta 29-06-2023], disponible en:
https://wiki.archlinux.org/title/Silent_boot

[46] Xinit, [consulta 29-06-2023], disponible en:
<https://wiki.archlinux.org/title/Xinit>

[47] Udiskie, [consulta 29-06-2023], disponible en:
<https://github.com/coldfix/udiskie>

[48] Pacman Hooks, [consulta 29-06-2023], disponible en:
<https://wiki.archlinux.org/title/Pacman#Hooks>

[49] Feral Gamemode, [consulta 29-06-2023], disponible en:
<https://wiki.archlinux.org/title/Gamemode>

[50] MangoHud, [consulta 29-06-2023], disponible en:
<https://github.com/flightlessmango/MangoHud>

[51] Homebrew Channel, [consulta 29-06-2023], disponible en:
<https://github.com/failOverflow/hbc>

[52] webMAN-MOD, [consulta 29-06-2023], disponible en:
<https://github.com/aldostools/webMAN-MOD>

[53] Buildroot Linux, [consulta 26-06-2023], disponible en:
<https://buildroot.org/>

[54] Python, [consulta 27-06-2023], disponible en:
[https://en.wikipedia.org/wiki/Python_\(programming_language\)](https://en.wikipedia.org/wiki/Python_(programming_language))

[55] My custom cabinet build, [consulta 27-06-2023], disponible en:
<https://i.redd.it/zz7huol9a4g41.jpg>

[56] Lakka Nintendo Switch, [consulta 26-06-2023], disponible en:
<https://hackinformer.com/2018/08/26/using-lakka-on-the-nintendo-switch-has-never-been-easier/>

[57] Emulators on Steam Deck, [consulta 26-06-2023], disponible en:
<https://www.trendradars.com/latest/article-863857-how-to-run-emulators-on-the-steam-deck/>

[58] DolphinOS-Development, [consulta 30-06-2023], disponible en:
<https://github.com/DolphinOS-Development>

[59] Descarga de DolphinOS en SourceForge, [consulta 30-06-2023], disponible en:
<https://sourceforge.net/projects/dolphinos/>

[60] Requisitos técnicos de Dolphin, [consulta 29-06-2023], disponible en:
<https://es.dolphin-emu.org/docs/faq/>

[61] Balena Etcher, [consulta 29-06-2023], disponible en:
<https://etcher.balena.io/>

[62] Ventoy, [consulta 29-06-2023], disponible en:
<https://ventoy.net/en/index.html>



Resumen/Abstract

En el apasionante mundo de los videojuegos, las videoconsolas juegan un papel crucial ya que proporcionan un medio dedicado y medianamente asequible para poder disfrutar de los videojuegos. Como en la mayoría de ámbitos tecnológicos, todo está en continuo cambio, por lo que con el paso del tiempo las consolas se van quedando obsoletas tanto por obsolescencia programada o por meramente llegar a su vida máxima.

Sin embargo, a través de la emulación, es posible averiguar o simular cómo funcionan internamente estas consolas, y poder hacer compatible su software, específicamente videojuegos, en otros dispositivos hardware, con incluso, arquitecturas diferentes. Ya sea en un ordenador, un móvil, tablet, o incluso consolas más actuales, se podrá seguir disfrutando de los videojuegos de esas consolas más antiguas.

No obstante, existe un aspecto que la mayoría de emuladores actuales no consiguen lograr completamente, ya sea por ser muy complejo o por no hallarse en los planes de desarrollo de los mismos. Es el caso del firmware del sistema o sistema operativo, y algunas de las aplicaciones del sistema.

Esto suele ser un elemento de alta complejidad a la hora de ser emulado porque se necesita una emulación más a bajo nivel, llegando a emular partes del hardware de la consola, para que el firmware pueda ejecutarse como si fuese hardware nativo.

Por suerte, uno de los emuladores que permiten cargar el firmware del sistema de forma sencilla es Dolphin, un emulador de Wii, GameCube y Gameboy Advance.

A través de Dolphin se puede cargar la aplicación “Menú del Sistema”, la cual proporciona el propio menú principal del sistema Wii junto con todas las aplicaciones preinstaladas. De esta forma, se le proporciona al usuario la misma experiencia que ofrecería la videoconsola, pero en cualquier hardware en el cual se pueda ejecutar este emulador.

Por ello, se ha desarrollado un sistema instalable, basado en una distribución GNU/Linux, el cual permite disfrutar de la misma experiencia que ofrecería una videoconsola Wii, en cualquier ordenador con arquitectura x86_64.

A través de un instalador gráfico, un usuario podrá instalar el sistema fácilmente en su ordenador, pudiendo instalarlo conjuntamente con su sistema operativo actual si así lo desea, manteniendo todos sus datos intactos. Tras realizar la instalación, el sistema está configurado automáticamente para arrancar exclusivamente el emulador Dolphin, cargando el diálogo de configuración inicial de Wii. En los sucesivos inicios, el sistema cargará automáticamente al menú de Wii. Además, mediante una herramienta web que el usuario podrá acceder a través del navegador web, el usuario podrá gestionar el sistema Linux base, cargar juegos almacenados en una memoria USB o abrir la configuración del emulador.

In the thrilling world of video games, video game consoles play an important role given that they provide a dedicated and affordable medium for allowing you to play video games.

As in almost every IT environment, everything is changing constantly, and this affects video game consoles as well, as they become deprecated either due to planned obsolescence or simply by reaching its maximum lifespan.

However, it is possible to find out or simulate how these video game consoles work internally through emulation, as well as making their software (specially video games) compatible with other devices that could even be running on different architectures. Through emulation it's possible to keep playing and enjoying older consoles games on a modern computer, phone, tablet or even on more recent consoles.

Nevertheless, there is an aspect that most current emulators don't manage to develop. This is the case of video games consoles' operating systems or system firmware.

The reasons may vary, but it is clear that this is a very complex thing to carry out, given that a lower level emulation is needed. Sometimes some hardware of the actual console needs to be emulated in order to run the system firmware.

Luckily, one of the few emulators that allows the system firmware to boot easily is Dolphin, a Wii, GameCube and GameBoy Advance emulator.

Through Dolphin it's possible to boot the "System Menu" application, that provides the actual Wii menu, including all the preinstalled applications. This way, the user is provided with the same experience that the video game console would provide, but on every hardware that this emulator could run on.

Because of this, this project provides an installable system, based on a GNU/Linux distribution, which allows an user to enjoy the same experience that the Wii video game console could offer, on every x86_64 architecture computer.

Through a graphical installer, an user will be able to easily install the system on their computer, being able to install it alongside their current operating system if desired, while keeping all their data intact. After system installation, the system is automatically configured to boot directly to Dolphin emulator, loading the Wii's initial configuration procedure. In subsequent startups, the system will automatically load the Wii menu.

This project provides, as well, a web application accessible through the Wii's web browser, that allows the user to manage the base Linux system, load games stored on a USB stick or even enter the emulator's configuration interface.