

Módulo 1

LECCIÓN 3

IDEs y versionamiento

IDEs y versionamiento

Para la sesión actual se busca que los estudiantes logren instalar el **IDE** con el que se desarrolla el nivel, sin ser el único disponible ni obligatorio pero resaltar que, si todos usan las mismas herramientas, el apoyo es más sencillo, ya que se genera una comunidad en torno a ella.

Para avanzar al proceso de instalación primero es necesario conocer qué es un IDE, un editor de código y algunas de las opciones más utilizadas.

¿Qué es y para qué sirve un IDE?

Un entorno de desarrollo integrado (IDE por sus siglas en inglés) es un sistema de software para el diseño de aplicaciones que combina herramientas del desarrollador comunes en una sola interfaz gráfica de usuario (GUI). Generalmente, **un IDE cuenta con las siguientes características:**



Editor de código fuente



Automatización de las compilaciones locales



Depurador

Algunos IDE son compatibles con un solo lenguaje, así que son mejores para un modelo de programación específico. Por ejemplo, IntelliJ es conocido principalmente como un IDE de Java. Otros IDE admiten una gran variedad de lenguajes de manera conjunta, como el IDE Eclipse que admite Java, XML, Python, entre otros.



Programa que sirve para probar otros programas y mostrar la ubicación de un error en el código original de forma gráfica.



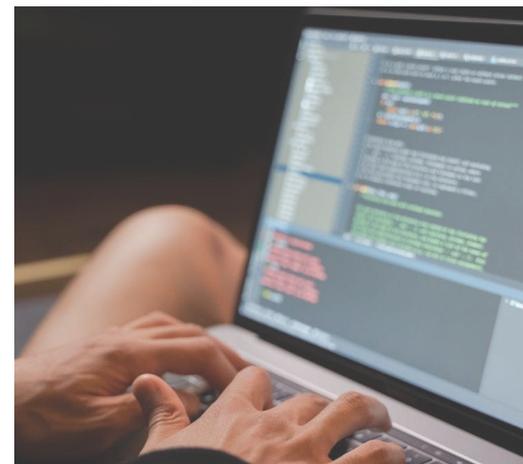
herramientas que automatizan las tareas sencillas y repetitivas como parte de la creación de una compilación local del software para que use el desarrollador, como la compilación del código fuente de la computadora en código binario, el empaquetado de ese código y la ejecución de pruebas automatizadas.



Editor de texto que ayuda a escribir el código de software con funciones como el resaltado de la sintaxis con indicaciones visuales, el relleno automático específico para el lenguaje y la comprobación de errores a medida que se escribe el código.

Algunos IDE más populares entre los programadores son del enlace 16:

- Visual Studio Code
- Sublime Text
- Atom
- Vim
- Eclipse
- Android Studio
- NetBeans
- JetBrains IntelliJ



A continuación, se procede a la instalación de Visual Studio Code y a la configuración de algunos aspectos como instalación de extensiones, paquetes de íconos y cambiar el tema. Se hace un recorrido con los estudiantes a través de las diferentes partes de la herramienta, la barra de opciones, el menú de configuración, algunas teclas de acceso rápido.

Se recomienda algún video básico o intermedio de configuración rápida para que los estudiantes puedan consultarlo por fuera de la clase, por ejemplo: [▶ Play](#)

O un curso un poco más largo como lo es: [▶ Play](#)

Finalmente, aprovechando que se está instalando herramientas, se deja lista la instalación de git desde su web: [🔗 Enlace](#)

- Por ahora solo interesa realizar el procedimiento de instalación inicial con los
- estudiantes para hacer configuraciones básicas desde el instalador como renombrar la
- rama master a main por defecto, esto evitará conflictos más adelante cuando se
- comience a integrar con otras herramientas como **github**.



Con las dos herramientas instaladas y configuradas, el docente puede realizar un breve repaso con los estudiantes y apoyar a los que se queden colgados en la instalación, con el proceso.

```
3 require File.expand_path("../config/application.rb", __FILE__)
4 # Prevent database truncation if the environment is production
5 abort("The Rails environment is running in production mode!")
6 require "spec_helper"
7 require "rspec/rails"
8
9 require "capybara/rspec"
10 require "capybara/rails"
11
12 Capybara.javascript_driver = :webkit
13 Category.delete_all; Category.create
14 Shoulda::Matchers.configure do |config|
15   config.integrate do |with|
16     with.test_framework :rspec
17     with.library :rails
18   end
19 end
20
21 # Add additional requires like this: require 'faker'
22
23 # Requires supporting ruby files with spec_helper, rails_helper,
24 # spec/support/* and its subdirectories. Files matching *_spec*_
25 # in _spec.rb will both be required by default. This can be
26 # run twice. It is recommended to do it only once.
27 # end with _spec.rb. You can configure this behavior like
28 # this:
29 # require_relative "spec_helper"
30 # end
31
32 # Results for 'request'
```



Enlaces





TIC

▶ TALENTO
TECH

AZ | PROYECTOS
EDUCATIVOS

