



ENTORNOS DE DESARROLLO INTEGRADO

IDEs

LOS IDES

- “Es un programa informático compuesto por un conjunto de herramientas de programación.”

Son herramientas de software diseñadas para proporcionar a los desarrolladores un entorno de trabajo completo y unificado para la creación, edición, depuración y gestión del mismo software. Además, sirven para proporcionar un entorno de trabajo completo y eficiente a los desarrolladores. Estas herramientas integran diversas funciones en una sola interfaz con el objetivo de facilitar y agilizar el proceso de desarrollo del software.

Reseña Histórica

Los IDE fueron posibles cuando se desarrollaba vía consola o terminal de la computadora. Los primeros sistemas no podían soportarlos, porque los programas eran preparados usando diagramas de flujo, introduciendo programas con tarjetas agujeradas (o papel cartón) antes de enviarlos a un compilador. Dartmouth BASIC fue el primer lenguaje en ser creado con un IDE (también fue el primero en ser diseñado para ser utilizado enfrente de la consola o la terminal). Este IDE fue basado en código y basado en comandos, y por esto no se parecía mucho a los IDE tan gráficos actuales. Sin embargo, la edición integrada, manejo de archivos, compilación, depurador y ejecutable en una manera consistente con los IDE modernos.

-
- "*Maestro*" es un producto de Softlab Múnich y fue el primer sistema de desarrollo integrado IDE, para software, creado en 1975. Maestro fue instalado por 22.000 programadores en todo el mundo. Hasta 1989, existían 6000 instalaciones en la República Federal de Alemania. Maestro fue sin duda el líder mundial en este campo durante los 70's y los 80's. Uno de los últimos Maestro puede ser encontrado en el Museo de Tecnología e Informática en Arlington. Uno de los primeros IDE con un concepto de plug-in fue Softbench.

Componentes

Editor de
código
fuente

Compilador

Depurador

Interprete

Cliente

Gestionador
de
proyectos

Funciones
integradas

Características generales de un IDE

Características

Distribución
multiplataforma

Soporte de múltiples
lenguajes de programación

Reconocimiento de sintaxis
(Intellisense)

Capacidades de depuración

Consola de comandos

Control de versiones

Soporte para
temas/personalización

Soporte para
extensiones/complementos

Herramientas especializadas

Soporte para espacios de
trabajo



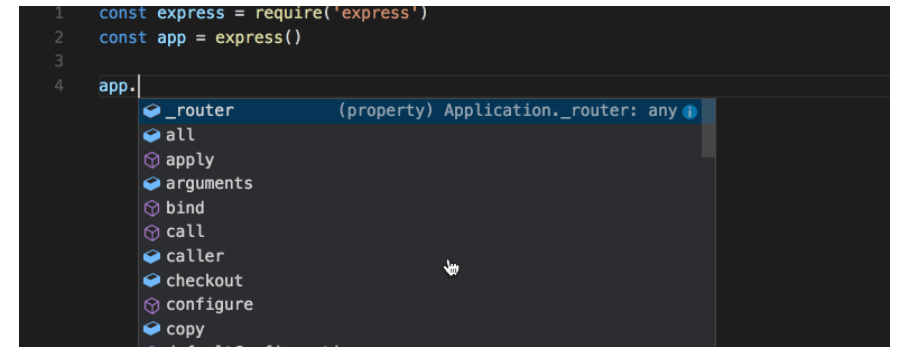
Distribución multiplataforma

Soporte multilenguaje



Reconocimiento de sintaxis

- Identificación del lenguaje
- Detección de errores
- Sugerencias
- Completado opcional de texto
- Comprobación de sintaxis



```
1 const express = require('express')
2 const app = express()
3
4 app.
```

The screenshot shows a code editor with four lines of JavaScript code. The first two lines are `const express = require('express')` and `const app = express()`. The third line is empty, and the fourth line starts with `app.`. A dropdown menu is open below `app.`, displaying a list of suggestions. The first suggestion is `_router` with the text `(property) Application._router: any` next to it. Other suggestions include `all`, `apply`, `arguments`, `bind`, `call`, `caller`, `checkout`, `configure`, and `copy`. Each suggestion is preceded by a small icon: a blue circle with a white dot for `_router`, `all`, `arguments`, `caller`, `checkout`, and `copy`; and a purple hexagon with a white dot for `apply`, `bind`, `call`, `configure`, and `copy`.

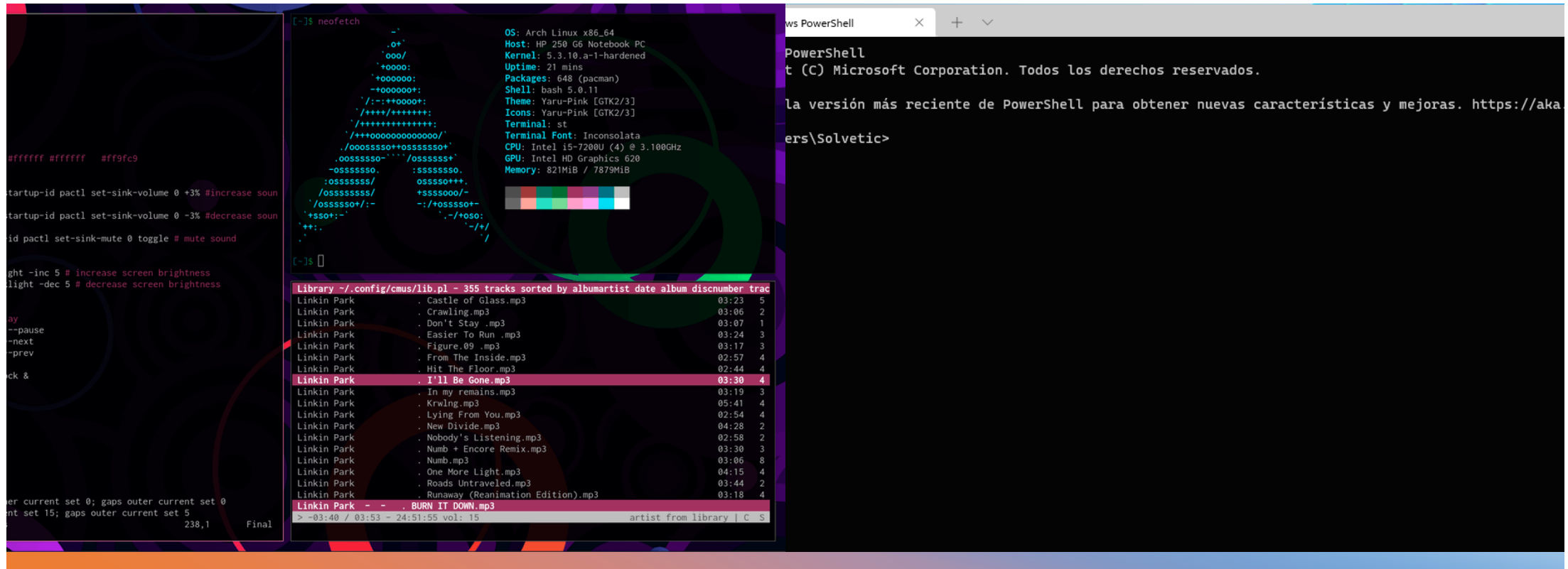


Capacidades de depuración

```
WiFi.cpp x Memory [0x42000800+1 WiFiClass::startProvision.dbgasm
579     sprintf(provSsid, "wifi101-%.2X%",
580             // start provisioning mode
581             startProvision(provSsid, "wifi101
583     }
584     return status();
585 }
586
587 uint8_t WiFiClass::startProvision(const
588 {
589     tstrM2MAPConfig strM2MAPConfig;
590
591     if (!_init) {
592         init();
593     }
594
595     // Enter Provision mode:
596     memset(&strM2MAPConfig, 0x00, sizeof(
597     strcpy((char *)&strM2MAPConfig.au8SSI
598     strM2MAPConfig.u8ListenChannel = chan
599     strM2MAPConfig.u8SecType = M2M_WIFI_S
600     strM2MAPConfig.u8SsidHide = SSID_MODE
601     strM2MAPConfig.au8DHCPServerIP[0] = 1
602     strM2MAPConfig.au8DHCPServerIP[1] = 1
603     strM2MAPConfig.au8DHCPServerIP[2] = 1
604     strM2MAPConfig.au8DHCPServerIP[3] = 1
605
606     if (m2m_wifi_start_provision_mode((ts
607         _status = WL_PROVISIONING_FAILED;
608         return _status;
609     }
610     _status = WL_PROVISIONING;
611     _mode = WL_PROV_MODE;
612
613     memset(_ssid, 0, M2M_MAX_SSID_LEN);
614     memcpy(_ssid, ssid, strlen(ssid));
615     m2m_memcpy((uint8 *)&_localip, (uint8
616     _submask = 0x0FFFFFFF;
617     _gateway = _localip;
618
619     #ifdef CONF_PERIPH
620     // WiFi led ON (rev A then rev B).
621
1  0x000006d4: f0 b5
2  0x000006d6: a3 b0
3  0x000006d8: 04 1c
4  0x000006da: 0d 1c
5  0x000006dc: 16 1c
6  0x000006de: 1f 1c
7  0x000006e0: 00 68
8  0x000006e2: 00 28
9  0x000006e4: 02 d1
10 0x000006e6: 20 1c
11 0x000006e8: ff f7 68 ff
12 0x000006ec: 68 46
13 0x000006ee: 00 21
14 0x000006f0: 88 22
15 0x000006f2: 0a f0 74 f8
16 0x000006f6: 68 46
17 0x000006f8: 29 1c
18 0x000006fa: 0a f0 6c f9
19 0x000006fe: 21 23
20 0x00000700: 69 46
21 0x00000702: cf 54
22 0x00000704: 01 23
23 0x00000706: 3f 22
24 0x00000708: 8b 54
25 0x0000070a: 00 21
26 0x0000070c: 40 22
27 0x0000070e: 68 46
28 0x00000710: 81 54
29 0x00000712: c0 21
30 0x00000714: 41 22
31 0x00000716: 81 54
32 0x00000718: a8 21
33 0x0000071a: 42 22
34 0x0000071c: 81 54
35 0x0000071e: 43 22
36 0x00000720: 83 54
37 0x00000722: 44 22
38 0x00000724: 83 54
39 0x00000726: 31 1c
40 0x00000728: 01 22
41 0x0000072a: 01 f0 e9 f9
42 0x0000072e: 00 28
43 0x00000730: 04 da

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
2: Task - Monitor >
> Executing task: platformio device monitor <

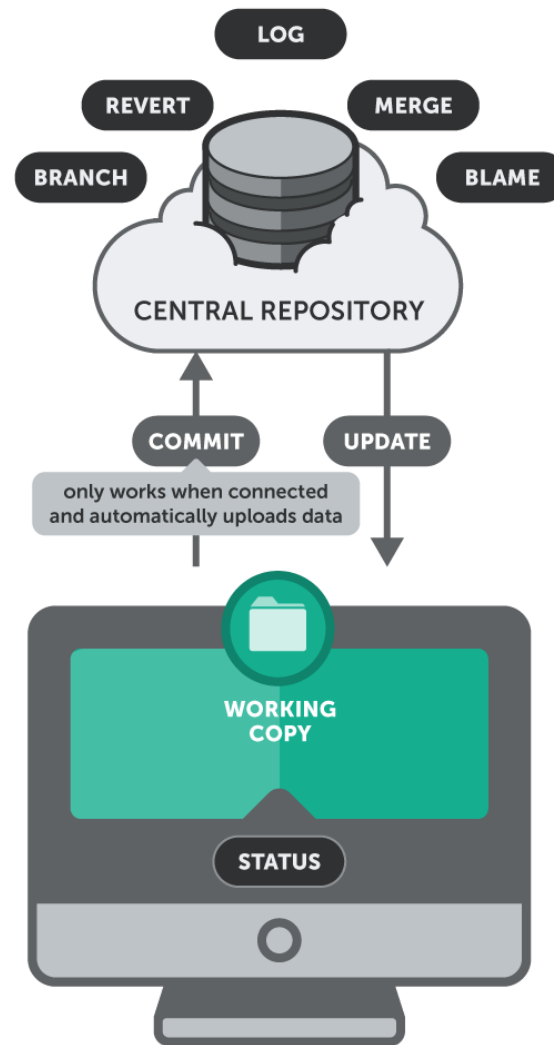
--- Miniterm on /dev/cu.usbmodemFA142 9600,8,N,1 ---
--- Quit: Ctrl+C | Menu: Ctrl+T | Help: Ctrl+T followed by Ctrl+H ---
Configuring WiFi shield/module...
Starting in provisioning mode...
```



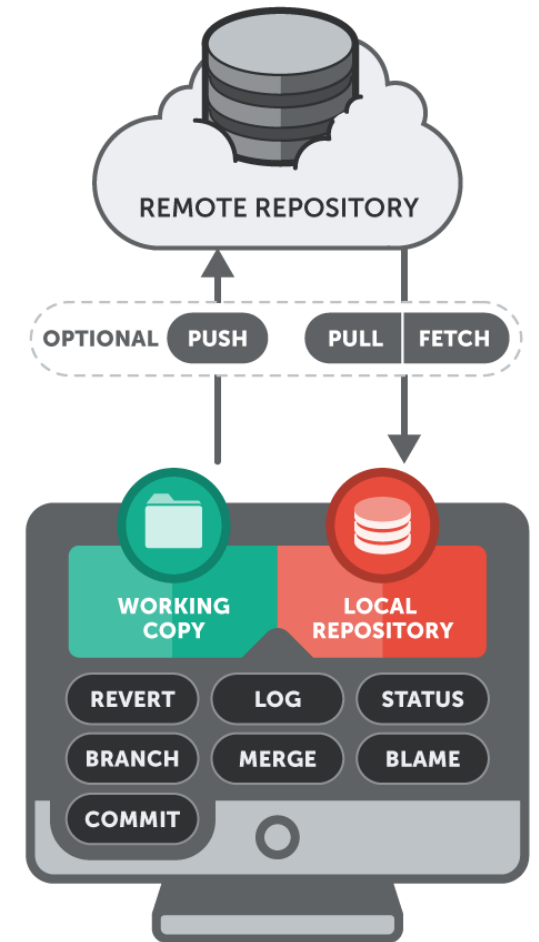
Terminal de comandos

Control de versiones

SUBVERSION



GIT

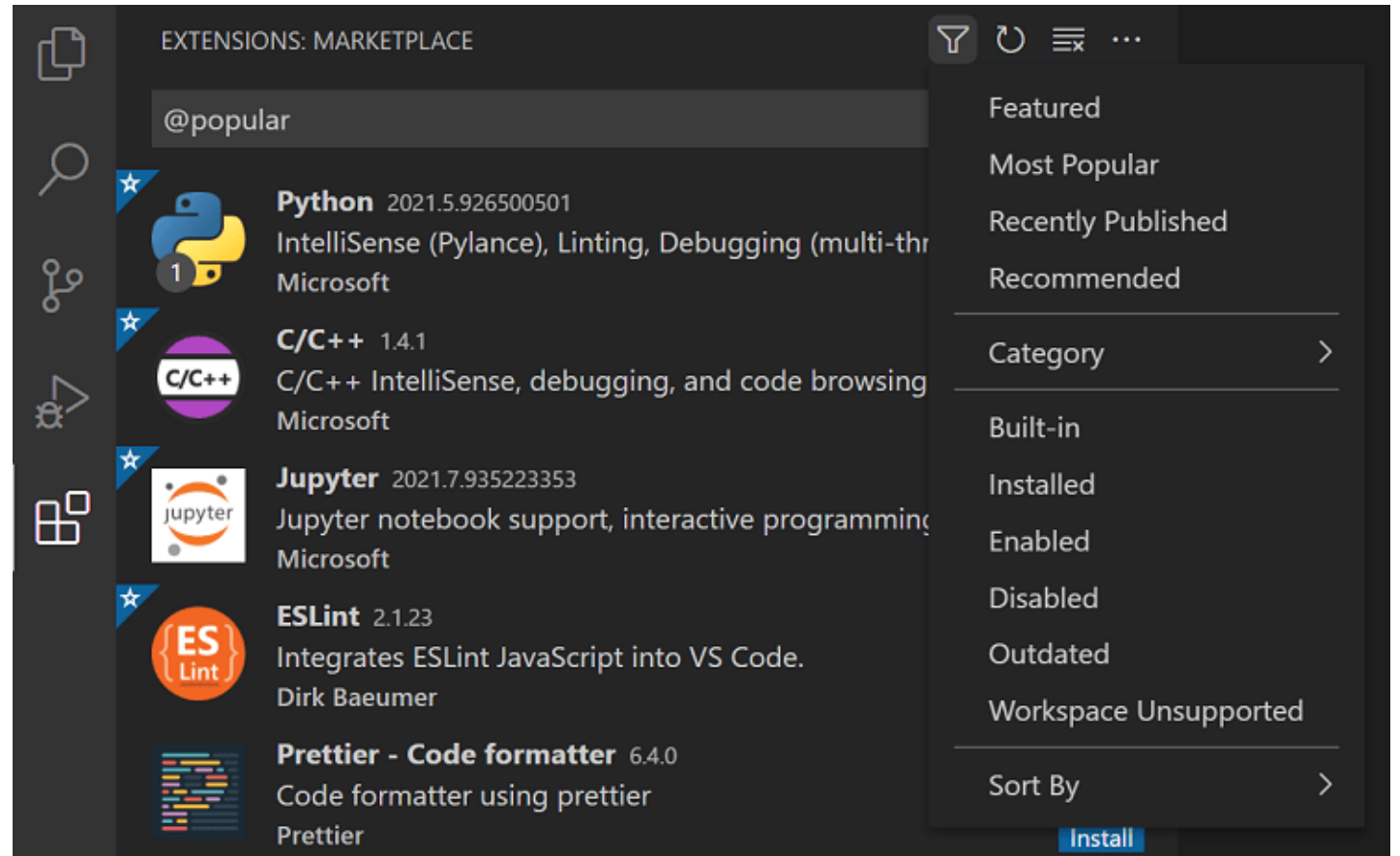



```
plugin > terminator-themes.py > terminator-themes > configure
26 def configure(self, widget, data = None):
27     ui = {}
28     vbox = Gtk.Dialog(_("Terminator themes"), None,
29                       Gtk.DialogFlags.MODAL)
30
31     headers = { "Accept": "application/vnd.github.v3+json" }
32     response = requests.get(self.base_url,
33                             headers=headers)
34
35     if response.status_code != 200:
36         gerr(_("Failed to get list of available themes")
37             )
38         return
39
40     self.themes_from_repo = response.json()["themes"]
41     self.profiles = self.terminal.config.list_profiles()
42
43     main_container = Gtk.HBox(spacing=5)
44     main_container.pack_start(self._create_themes_grid
45                               (ui), True, True, 0) #Left column
46     main_container.pack_start(self._create_settings_grid
47                               (ui), True, True, 0) #Right column
```

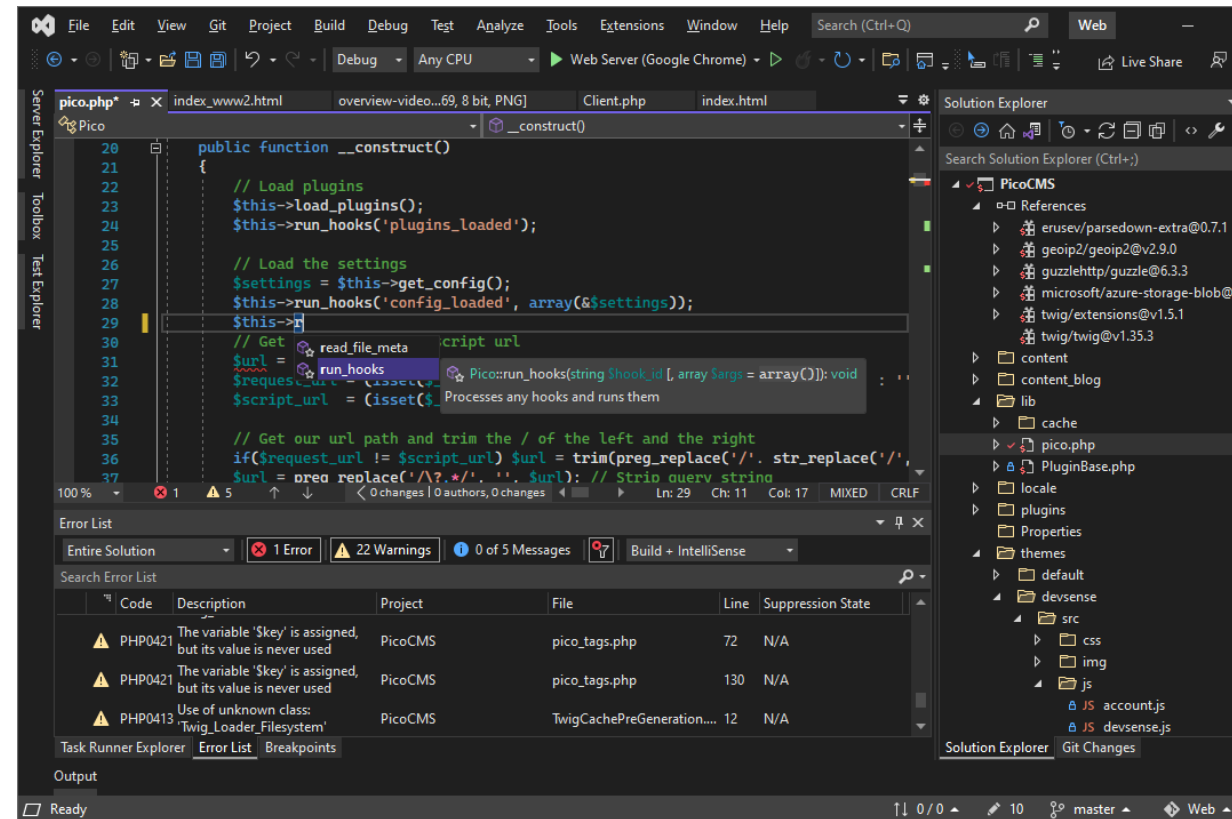
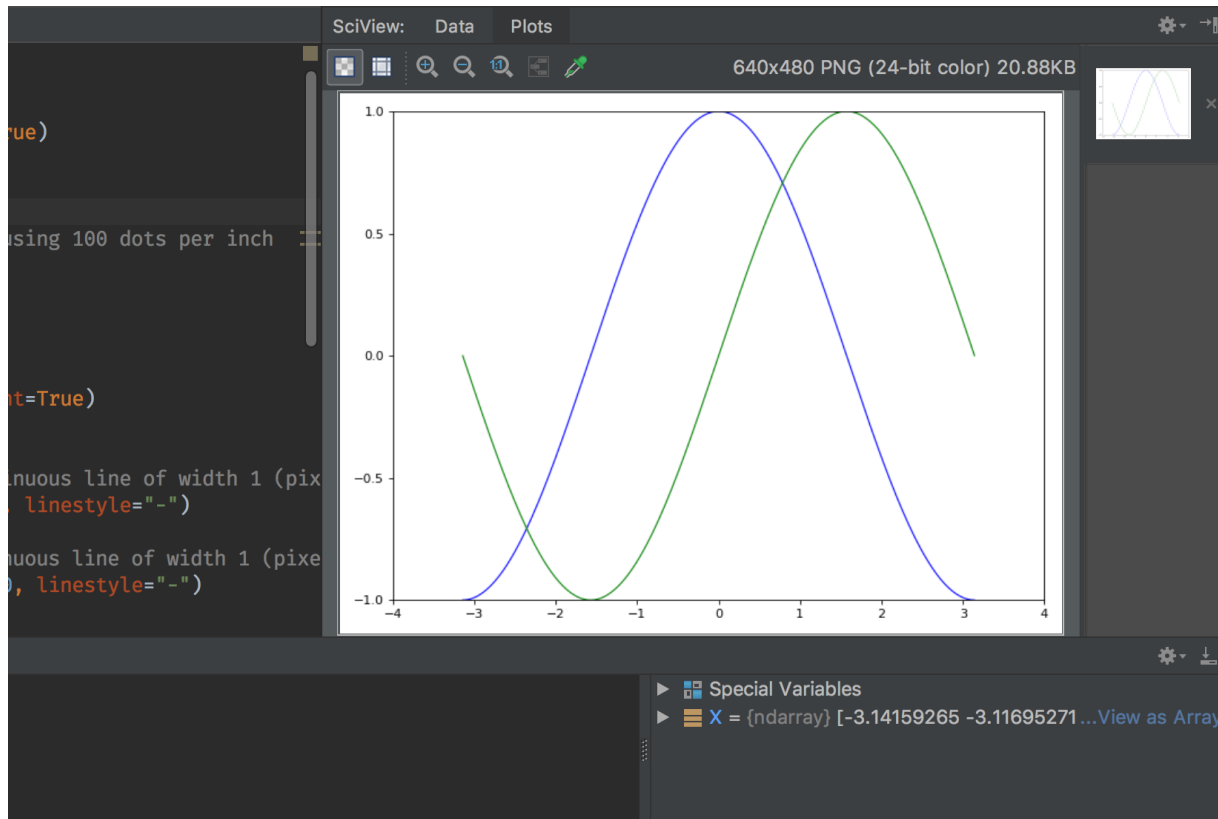
```
.eslintrc.js A.jsx x .npmrc gatsby-config.js
75 * A dynamic and failsafe component which either renders to a base HTML link
76 * `` (anchor) or a "Gatsby Link" based on the target/URL type, internal
77 * or external, passed to the `to` and `href` props.
78 */
79 const A = ({ children, href, to, linkRef, ...passProps }) =>
80   isRouteInternal(to) || isRouteInternal(href) ? (
81     <BaseGatsbyLink innerRef={linkRef} to={to || href} {...passProps}>
82       {children}
83     </BaseGatsbyLink>
84   ) : (
85     <BaseComponent ref={linkRef} href={href || to} target="_blank" {...passProps}>
86       {children}
87     </BaseComponent>
88   );
89
90 export default React.forwardRef(({ children, ...passProps }, ref) => (
91   <A linkRef={ref} {...passProps}>
92     {children}
93   </A>
94 ));
```

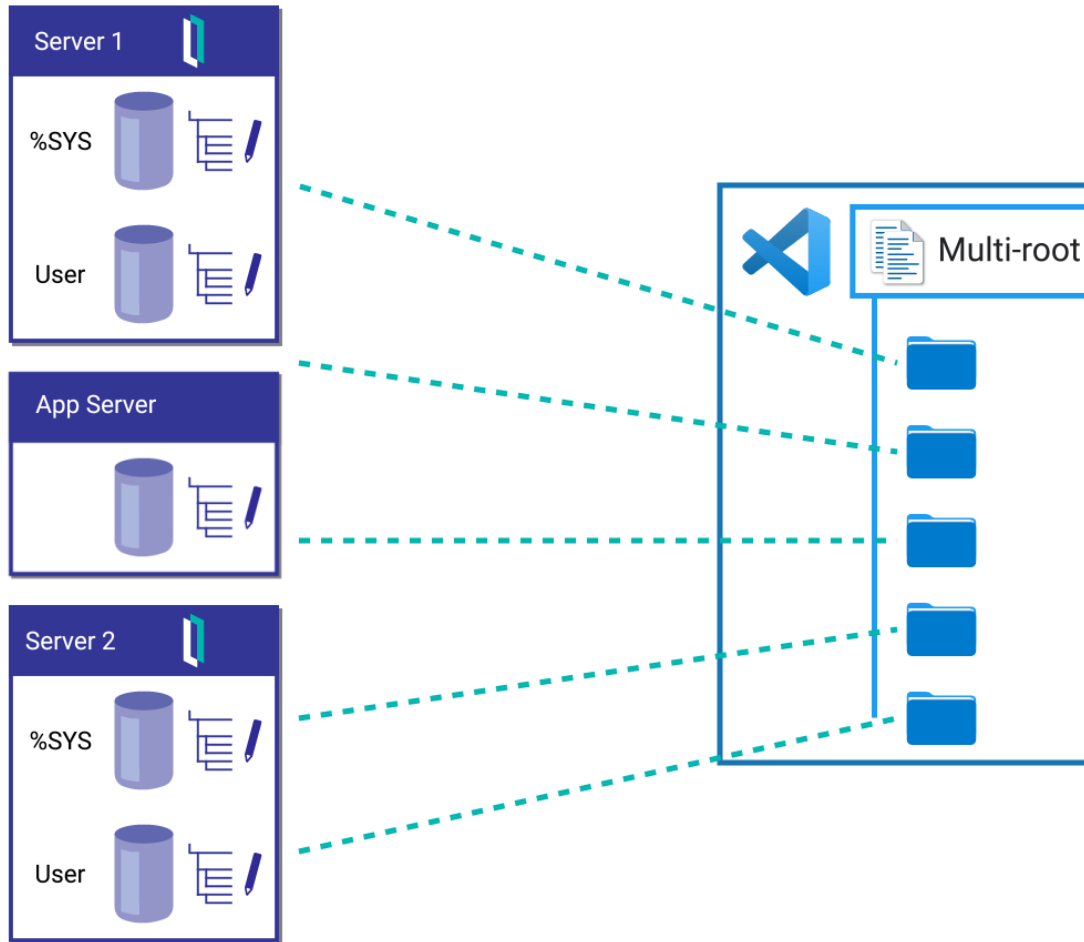
Soporte para temas/personalización

Soporte para extensiones



Herramientas especializadas





Espacios de trabajo

- Agrupan uno o varios proyectos en un mismo directorio
- Automatizan el tener que iniciar el trabajo
- Guardan configuraciones
- Guardan variables del entorno
- Agilizan el proceso de desarrollo



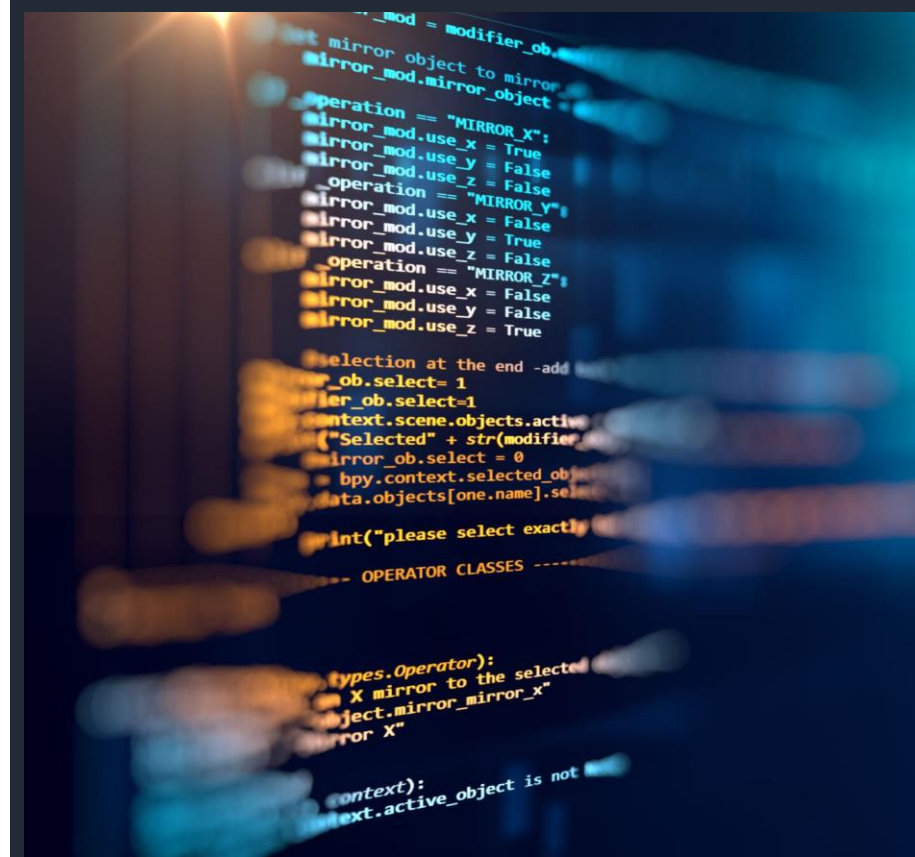
Tipos de IDEs

Mas populares

Los desarrolladores instalan y ponen en marcha IDE locales directamente en sus equipos locales. También tienen que descargar e instalar varias bibliotecas adicionales dependiendo de sus preferencias de codificación, los requisitos del proyecto y el lenguaje de desarrollo. Mientras que las IDE locales son personalizables y no necesitan de una conexión a Internet una vez instalados, presentan varios desafíos, como los que se muestran a continuación:

- Pueden consumir mucho tiempo y su configuración puede resultar difícil.
- Consumen recursos de equipos locales y pueden ralentizar el rendimiento de los equipos de forma significativa.
- Las diferencias de configuración entre el equipo local y el entorno de producción pueden generar errores en el software.

IDE locales



IDE en la nube

- Los desarrolladores utilizan IDE en la nube para escribir, editar y compilar código directamente en el navegador para prescindir de la necesidad de descargar software en sus equipos locales. Los IDE basados en la nube ofrecen varias ventajas en comparación con los IDE tradicionales. Estos son algunos de dichas ventajas:

Entorno de desarrollo estandarizado

Los equipos de desarrollo de software pueden configurar un IDE basado en la nube de forma central para crear un entorno de desarrollo estándar. Este método los ayuda a evitar errores que pueden ocurrir debido a las diferencias en las configuraciones de los equipos locales.

Independencia de plataformas

Los IDE en la nube funcionan en el navegador y son independientes de los entornos de desarrollo locales. Esto significa que se conectan directamente a la plataforma de la nube del vendedor y los desarrolladores pueden utilizarlos desde cualquier equipo.

Mejor rendimiento

Crear y compilar funciones en un IDE requiere de una gran capacidad de memoria, lo que puede ralentizar el equipo del desarrollador. El IDE en la nube utiliza recursos de cómputo de la nube y libera los recursos del equipo local.



Ventajas de Utilizar los IDE

Beneficios de los Entornos de Desarrollo Integrados
(IDE)



Inicio Rápido

- Los IDE permiten a los desarrolladores comenzar a programar aplicaciones nuevas con rapidez, eliminando la necesidad de configurar herramientas manualmente.





Eficiencia

- Todas las herramientas necesarias están integradas en un mismo entorno, lo que ahorra tiempo y aumenta la eficiencia.



Facilidad de Aprendizaje

- Los nuevos desarrolladores pueden ponerse al día rápidamente al usar un IDE, ya que todas las herramientas son familiares y están disponibles en un solo lugar.



Características de Ahorro de Tiempo

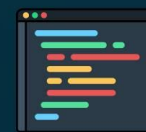
- Los IDE ofrecen funciones como relleno inteligente y generación automatizada de código, lo que acelera el desarrollo y reduce la escritura manual de código.

Resumen

- En la mayoría de los casos, los equipos de desarrollo optan por IDE preconfigurados. La eficiencia, la estandarización y la automatización que ofrecen los IDE modernos superan las consideraciones de personalización

EDITOR VS IDE

Con ambos puedes escribir código, pero ¿en qué se diferencian?



Software ligero con ayudas para escribir código (resaltado de sintaxis, autocompletado, etc).



Integra un editor con las herramientas que necesitas como desarrollador (debugger, compilador, etc).



Soporta **múltiples lenguajes** y tecnologías.



Enfocado en archivos (no tienen el concepto de proyecto).



Puedes agregar plugins para darle el poder de un IDE pero te toca configurar cada uno a mano.



Se especializa en **un lenguaje o tecnología** (Java, Python, Go, Android, etc).



Enfocado en proyectos completos. Desde la primera línea hasta la salida a producción.



Trae **herramientas integradas y configuradas** (ej. Android Studio trae un emulador de Android).


EJEMPLOS DE EDITORES



EJEMPLOS DE IDES



Domina la tecnología con EDteam y #NuncaTeDetengas

 ed.team/cursos





Conclusion